A CUBICAL LANGUAGE FOR BISHOP SETS

JONATHAN STERLING, CARLO ANGIULI, AND DANIEL GRATZER

Carnegie Mellon University
e-mail address: jmsterli@cs.cmu.edu

Carnegie Mellon University
e-mail address: cangiuli@cs.cmu.edu

Aarhus University
e-mail address: gratzer@cs.au.dk

ABSTRACT. We present XTT, a version of Cartesian cubical type theory specialized for Bishop sets à la Coquand [Coq17], in which every type enjoys a *definitional* version of the uniqueness of identity proofs. Using cubical notions, XTT reconstructs many of the ideas underlying Observational Type Theory [AM06, AMS07], a version of intensional type theory that supports function extensionality. We prove the canonicity property of XTT (that every closed boolean is definitionally equal to a constant) by Artin gluing.

1. INTRODUCTION

Little attention has been paid to notions of liberty and fraternity in dependent type theory, but the same cannot be said about equality. Why? To define the typing judgment $a : A$ we must determine which types are equal—because terms of type A may be cast (coerced) to any type A' equal to A—but in the presence of dependency, equality of types is contingent on equality of terms. In this way, dependency transmutes term equality from a purely semantic consideration to a core aspect of syntax.

As a practical matter, it is desirable to automate as many of these coercions as possible. To that end, type theorists have spent decades refining decision procedures for type equality modulo e.g., α-, β-, δ-, and certain η-laws [Coq91, Coq96, SH00, HP05, SH06, ACD08, ACP09, AS12, Abe13, AOV17, GSB19]. Unfortunately, not all desirable coercions can be automated—for instance, mathematical equality of functions $\mathbb{N} \to \mathbb{N}$ is famously undecidable. The collection of *automated* equations in a given type theory is called *definitional equality*;[1] the more equations are definitional, the less time users must spend providing coercions.

Key words and phrases: cubical type theory, Bishop sets, proof irrelevance, Artin gluing, logical predicates.

[1]Historically, philosophical considerations have motivated explanations of *definitional equality* as a scientific concept independent of a specific theory, sometimes leading to a notion of "equality" that is not a congruence (e.g., not respected by λ) [ML75a]; we argue that our theory-specific notion, based on the phenomenal aspect of automated conversion, is more reflective of everyday practice. As a programmatic matter, we moreover rule out any kind of "equality" for which the operations of type theory are not functional.

Preprint submitted to
Logical Methods in Computer Science

Next, one must determine which coercions, if any, are recorded in *terms*. A priori, which coercions are recorded is independent of which equations are definitional, but in practice these considerations are inextricably linked—silent coercions along non-definitional equations typically disrupt the aforementioned decision procedures. The canonical example is extensional type theory [Hof97], in which one can silently coerce terms from any type to any other under a contradictory assumption. In particular, elements of A and $A \to A$ are identified in context $x : 0$, allowing users to encode fixed point combinators; as a result, any decision procedure which relies on β-reduction will no longer terminate.

A third consideration is the relationship between judgmental and propositional equality. Following Martin-Löf [ML87], type theorists arrange concepts of interest into judgments, or top-level forms of assertion, such as typehood (A type), membership ($a : A$), entailment ($\Gamma \vdash \mathcal{J}$), and *judgmental equality* of types ($A = A'$ type) and terms ($a = a' : A$). To account for higher-order concepts, rather than admit higher-order judgments, we usually internalize judgmental notions as types: dependent sums internalize context extension, dependent products internalize entailment, and *propositional equality* should in some sense internalize judgmental equality.

1.1. **Notions of equality in type theory.** In the past fifty years, researchers have considered myriad presentations of equality in type theory. Almost always, judgmental equality is a congruence (reflexive, symmetric, transitive, and respected by all type and term formers) along which coercion is silent, expressed by the *conversion* rule that if $\Gamma \vdash A = A'$ *type* and $\Gamma \vdash a : A$ then $\Gamma \vdash a : A'$. However, formulations of coercion, definitional equality, and propositional equality differ widely; we proceed by outlining several existing approaches.

1.1.1. *Equality reflection.* The simplest way to internalize judgmental equality as a type is to provide introduction and elimination rules making the existence of a proof of $\mathsf{Eq}_A(a, a')$ equivalent to the judgment $a = a' : A$:

$$\frac{\Gamma \vdash a = a' : A}{\Gamma \vdash \mathsf{refl} : \mathsf{Eq}_A(a, a')} \qquad \frac{\Gamma \vdash p : \mathsf{Eq}_A(a, a')}{\Gamma \vdash a = a' : A}$$

The elimination rule above is known as *equality reflection*, and is characteristic of extensional versions of type theory [ML84]. Reflection immediately endows $\mathsf{Eq}_A(a, a')$ with many desirable properties: it is automatically a congruence, admits coercion (via conversion), and enjoys *uniqueness of identity proofs* (UIP, that any two elements of $\mathsf{Eq}_A(a, a')$ are equal).

Unfortunately, because propositional equality is undecidable, equality reflection causes all the judgments of type theory to become undecidable; worse, as hinted previously, even the "definitional fragment" of the resulting judgmental equality can no longer be automated, because β-reduction of open terms may diverge.[2] Therefore, proof assistants for extensional type theories cannot support type checking, and rely instead on tactic-based construction of typing derivations.

One exemplar of this approach is the Nuprl proof assistant [CAB+86] along with its descendants, including MetaPRL [Hic01] and RedPRL [ACH+18b]. These type theories are designed as "windows on the truth" of a single intended semantics inspired by Martin-Löf's computational meaning explanations, interpreting types as partial equivalence relations over

[2] If, on the other hand, one omits β-equivalence from judgmental equality, it is possible to retain decidability of judgments in a dependently-typed language with divergent terms, as in the Zombie language [SW15].

untyped terms [ML79, All87]. Nuprl-family proof assistants employ a host of reasoning principles not validated by other models of type theory, including intuitionistic continuity principles [RB16], computational phenomena such as exceptions and partiality [Cra98], and "direct computation" rules which use untyped rewrites to establish well-formedness subgoals.

As a consequence of its fundamentally untyped nature, formalizing a theorem in Nuprl does not imply the correctness of the corresponding theorem in standard classical mathematics (the global mathematics of constant or discrete sets), nor even in most forms of constructive mathematics (the local mathematics of variable and cohesive sets). It is worth noting that the problem is *not* located in the presence of anti-classical principles (which are interpretable in logic over a variety of topoi), and rather arises from the commitment to untyped ontology.

The creators of the Andromeda proof assistant [BGH+16] have introduced another approach to implementing equality reflection, in which judgmental equality is negotiated by means of algebraic effects and handlers [BP15]; in essence, handlers allow users to provide the out-of-band proofs of judgmental equality that are present in the derivations (but not the terms) of extensional type theory. In contrast to Nuprl, a proof formalized in Andromeda can be seen to imply the corresponding informal statement in any variety of classical or constructive mathematics, a consequence of the fact that an interpretation of Andromeda's extensional type theory may be found lying over any topos.

1.1.2. *Intensional type theory.* The *identity type* of intensional type theory (ITT) [ML75b, NPS90] offers a much more restrictive internalization of judgmental equality, characterized by the following rules:

$$\frac{\Gamma \vdash a : A}{\Gamma \vdash \mathsf{refl}(a) : \mathsf{Id}_A(a,a)} \qquad \frac{\begin{array}{l}\Gamma, x : A, y : A, z : \mathsf{Id}_A(x,y) \vdash C(x,y,z)\ \mathit{type} \\ \Gamma \vdash p : \mathsf{Id}_A(a,a') \qquad \Gamma, x : A \vdash c : C(x,x,\mathsf{refl}(x))\end{array}}{\Gamma \vdash \mathsf{J}_{x,y,z.C}(p; x.c) : C(a,a',p)}$$

$$\frac{}{\Gamma \vdash \mathsf{J}_{x,y,z.C}(\mathsf{refl}(a); x.c) = [a/x]c : C(a,a,\mathsf{refl}(a))}$$

The elimination form J endows $\mathsf{Id}_A(a,a')$ with many properties expected of an equality connective, including symmetry, transitivity, and coercion, which can be defined as follows:

$$\mathsf{coerce}\ (p : \mathsf{Id}_{\mathcal{U}}(A,B)) : A \to B \stackrel{\mathit{def}}{=\!=} \mathsf{J}_{x,y,z.x \to y}(p; \_.\lambda x.x)$$

The tradeoffs of ITT are well-understood. By requiring explicit coercion for non-$\alpha/\beta/\delta/\eta$ equations, ITT presents a theory with decidable judgments. In practice, however, explicit coercions accumulate in types and terms, requiring even more explicit coercions to mediate between previously-used coercions. This is because coercions simplify only when applied to identity proofs of the form $\mathsf{refl}(A)$; a coercion may mediate between definitionally equal types and nevertheless fail to reduce (e.g., for a variable of type $\mathsf{Id}_{\mathcal{U}}(A,A)$).

In addition to these practical considerations, identity types also fail to evince several properties which some may expect of an equality connective, such as UIP and function extensionality, the principle that $(x : A) \to \mathsf{Id}_B(f(x), g(x))$ implies $\mathsf{Id}_{A \to B}(f,g)$ [Str94, HS98]. In light of homotopy type theory [Uni13], it is reasonable to consider an equality connective without UIP, but theorists and practitioners alike generally agree that function extensionality is desirable. These shortcomings are sometimes addressed by adjoining axioms for function extensionality or UIP (or univalence), but axioms in the identity type cause even more coercions to become irreducible.

1.1.3. *Setoids.* Another way to avoid the shortcomings of identity types in ITT is to work in *setoids* [Hof95], or Bishop sets [Bis67], an exact completion which replaces types by pairs of a carrier type $|A|$ and a type-valued "equivalence relation" $=_A$. Each type former is lifted to setoids extensionally: the setoid of functions $(|A|, =_A) \to (|B|, =_B)$ consists of functions $f : |A| \to |B|$ equipped with proofs $f_= : (x, y : |A|) \to x =_A y \to f(x) =_B f(y)$ that they respect equivalence.

The framework of setoids allows users of type theory to ensure their constructions are appropriately extensional, at the cost of manually proving those conditions. In contrast, respect for the identity type is automatic (by its elimination principle) but insufficiently powerful to imply function extensionality. An ideal treatment of equality should, unlike setoids, take advantage of the fact that constructions in type theory *do* respect function extensionality; syntactically well-behaved examples of this approach include Observational Type Theory, cubical type theory, and XTT, discussed below. Another recent proposal is to translate a type-theoretic language, *setoid type theory*, into setoids in ITT; however, it is unknown whether this language is complete or enjoys good syntactic properties [ABKT19].

1.1.4. *Observational Type Theory.* The first systematic account of extensional equality types in intensional type theory was *Observational Type Theory* (OTT) [AM06, AMS07], which built on earlier work by Altenkirch and McBride [Alt99, McB99]. The main idea of OTT is to consider a closed (inductive-recursive) universe of types, and to define propositional equality and its operations by recursion on type structure. Concretely, for any two types A, B there is a type of proofs that A equals B, and a coercion operation sending these proofs to functions $A \to B$; then, for any $a : A$ and $b : B$, there is a type of proofs that a heterogeneously equals b, and a coherence operation stating that terms heterogeneously equal their coercions. Propositional equality in OTT satisfies a definitional form of UIP.

Because OTT's equality types are defined by recursion, they will unfold into complex types (reminiscent of the equality relations on setoids) when sufficiently specialized; in XTT, path types do not unfold, but are easily characterized when necessary. In both OTT and XTT, however, any algorithm to check definitional equality of coercions must rely on the ability to "do typecase" on elements of the universe. Although typecase is acceptable or even desirable in programming [Con82, CZ84, HM95, Dag13], its presence rules out any interpretation of the universe as a mathematical universe (i.e., a family which (weakly) classifies all small families).

Observational Type Theory also pioneered the idea that coercions should compute on non-reflexive proofs of equality, a design principle which also plays a significant role in the usability of cubical path types in contrast to standard identity types. Recently, McBride and collaborators have made progress toward a cubical version of OTT based on a different cube category and coercion operation than the ones considered in XTT [CFM18].

1.1.5. *Cubical type theory.* Homotopy type theory arises from the observation that ITT is compatible with Voevodsky's *univalence axiom*, which states that every equivalence (coherent isomorphism) between types A and B gives rise to an identity proof $\mathsf{Id}_{\mathcal{U}}(A, B)$ [KL16, Uni13]. Univalence solves longstanding difficulties with type-theoretic universes—transforming them into object classifiers in the sense of higher topos theory [Lur09]—and contradicts UIP because two types can be equivalent in several inequivalent ways. Unfortunately, adding

univalence to ITT as an axiom results in coercions that are "stuck" on non-reflexive identity proofs, as in the case of adding axioms for function extensionality or UIP.

To address this problem, researchers have developed a number of *cubical type theories* [CCHM17, ABC$^+$19, AHH18] whose propositional equality and coercion operations support univalence in a computationally well-behaved way. The core idea is to introduce a judgmental notion of equality proof which is then internalized as the *path type*.

Concretely, cubical type theories extend type theory with an abstract interval $\mathbb{I}$ populated by *dimension variables* $i : \mathbb{I}$ and constant endpoints $0, 1 : \mathbb{I}$. A type parametrized by a dimension variable $i : \mathbb{I} \vdash A$ *type* represents a proof that $[0/i]A$ and $[1/i]A$ are equal types, and a term $i : \mathbb{I} \vdash a : A$ is a heterogeneous equality proof between $[0/i]a : [0/i]A$ and $[1/i]a : [1/i]A$. Coercion in cubical type theory is a primitive operation which computes based on the structure of the proof $i : \mathbb{I} \vdash A$ *type*, and admits an OTT-style "coherence" operation as a special case. Because cubical type theory defines propositional equality in A using parametrized elements of A, propositional equality automatically inherits the properties of each type and therefore satisfies function extensionality and related principles.

There are several versions of cubical type theory. De Morgan cubical type theory [CCHM17], a variant of which is implemented in Cubical Agda [VMA19], additionally equips $\mathbb{I}$ with negation and binary minimum and maximum operations. Cartesian cubical type theory [ABC$^+$19, AHH18], implemented in the RedPRL [ACH$^+$18b] and `redtt` [ACH$^+$18a] proof assistants, imposes no further structure on $\mathbb{I}$ but requires a stronger coercion operation. In addition, Awodey, Cavallo, Coquand, Riehl, and Sattler [Rie19] have recently proposed an *equivariant* Cartesian cubical type theory which is homotopically well-behaved, and Cavallo, Mörtberg, and Swan [CMS20] have developed a common generalization of the De Morgan and Cartesian type theories.

1.1.6. *Our contribution: XTT.* We describe XTT, a type theory without equality reflection whose propositional equality connective satisfies function extensionality and definitional UIP. XTT was introduced in a preliminary version of this work, which appeared in the 4th International Conference on Formal Structures for Computation and Deduction under the title *Cubical Syntax for Reflection-Free Extensional Equality* [SAG19].

Using ideas from Cartesian cubical type theory, XTT reconstructs the decisive aspects of OTT in a more modular, judgmental fashion. For instance, instead of defining equality separately at each type, we define path types uniformly in terms of dimension variables; similarly, we impose UIP by means of a *boundary separation* rule which does not mention path types.

Compared to other cubical type theories [CCHM17, ABC$^+$19, AHH18], XTT has clear advantages and disadvantages. Whereas other cubical type theories have two separate connectives for path types and identity types, XTT's path types strictly satisfy the rules of Martin-Löf's identity types definitionally. In addition, the rules governing composition—the most complex rules of every cubical type theory—are substantially simpler in XTT than in Cartesian cubical type theory. On the other hand, these simplifications are only possible because XTT is concerned with *Bishop sets*, a specific kind of cubical set analogous to a setoid [Coq17], of which univalent universes and higher inductive types are not instances.

In addition to the XTT calculus, our second contribution is an abstract canonicity proof for XTT using the language of categorical gluing, summarized in Section 1.2.3.

1.2. **Metatheory.** Type checkers rely on global invariants of type theories that are easily disrupted—indeed, we have already seen that the equality reflection rule single-handedly destroys type checking. Consequently, type theorists devote much effort to proving that various calculi are well-behaved, in the form of the *canonicity* and *normalization* metatheorems.

Canonicity states that any closed term of boolean (or natural number) type is judgmentally equal to either tt or ff (resp., a numeral). Canonicity expresses a weak form of completeness for base types which is analogous to the existence property of intuitionistic logic [Tv88]. Although most type theories (including ETT, ITT, OTT, and XTT) enjoy canonicity, it can fail when a type theory is extended by a new construct whose behavior is not sufficiently determined by new equations, as in the extension of ITT by function extensionality or univalence axioms. Indeed, the main motivation behind cubical type theory was to develop a univalent type theory satisfying canonicity.

Normalization is a generalization of canonicity to open terms which characterizes the open terms of every type up to judgmental equality. These characterizations can be quite complex: the normal forms of boolean type include constants tt and ff, variables $x : \mathsf{bool}$, projections of variables $x : \mathsf{bool} \times \mathsf{bool}$, etc. Unlike canonicity, normalization does not measure the strength of judgmental equality: normalization theorems can hold for ITT extended with axioms, and hold trivially if one limits judgmental equality to α-equivalence.

Conversely, while a failure of canonicity may indicate that judgmental equality is too weak, a failure of normalization usually indicates that judgmental equality is altogether intractable. Consider the judgmental injectivity of type constructors, a consequence of normalization: if $\Gamma \vdash A \to B = A' \to B'$ *type* then $\Gamma \vdash A = A'$ *type* and $\Gamma \vdash B = B'$ *type*. Injectivity is crucial for type checking because it enables the well-typedness of the application of $f : A \to B$ to $a : A'$ to be reduced to checking whether $A = A'$. A priori, two function types may be equal because both are equal to a third type C by a sequence of β/η equalities; ruling this out generally requires a full characterization of equality via normalization.

1.2.1. *Categories of models.* The rules of type theory are a complex mutual definition and simultaneous quotient of the collections of contexts, types, and terms. Type theorists often make such definitions precise by passing to a specialized setting known as a *logical framework*, offloading the bureaucratic aspects of the theory, including in many cases the treatment of variable binding and hypothetical judgment (as in the Edinburgh Logical Framework [HHP93]), but far more importantly, the compatibility of every operation with definitional equality (as in Martin-Löf's Logical Framework [NPS90] and Cartmell's generalized algebraic theories [Car86]).

Recall that canonicity and normalization only hold for the *smallest* type theory generated by a collection of rules, as they may be easily refuted by adding rules. Thus, for the purposes of metatheory, the construction of a type theory must come equipped with an *induction principle* stating in what sense it is the smallest. These induction principles are in fact up for debate, as choosing an induction principle is tantamount to fixing the range of possible interpretations of the syntax. For example, in an ELF encoding of type theory, judgmental equality can have no special status and therefore admits non-trivial interpretations, whereas mathematicians generally require that it be interpreted as mathematical equality.

In the language of category theory, these considerations amount to specifying a category of models of a type theory and exhibiting an initial object in that category. Luckily, the rules of XTT are sufficiently non-exotic as to allow us to obtain its functorial semantics

by appealing to general existence theorems [Car86, KKA19, Uem19], thereby sidestepping more general considerations raised by Voevodsky in his famous initiality conjecture [Voe16].

In previous work [SAG19], we specified an early version of XTT's semantics by regarding XTT as a generalized algebraic theory (GAT) in the sense of Cartmell [Car86]. Models of GATs determine choices of objects up to equality, and morphisms of models preserve these choices strictly; models of the GAT of type theory thus determine *up to isomorphism* a category of contexts, and *up to equality* the context extension. This is a much stronger notion of "category" than can be comfortably manipulated using the language of category theory: universal properties determine object-level structure only up to canonical isomorphism, and category-level structure only up to categorical equivalence ("isomorphism up to isomorphism"). Accordingly, in the GAT discipline, one cannot usually general existence theorems but must instead provide explicit constructions in order to strictly determine and preserve object-level structures.

We advocate for a more categorical viewpoint, in which morphisms of models preserve structures only up to coherent isomorphism. In this paper, following Sterling and Angiuli [SA20], we instantiate Uemura's framework [Uem19] to generate a functorial semantics for XTT; models form a 2-category with a bi-initial object, and morphisms satisfy compatibilities like $F(\Gamma.A) \cong F(\Gamma).F(A)$ which generalize pseudomorphisms of natural models [CD14, New18]. While it may seem at first that these canonical isomorphisms would incur additional bureaucracy, these weaker morphisms in fact enable us to work much more abstractly, choosing representations of objects only locally and as needed. Consequently, we have managed to avoid nearly all the concrete computations that characterized the technical development of our previous work on XTT [SAG19].

1.2.2. *Artin gluing.* Most metatheorems famously cannot be proven by a straightforward induction on the rules of type theory but require instead a more semantic induction principle, such as the method of computability pioneered by Tait for the simply typed λ-calculus [Tai67] and further developed by Girard [Gir71, Gir72], Martin-Löf [ML75b], and others. These methods associate to each type/context a proof-irrelevant predicate or relation over its elements, and then establish that every element satisfies the predicate associated to its type. To prove canonicity, one defines the elements of a type to be its closed terms, and the predicate over $\cdot \vdash b : \mathsf{bool}$ states that $b \Downarrow \mathsf{tt}$ or $b \Downarrow \mathsf{ff}$ where $\Downarrow$ is a deterministic evaluation relation contained in judgmental equality.

These techniques have a few major disadvantages in the context of dependent type theory. First, evaluation must be defined on typed terms modulo α-equivalence, not judgmental equality, because evaluation draws distinctions between β-equivalent terms (e.g., tt is an output of evaluation whereas $(\lambda x.x)(\mathsf{tt})$ is not); therefore, evaluation cannot be studied using the machinery of models of type theory.[3] Secondly, the predicate over $\cdot \vdash A : \mathcal{U}$ should intuitively state that A determines a type and thence a predicate over its elements, but a (proof-irrelevant) predicate for $\mathcal{U}$ cannot store the data of a predicate for each A; instead, we define a global lookup table for predicates [All87, Har92], and store in the predicate over $\cdot \vdash A : \mathcal{U}$ the assertion that A has an entry in the table. However, constructing these type

[3]This subequational aspect of evaluation is particularly thorny in cubical type theories because evaluation does not strictly respect dimension substitution, necessitating a technical condition known as closure under "coherent expansion" [Hub18, Ang19].

$$
\begin{array}{llll}
\textit{(contexts)} & \Gamma, \Delta & ::= & \cdot \mid \Gamma, i : \mathbb{I} \mid \Gamma, \phi \mid \Gamma, x : A \\
\textit{(dimensions)} & r, s & ::= & i \mid 0 \mid 1 \\
\textit{(face formulas)} & \phi, \psi & ::= & r = s \mid \phi \vee \psi \\
\textit{(types)} & A, B & ::= & \mathsf{el}(a) \mid (x : A) \to B \mid (x : A) \times B \mid \mathsf{path}_{i.A}(a, b) \mid \mathsf{bool} \mid \mathsf{set} \\
\textit{(terms)} & a, b & ::= & x \mid \lambda x.a \mid a\ b \mid \langle a, b \rangle \mid \mathsf{fst}(a) \mid \mathsf{snd}(a) \mid \lambda i.a \mid a\ r \mid \mathsf{tt} \mid \mathsf{ff} \mid \\
& & & \mathsf{if}_{x.A}(a; b, b') \mid [] \mid [\phi \to a \mid \psi \to b] \mid \mathsf{com}\langle s \rangle_{i.a}^{r \rightsquigarrow r'}\ i.b \mid \\
& & & \mathsf{coe}_{i.a}^{r \rightsquigarrow r'}\ b \mid (x : a) \mathbin{\widehat{\to}} b \mid (x : a) \mathbin{\widehat{\times}} b \mid \widehat{\mathsf{path}}_{i.a}(b, b') \mid \widehat{\mathsf{bool}} \mid \\
& & & \mathsf{case}_{x.A}\ a\ [\mathsf{pi}(x, x') \mapsto a' \mid \cdots \mid \mathsf{bool} \mapsto a'']
\end{array}
$$

Figure 1: A summary of the raw syntax of XTT.

systems requires fixing a collection of types at the outset, making these proofs brittle and difficult to extend.

Recently, type theorists have discovered that these difficulties can be overcome by considering instead *proof-relevant* predicates [AK16, Coq19, Shu15], and that the resulting constructions are best understood as instances of *Artin gluing* [AGV72, Exposé I, Ch. 9].[4]

Gluing-based techniques for type theory are perhaps most developed in the context of *weak* metatheorems such as homotopy canonicity [KS19, Shu15] and homotopy parametricity [Uem17], where it suffices to consider mathematically natural notions of model in which substitution does not strictly commute with the constructs of type theory [Joy17]. Canonicity and normalization are also susceptible to gluing arguments, but these arguments have generally relied on explicit constructions and computations rather than leveraging categorical techniques and results as in the weak case [Coq19, KHS19].

1.2.3. *Our contribution.* In this paper, we prove a canonicity theorem for XTT stating that any closed term of boolean type in the initial model is judgmentally equal to either tt or ff. Our canonicity proof builds on results of Sterling and Angiuli [SA20] concerning the gluing of models of type theory along a flat functor. We emphasize the conceptual nature of our canonicity proof, which avoids the explicit computations that pervaded both our prior work on XTT [SAG19] and much of the related work.

2. XTT: a cubical language for Bishop sets

We begin by introducing the XTT language (Figure 1) and sketching how we recover (and improve upon) ordinary type-theoretic equality reasoning for Bishop sets. For readability, we omit structural rules, congruence rules, and obvious premises to equational rules; readers can find a fully precise characterization of XTT as the bi-initial object in a 2-category of models described in Section 4. Our presentation differs slightly from the original formulation of XTT [SAG19]; we remark on these differences as they appear.

[4]Researchers have been aware of connections between computability predicates and gluing for much longer, but restricted to the *fiberwise proof-irrelevant* fragment of a gluing category [MS93, JT93, FS99].

2.1. **Judgmental structure of XTT.** Like other cubical type theories [CCHM17, ABC+19, AHH18, RS17], XTT extends the judgmental apparatus of type theory with an abstract interval $\mathbb{I}$ and a collection $\mathbb{F}$ of *face formulas*, or propositions ranging over the interval. Neither $\mathbb{I}$ nor $\mathbb{F}$ are types, but we can extend contexts by assumptions of either sort, in addition to ordinary typing assumptions. (Previously, we collected assumptions of $\mathbb{I}$ and $\mathbb{F}$ in a separate context Ψ to the left of Γ.)

$$\frac{}{\cdot\ ctx} \qquad \frac{\Gamma\ ctx}{\Gamma, i : \mathbb{I}\ ctx} \qquad \frac{\Gamma\ ctx \qquad \Gamma \vdash \phi : \mathbb{F}}{\Gamma, \phi\ ctx} \qquad \frac{\Gamma\ ctx \qquad \Gamma \vdash A\ type}{\Gamma, x : A\ ctx}$$

Assumptions of all three sorts are subject to the structural rules of hypothesis, substitution, and weakening. In addition to dimension variables $i : \mathbb{I}$, the interval has two global elements 0 and 1 representing its endpoints. (We call such an interval *Cartesian* because it is the free finite-product theory on two generators [Awo18a].) The face formulas $\mathbb{F}$ are closed under disjunction (unlike [SAG19]) and equality of dimensions.

$$\frac{}{\Gamma \vdash 0 : \mathbb{I}} \qquad \frac{}{\Gamma \vdash 1 : \mathbb{I}} \qquad \frac{\Gamma \vdash r : \mathbb{I} \qquad \Gamma \vdash s : \mathbb{I}}{\Gamma \vdash r = s : \mathbb{F}} \qquad \frac{\Gamma \vdash \phi : \mathbb{F} \qquad \Gamma \vdash \psi : \mathbb{F}}{\Gamma \vdash \phi \vee \psi : \mathbb{F}}$$

Given $\Gamma \vdash \phi : \mathbb{F}$, we write $\Gamma \vdash \phi\ true$ when ϕ holds under the assumptions in Γ. The rules governing this judgment are the evident ones, with two caveats. First, we consider two face formulas equal when they are interprovable. Secondly, under an assumption of $r = s$, one obtains a judgmental equality $r = s : \mathbb{I}$; we may safely adopt this principle because, unlike propositional equality in arbitrary types, $r = s$ is decidable.

$$\frac{\Gamma \vdash r = s : \mathbb{I}}{\Gamma \vdash r = s\ true} \qquad \frac{\Gamma \vdash r = s\ true}{\Gamma \vdash r = s : \mathbb{I}} \qquad \frac{\Gamma \vdash \phi\ true}{\Gamma \vdash \phi \vee \psi\ true} \qquad \frac{\Gamma \vdash \psi\ true}{\Gamma \vdash \phi \vee \psi\ true}$$

$$\frac{\Gamma \vdash \phi \vee \psi\ true \qquad \Gamma, \phi \vdash \chi\ true \qquad \Gamma, \psi \vdash \chi\ true}{\Gamma \vdash \chi\ true} \qquad \frac{\Gamma, \phi \vdash \psi\ true \qquad \Gamma, \psi \vdash \phi\ true}{\Gamma \vdash \phi = \psi : \mathbb{F}}$$

In XTT, maps out of $\mathbb{I}$ correspond to equality proofs: $i : \mathbb{I} \vdash A\ type$ is a proof that $[0/i]A$ and $[1/i]A$ are equal types, and $i : \mathbb{I} \vdash a : A$ is a proof that $[0/i]a : [0/i]A$ and $[1/i]a : [1/i]A$ are equal elements, modulo the proof A that $[0/i]A$ and $[1/i]A$ are equal types. Assumptions of face formulas act as *constraints*, restricting the domain of maps out of $\mathbb{I}^n$. A hypothesis of $i = 0$ sets i to 0 in the hypotheses and conclusions that follow, whereas a type/element under the false constraint $0 = 1$ is nothing at all. Finally, an element under a disjunction $\phi \vee \psi$ is a pair of elements under ϕ and ψ that agree on the overlap ϕ, ψ.

Unlike [SAG19], and following [CCHM17], we include syntax for these *partial elements* defined on nullary and binary disjunctions:

$$\frac{\Gamma \vdash 0 = 1\ true \qquad \Gamma \vdash A\ type}{\Gamma \vdash [\,] : A} \qquad \frac{\Gamma \vdash 0 = 1\ true \qquad \Gamma \vdash a : A}{\Gamma \vdash a = [\,] : A}$$

$$\frac{\Gamma \vdash \phi \vee \psi\ true \qquad \Gamma, \phi \vdash a_\phi : A \qquad \Gamma, \psi \vdash a_\psi : A \qquad \Gamma, \phi, \psi \vdash a_\phi = a_\psi : A}{\Gamma \vdash [\phi \to a_\phi \mid \psi \to a_\psi] : A}$$

$$\frac{\Gamma \vdash \phi\ \mathit{true}}{\Gamma \vdash [\phi \to a_\phi \mid \psi \to a_\psi] = a_\phi : A} \qquad \frac{\Gamma \vdash \psi\ \mathit{true}}{\Gamma \vdash [\phi \to a_\phi \mid \psi \to a_\psi] = a_\psi : A}$$

$$\frac{\Gamma \vdash \phi \vee \psi\ \mathit{true} \qquad \Gamma \vdash a : A}{\Gamma \vdash a = [\phi \to a \mid \psi \to a] : A}$$

Notation 2.1 (Boundary)**.** The interval has more generalized points than 0 and 1; therefore, it is not the case that $i : \mathbb{I} \vdash i = 0 \vee i = 1\ \mathit{true}$. This formula, called the *boundary* of i, is important for expressing the rules of path types and compositions; we therefore impose the following notation:

$$\partial(r) \stackrel{\mathit{def}}{=\!=} r = 0 \vee r = 1$$ ❦

Notation 2.2 (Judgmental restriction)**.** We often want to consider a total term whose subcube coincides with some other term. We will write $\Gamma \vdash a : A\ [\phi \to b]$ to abbreviate that a is a term that restricts on ϕ to b, that is:

$$\frac{\Gamma \vdash a : A \qquad \Gamma, \phi \vdash a = b : A}{\Gamma \vdash a : A\ [\phi \to b]}$$ ❦

Example 2.3. Combining Notations 2.1 and 2.2, we may succinctly express the situation where $p(i) : A$ exhibits a *path* (proof of equality) between two elements $a_0, a_1 : A$, writing $i : \mathbb{I} \vdash p(i) : A\ [\partial(i) \to a]$ where $a \stackrel{\mathit{def}}{=\!=} [i = 0 \to a_0 \mid i = 1 \to a_1]$. ❦

2.2. **Dependent path types in XTT.** The rules for dependent product, dependent sum, and boolean types in XTT are completely standard and are located in Figure 2. Cubical type theories internalize the judgmental "equality situation" of Example 2.3 by means of *(dependent) path types*; a path type is, in essence, a dependent function out of the interval subject to a restriction on the boundary of this function (i.e. the behavior of this function on the endpoints $0, 1 : \mathbb{I}$).

Given a line of types $i : \mathbb{I} \vdash A\ \mathit{type}$ and two elements $a_0 : [0/i]A, a_1 : [1/i]A$, the type of paths between a_0 and a_1 is written $\mathsf{path}_{i.A}(a_0, a_1)$; because the type A can depend on i, path types express a kind of heterogeneous equality (though different from the one proposed by McBride [McB99]). The rules for path types are summarized below:

FORMATION

$$\frac{\Gamma, i : \mathbb{I} \vdash A\ \mathit{type} \qquad \Gamma \vdash a_0 : [0/i]A \qquad \Gamma \vdash a_1 : [1/i]A}{\Gamma \vdash \mathsf{path}_{i.A}(a_0, a_1)\ \mathit{type}}$$

INTRODUCTION

$$\frac{\Gamma, i : \mathbb{I} \vdash a : A}{\Gamma \vdash \lambda i.a : \mathsf{path}_{i.A}([0/i]a, [1/i]a)}$$

ELIMINATION

$$\frac{\Gamma \vdash p : \mathsf{path}_{i.A}(a_0, a_1) \qquad \Gamma \vdash r : \mathbb{I}}{\Gamma \vdash p(r) : [r/i]A\ [\partial(r) \to [r = 0 \to a_0 \mid r = 1 \to a_1]]}$$

COMPUTATION

$$\frac{}{\Gamma \vdash (\lambda i.a)(r) = [r/i]a : [r/i]A}$$

UNIQUENESS

$$\frac{}{\Gamma \vdash p = \lambda i.p(i) : \mathsf{path}_{i.A}(a_0, a_1)}$$

Remark 2.4. One can also express the data of a path type as a line of types $i : \mathbb{I} \vdash A\ \mathit{type}$ together with a partial element $i : \mathbb{I}, \partial(i) \vdash a : A$; then, the elements of the path type $\mathsf{path}_{i.A}(i.a)$ would consist in elements $i : \mathbb{I} \vdash p : A\ [\partial(i) \to a]$. In fact, we use exactly this style of definition in our mathematical version of the syntax of XTT (see Section 4). ❦

$$\frac{\Gamma \vdash A\ \mathit{type} \qquad \Gamma, x : A \vdash B\ \mathit{type}}{\Gamma \vdash (x : A) \to B\ \mathit{type}} \qquad \frac{\Gamma, x : A \vdash b : B}{\Gamma \vdash \lambda x.b : (x : A) \to B}$$

$$\frac{\Gamma \vdash f : (x : A) \to B \qquad \Gamma \vdash a : A}{\Gamma \vdash f(a) : [a/x]B} \qquad \frac{}{\Gamma \vdash (\lambda x.b)(a) = [a/x]b : [a/x]B}$$

$$\frac{}{\Gamma \vdash f = \lambda x.f(x) : (x : A) \to B}$$

$$\frac{\Gamma \vdash A\ \mathit{type} \qquad \Gamma, x : A \vdash B\ \mathit{type}}{\Gamma \vdash (x : A) \times B\ \mathit{type}} \qquad \frac{\Gamma, x : A \vdash B\ \mathit{type} \qquad \Gamma \vdash a : A \qquad \Gamma \vdash b : [a/x]B}{\Gamma \vdash \langle a, b \rangle : (x : A) \times B}$$

$$\frac{\Gamma \vdash p : (x : A) \times B}{\Gamma \vdash \mathsf{fst}(p) : A} \qquad \frac{\Gamma \vdash p : (x : A) \times B}{\Gamma \vdash \mathsf{snd}(p) : [\mathsf{fst}(p)/x]B} \qquad \frac{}{\Gamma \vdash \mathsf{fst}(\langle a, b \rangle) = a : A}$$

$$\frac{}{\Gamma \vdash \mathsf{snd}(\langle a, b \rangle) = b : [a/x]B} \qquad \frac{}{\Gamma \vdash p = \langle \mathsf{fst}(p), \mathsf{snd}(p) \rangle : (x : A) \times B}$$

$$\frac{}{\Gamma \vdash \mathsf{bool}\ \mathit{type}} \qquad \frac{}{\Gamma \vdash \mathsf{tt} : \mathsf{bool}} \qquad \frac{}{\Gamma \vdash \mathsf{ff} : \mathsf{bool}}$$

$$\frac{\Gamma, x : \mathsf{bool} \vdash C\ \mathit{type} \qquad \Gamma \vdash b : \mathsf{bool} \qquad \Gamma \vdash c_{\mathsf{tt}} : [\mathsf{tt}/x]C \qquad \Gamma \vdash c_{\mathsf{ff}} : [\mathsf{ff}/x]C}{\Gamma \vdash \mathsf{if}_{x.C}(b; c_{\mathsf{tt}}, c_{\mathsf{ff}}) : [b/x]C}$$

$$\frac{}{\Gamma \vdash \mathsf{if}_{x.C}(\mathsf{tt}; c_{\mathsf{tt}}, c_{\mathsf{ff}}) = c_{\mathsf{tt}} : [\mathsf{tt}/x]C} \qquad \frac{}{\Gamma \vdash \mathsf{if}_{x.C}(\mathsf{ff}; c_{\mathsf{tt}}, c_{\mathsf{ff}}) = c_{\mathsf{ff}} : [\mathsf{ff}/x]C}$$

Figure 2: Rules for dependent products, dependent sums, and booleans. We omit a number of obvious premises that must be included for a fully precise presentation.

We are already prepared to see one of the advantages of cubical type theories over intensional Martin-Löf type theory. The benefit of using maps out of the interval to represent equality is that equations in A naturally take the form of (parametrized) elements of A; therefore, the "introduction rules" for equality in A are the same as the introduction rules for A itself.

Example 2.5. Function extensionality, provable in cubical type theories, provides a particularly convincing example, considering that it can be derived directly using only the rules for dependent function and path types. Given two functions $f, g : (x : A) \to B$ and a family of paths $h : (x : A) \to \mathsf{path}_{\_.B}(f(x), g(x))$, we have:

$$\lambda i.\lambda x.h(x)(i) : \mathsf{path}_{\_.(x:A)\to B}(f, g)$$ ❦

2.3. **Universe of Bishop sets.** XTT is equipped with a universe of *Bishop sets*, i.e. types that satisfy a definitional version of the unicity of identity proofs. As was the case for Observational Type Theory, it is essential that that this universe is *closed* — a matter we will discuss in more detail in Sections 2.3.4 and 8.1.

In our original presentation of XTT [SAG19], we required that all types were Bishop sets. Here, we require this property only of elements of the universe, in order to suggest how one might integrate XTT into a standard (univalent) Cartesian cubical type theory

in which not all types are Bishop sets (notably, univalent universes and higher inductive types). Additionally, whereas we previously described an infinite and cumulative hierarchy of universes à la Coquand,[5] here we have opted to specify only a single universe à la Tarski for the sake of simplicity and clarity of presentation.

We begin with the basic formation rules for the universe of Bishop sets:

$$\frac{}{\Gamma \vdash \mathsf{set}\ \mathit{type}} \qquad \frac{\Gamma \vdash \hat{A} : \mathsf{set}}{\Gamma \vdash \mathsf{el}(\hat{A})\ \mathit{type}}$$

Notation 2.6. As above, we adopt the convention of writing $\hat{A}$ for an element of set; then, we will write $a \in \hat{A}$ as a shorthand for $a : \mathsf{el}(\hat{A})$. ❦

2.3.1. *Boundary separation and UIP.* What makes types classified by set special is that they satisfy the *boundary separation* principle below, a modular reconstruction of the uniqueness of identity proofs:

$$\textsc{boundary separation}\quad \frac{\Gamma \vdash \hat{A} : \mathsf{set} \qquad \Gamma \vdash r : \mathbb{I} \qquad \Gamma, \partial(r) \vdash a = b \in \hat{A}}{\Gamma \vdash a = b \in \hat{A}}$$

To see that boundary separation implies the unicity of identity proofs, we consider context $\Gamma \stackrel{\mathit{def}}{=\!=} (\Delta, \hat{A} : \mathsf{set}, a \in \hat{A}, b \in \hat{A}, p : \mathsf{path}_{\_.\mathsf{el}(\hat{A})}(a,b), q : \mathsf{path}_{\_.\mathsf{el}(\hat{A})}(a,b))$; we may derive $\Gamma \vdash p = q : \mathsf{path}_{\_.\mathsf{el}(\hat{A})}(a,b)$ as follows:

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\Gamma, i : \mathbb{I}, \partial(i) \vdash [i = 0 \to a \mid i = 1 \to b] = [i = 0 \to a \mid i = 1 \to b] \in \hat{A}}}{\Gamma, i : \mathbb{I}, \partial(i) \vdash [i = 0 \to p(i) \mid i = 1 \to p(i)] = [i = 0 \to q(i) \mid i = 1 \to q(i)] \in \hat{A}} \qquad \dfrac{\Gamma, i : \mathbb{I}, \partial(i) \vdash p(i) = q(i) \in \hat{A}}{\Gamma, i : \mathbb{I} \vdash p(i) = q(i) \in \hat{A}}\ \textsc{boundary separation}}{\Gamma \vdash \lambda i.p(i) = \lambda i.q(i) : \mathsf{path}_{\_.\mathsf{el}(\hat{A})}(a,b)}}{\Gamma \vdash p = q : \mathsf{path}_{\_.\mathsf{el}(\hat{A})}(a,b)}$$

2.3.2. *Coercion and composition.* Another important aspect of Bishop sets in XTT is that they support *coercion* and *composition* operations:

$$\frac{\Gamma, i : \mathbb{I} \vdash \hat{A} : \mathsf{set} \qquad \Gamma \vdash a \in [r/i]\hat{A}}{\Gamma \vdash \mathsf{coe}^{r \rightsquigarrow r'}_{i.\hat{A}}\, a \in [r'/i]\hat{A}\ [r' = r \to a]} \qquad \frac{\Gamma, i : \mathbb{I} \vdash \hat{A} : \mathsf{set} \qquad \Gamma, i : \mathbb{I}, i = r \vee \partial(s) \vdash a \in \hat{A}}{\Gamma \vdash \mathsf{com}\langle s \rangle^{r \rightsquigarrow r'}_{i.\hat{A}}\, i.a \in [r'/i]\hat{A}\ [r' = r \vee \partial(s) \to a]}$$

In essence, these operations implement the action of paths in every set, simultaneously enabling coercions between equal types, as well providing a way to compose and invert paths. The coercion operation above allows, in particular, an element of a set to be transformed into an element of any equal set: this is the action of $\mathsf{coe}^{0 \rightsquigarrow 1}_{i.\hat{A}}\, a$. In Observational Type Theory, there is an additional *coherence* operation that (heterogeneously) equates a with its coercion

[5] Universes à la Coquand [Coq13] differ from universes à la Tarski in a few ways: one eschews the standard $\Gamma \vdash A\ \mathit{type}$ judgment for a stratified judgment $\Gamma \vdash A\ \mathit{type}_i$, and then the rules for each universe $\mathcal{U}_i$ exhibit an isomorphism between the collection of types of level i and the collection of elements of $\mathcal{U}_i$.

$\mathsf{coe}^{0 \rightsquigarrow 1}_{i.\hat{A}}\, a$; in XTT (and Cartesian cubical type theories generally), this is accomplished using another instance of the general coercion operator called a "filler":

$$(\lambda i.\mathsf{coe}^{0 \rightsquigarrow i}_{i.\hat{A}}\, a) : \mathsf{path}_{i.\mathsf{el}(\hat{A})}(a, \mathsf{coe}^{0 \rightsquigarrow 1}_{i.\hat{A}}\, a)$$

Composition is analogous to coercion, except that it may additionally constrain the result to match a partial element defined on a boundary $\partial(s)$ for some $s : \mathbb{I}$. (Because of boundary separation and regularity, XTT's composition operator is substantially simpler than those of other cubical type theories, which consider partial elements defined on arbitrary $\phi : \mathbb{F}$.) Composition can be used to define combinators expressing the symmetry and transitivity of equality, as well as to implement Martin-Löf's J eliminator. In fact, to express symmetry and transitivity, it suffices to first consider the case where $\hat{A}$ doesn't depend on i, called *homogeneous composition*:

$$\mathsf{hcom}\langle s \rangle^{r \rightsquigarrow r'}_{\hat{A}}\, i.a \stackrel{def}{=\!=} \mathsf{com}\langle s \rangle^{r \rightsquigarrow r'}_{\_.\hat{A}}\, i.a$$

Example 2.7 (Symmetry)**.** Let $\hat{A} : \mathsf{set}$ and let $a, b \in \hat{A}$ and let $p : \mathsf{path}_{\_.\mathsf{el}(\hat{A})}(a, b)$. We may use homogeneous composition to define an inverse path $\bar{p} : \mathsf{path}_{\_.\mathsf{el}(\hat{A})}(b, a)$:

$$\bar{p} \stackrel{def}{=\!=} \lambda i.\mathsf{hcom}\langle i \rangle^{0 \rightsquigarrow 1}_{\hat{A}}\, j.[j = 0 \vee i = 1 \to p(0) \mid i = 0 \to p(j)]$$ ❦

Example 2.8 (Transitivity)**.** Let $\hat{A} : \mathsf{set}$ and let $a, b, c \in \hat{A}$ and let $p : \mathsf{path}_{\_.\mathsf{el}(\hat{A})}(a, b), q : \mathsf{path}_{\_.\mathsf{el}(\hat{A})}(b, c)$. We may use homogeneous composition to define a composite path $p \cdot q : \mathsf{path}_{\_.\mathsf{el}(\hat{A})}(a, c)$:

$$p \cdot q \stackrel{def}{=\!=} \lambda i.\mathsf{hcom}\langle i \rangle^{0 \rightsquigarrow 1}_{\hat{A}}\, j.[j = 0 \vee i = 0 \to p(i) \mid i = 1 \to q(j)]$$ ❦

Remark 2.9. By boundary separation, the symmetry and transitivity operators act very strictly. For instance, one has $\bar{p} \cdot p = p$ and $p \cdot (q \cdot w) = (p \cdot q) \cdot w$ definitionally. In the absence of boundary separation, these coherences would hold up to another path, using a more complex instance of the composition operation. ❦

Example 2.10 (Identity type)**.** Using composition, we may define a combinator with the same type as Martin-Löf's J eliminator for the identity type. Let $\hat{A} : \mathsf{set}$ be a set and $x \in \hat{A}, y \in \hat{A}, z : \mathsf{path}_{\_.\mathsf{el}(\hat{A})}(x, y) \vdash \hat{C}(x, y, z) : \mathsf{set}$ be a motive of induction. Fixing $a, b \in \hat{A}$ and $p : \mathsf{path}_{\_.\mathsf{el}(\hat{A})}(a, b)$ and $x : A \vdash c(x) \in \hat{C}(x, x, \lambda\_.x)$, we may define an element $\mathsf{J}_{\hat{C}}(p, c) \in \hat{C}(a, b, p)$ as follows:

$$\mathsf{J}_{\hat{C}}(p, c) \stackrel{def}{=\!=} \mathsf{coe}^{0 \rightsquigarrow 1}_{i.\hat{C}(p(0), p(i), \lambda j.\mathsf{hcom}\langle i \rangle^{0 \rightsquigarrow j}_{\hat{A}}\, k.[k=0 \vee i=0 \to p(0) \mid i=1 \to p(k)])}\, c(p(0))$$ ❦

What is the behavior of the J combinator from Example 2.10 on a *reflexive* proof of equality $\lambda\_.a : \mathsf{path}_{\_.\mathsf{el}(\hat{A})}(a, a)$? From Martin-Löf type theory, we would expect $\mathsf{J}_{\hat{C}}(\lambda\_.a, c)$ to compute to $c(a)$; in ordinary cubical type theory, this equation only holds up to another path, but in XTT we can force it to hold using the following *regularity* principle:

$$\textsc{coercion regularity}\quad \frac{\Gamma, i : \mathbb{I}, j : \mathbb{I} \vdash \hat{A} = [j/i]\hat{A} : \mathsf{set}}{\Gamma \vdash \mathsf{coe}^{r \rightsquigarrow r'}_{i.\hat{A}}\, a = a \in [r'/i]\hat{A}}$$

Remark 2.11. Considering the boundary $\partial(s)$, the boundary separation rule ensures that the standard decomposition of composition into homogeneous composition and coercion [ABC+19, AHH17] holds definitionally:

$$\mathsf{com}\langle s\rangle^{r\rightsquigarrow r'}_{i.\hat{A}}\ i.a = \mathsf{hcom}\langle s\rangle^{r\rightsquigarrow r'}_{[r'/i]\hat{A}}\ i.\mathsf{coe}^{i\rightsquigarrow r'}_{i.\hat{A}}\ a$$

Consequently, the following regularity rule for composition is also derivable:

$$\frac{\Gamma, i : \mathbb{I}, j : \mathbb{I} \vdash \hat{A} = [j/i]\hat{A} : \mathsf{set} \qquad \Gamma, i : \mathbb{I}, i = r \vee \partial(s), j : \mathbb{I}, j = r \vee \partial(s) \vdash a = [j/i]a \in \hat{A}}{\Gamma \vdash \mathsf{com}\langle s\rangle^{r\rightsquigarrow r'}_{i.\hat{A}}\ i.a = [r'/i]a \in [r'/i]\hat{A}}$$

❦

2.3.3. *Closure of the universe under connectives.* The universe is closed under codes for connectives in the standard way, by adding introduction forms for each code and equations governing the behavior of $\mathsf{el}(-)$ on codes:

$$\frac{\Gamma \vdash \hat{A} : \mathsf{set} \qquad \Gamma, x \in \hat{A} \vdash \hat{B} : \mathsf{set}}{\Gamma \vdash (x : \hat{A}) \hat{\to} \hat{B} : \mathsf{set}} \qquad \frac{\Gamma \vdash \hat{A} : \mathsf{set} \qquad \Gamma, x \in \hat{A} \vdash \hat{B} : \mathsf{set}}{\Gamma \vdash (x : \hat{A}) \hat{\times} \hat{B} : \mathsf{set}}$$

$$\frac{\Gamma, i : \mathbb{I} \vdash \hat{A} : \mathsf{set} \qquad \Gamma \vdash a \in [0/i]\hat{A} \qquad \Gamma \vdash b \in [1/i]\hat{A}}{\Gamma \vdash \widehat{\mathsf{path}}_{i.\hat{A}}(a, b) : \mathsf{set}} \qquad \frac{}{\Gamma \vdash \widehat{\mathsf{bool}} : \mathsf{set}}$$

$$\begin{aligned}
&\Gamma \vdash \mathsf{el}((x : \hat{A}) \hat{\to} \hat{B}) = (x : \mathsf{el}(\hat{A})) \to \mathsf{el}(\hat{B})\ \textit{type}\\
&\Gamma \vdash \mathsf{el}((x : \hat{A}) \hat{\times} \hat{B}) = (x : \mathsf{el}(\hat{A})) \times \mathsf{el}(\hat{B})\ \textit{type}\\
&\Gamma \vdash \mathsf{el}(\widehat{\mathsf{path}}_{i.\hat{A}}(a, b)) = \mathsf{path}_{i.\mathsf{el}(\hat{A})}(a, b)\ \textit{type}\\
&\Gamma \vdash \mathsf{el}(\widehat{\mathsf{bool}}) = \mathsf{bool}\ \textit{type}
\end{aligned}$$

Considering that BOUNDARY SEPARATION applies to all elements of set, we are restricted to connectives that preserve the condition of being boundary separated. Next, we must include equations specifying the behavior of coe and com on each type code. We begin with coercion, verifying in each case that the equation is compatible with COERCION REGULARITY.

$$\begin{aligned}
\mathsf{coe}^{r\rightsquigarrow r'}_{i.(x:\hat{A})\hat{\to}\hat{B}}\ f &= \lambda x.\mathsf{coe}^{r\rightsquigarrow r'}_{i.[\mathsf{coe}^{r'\rightsquigarrow i}_{i.\hat{A}}\ x/x]\hat{B}}\ f(\mathsf{coe}^{r'\rightsquigarrow r}_{i.\hat{A}}\ x)\\
\mathsf{coe}^{r\rightsquigarrow r'}_{i.(x:\hat{A})\hat{\times}\hat{B}}\ p &= \langle \mathsf{coe}^{r\rightsquigarrow r'}_{i.\hat{A}}\ \mathsf{fst}(p), \mathsf{coe}^{r\rightsquigarrow r'}_{i.[\mathsf{coe}^{r\rightsquigarrow i}_{i.\hat{A}}\ \mathsf{fst}(p)/x]\hat{B}}\ \mathsf{snd}(p)\rangle\\
\mathsf{coe}^{r\rightsquigarrow r'}_{i.\widehat{\mathsf{path}}_{j.\hat{A}}(a_0,a_1)}\ p &= \lambda j.\mathsf{com}\langle j\rangle^{r\rightsquigarrow r'}_{i.\hat{A}}\ _.p(j)
\end{aligned}$$

Of course, COERCION REGULARITY implies $\mathsf{coe}^{r\rightsquigarrow r'}_{i.\widehat{\mathsf{bool}}}\ a = a$. We additionally observe that the behavior of homogeneous composition (and thence general composition) is totally determined by the combination of the above and boundary separation; in particular, the following equations are derivable by pivoting on $\partial(s)$:

$$\begin{aligned}
\mathsf{hcom}\langle s\rangle^{r\rightsquigarrow r'}_{(x:\hat{A})\hat{\to}\hat{B}}\ i.f &= \lambda x.\mathsf{hcom}\langle s\rangle^{r\rightsquigarrow r'}_{\hat{B}}\ i.f(x)\\
\mathsf{hcom}\langle s\rangle^{r\rightsquigarrow r'}_{(x:\hat{A})\hat{\times}\hat{B}}\ i.p &= \langle \mathsf{hcom}\langle s\rangle^{r\rightsquigarrow r'}_{\hat{A}}\ i.\mathsf{fst}(p), \mathsf{com}\langle s\rangle^{r\rightsquigarrow r'}_{i.[\mathsf{hcom}\langle s\rangle^{r\rightsquigarrow i}_{\hat{A}}\ i.\mathsf{fst}(p)/x]\hat{B}}\ i.\mathsf{snd}(p)\rangle
\end{aligned}$$

$$\mathsf{hcom}\langle s\rangle^{r\rightsquigarrow r'}_{\widehat{\mathsf{path}}_{j.\hat{A}}(a,b)}\ i.p = \lambda j.\mathsf{hcom}\langle s\rangle^{r\rightsquigarrow r'}_{\hat{A}}\ i.p(j)$$

2.3.4. *Algorithmic type checking and typecase.* In this paper, we do not present an algorithm for type checking XTT; such an algorithm would be important to fully substantiate our claim that XTT can act as a more tractable alternative to extensional type theory. However, type checking in the presence of the boundary separation rule requires us to add further constructs to XTT, as outlined below.

Most type checking algorithms going back to the work of Coquand [Coq96] check $M : A$ by first evaluating A to a weak-head normal form, in order to determine whether A is a dependent product type, a dependent sum type, the booleans, etc. Such a determination is crucial because the head constructor of A, in turn, determines how to type check M (e.g., by applying it to an argument, considering its projections, etc.).

Consider the case that we have a variable $x : \mathsf{path}_{\_.\mathsf{set}}(\hat{A} \mathbin{\hat{\times}} \hat{B}, \hat{C} \mathbin{\hat{\times}} \hat{D})$ in scope, and we are attempting to check $\langle u, v\rangle \in x(i)$ for some variable $i : \mathbb{I}$. The equational rules of XTT do not suggest any reductions for $x(i)$, so we might naïvely return a type error: $\langle u, v\rangle$ can only be an element of a product type, but $\mathsf{el}(x(i))$ appears to be neutral.

Such a strategy is, however, not complete. Suppose that in addition, we have proofs $p : \mathsf{path}_{\_.\mathsf{set}}(\hat{A}, \hat{C})$ and $q : \mathsf{path}_{\_.\mathsf{set}}(\hat{B}, \hat{D})$ in scope. Then we may form the path $\lambda j.p(j) \mathbin{\hat{\times}} q(j) : \mathsf{path}_{\_.\mathsf{set}}(\hat{A} \mathbin{\hat{\times}} \hat{B}, \hat{C} \mathbin{\hat{\times}} \hat{D})$; by boundary separation, $\mathsf{el}(x(i)) = \mathsf{el}(p(i)) \times \mathsf{el}(q(i))$ definitionally, and therefore we *must* proceed to check $u \in p(i)$ and $v \in q(i)$ (and possibly succeed).

Of course, an algorithm cannot guess out of thin air whether such p, q exist! A way around this impasse, pioneered in OTT [AM06, AMS07], is to ensure that from a path between $\hat{A} \mathbin{\hat{\times}} \hat{B}$ and $\hat{C} \mathbin{\hat{\times}} \hat{D}$, we can *always* obtain a path between $\hat{A}$ and $\hat{C}$, etc. Under those circumstances, we have a uniform strategy to type check terms on neutral equations between product types (etc.): ignore the proof of equality, and consider only its boundary.

This approach does *not* make sense for mathematical sets or spaces, since there are more ways for two product sets to be equal than that their components are equal, but it does make sense for closed universes, such as the inductive-recursive universes of Martin-Löf [ML84]. We discuss the semantic disadvantages of these closed universes in Section 8.1.

A simple way to support this "injectivity up to paths" of type constructors in XTT is to add a "typecase" operator enabling intensional analysis of sets [Con82, CZ84, HM95].

$$\frac{\begin{array}{l}\Gamma, u : \mathsf{set} \vdash C\ \mathit{type}\\ \Gamma, u : \mathsf{set}, v : \mathsf{el}(u) \to \mathsf{set} \vdash c_\Pi : [(x : u) \mathbin{\hat{\to}} v(x)/u]C\\ \Gamma, u : \mathsf{set}, v : \mathsf{el}(u) \to \mathsf{set} \vdash c_\Sigma : [(x : u) \mathbin{\hat{\times}} v(x)/u]C\\ \Gamma, u_0 : \mathsf{set}, u_1 : \mathsf{set}, u_p : \mathsf{path}_{\_.\mathsf{set}}(u_0, u_1), x_0 \in u_0, x_1 \in u_1 \vdash c_p : [\widehat{\mathsf{path}}_{i.u_p(i)}(x_0, x_1)/u]C\\ \Gamma \vdash c_b : [\widehat{\mathsf{bool}}/u]C\\ \Gamma \vdash \hat{A} : \mathsf{set}\end{array}}{\Gamma \vdash \mathsf{case}_{u.C}\ \hat{A}\ \begin{bmatrix}\mathsf{pi}(u, v) \to c_\Pi\\ \mathsf{sg}(u, v) \to c_\Sigma\\ \mathsf{path}(u_0, u_1, u_p, x_0, x_1) \to c_p\\ \mathsf{bool} \to c_b\end{bmatrix} : [\hat{A}/u]C}$$

Then, the obvious reduction rules are added:

$$\mathsf{case}_{u.C}\ ((z : \hat{A}) \mathbin{\hat{\to}} \hat{B})\ [\cdots] = [\hat{A}, \lambda z.\hat{B}/u, v]c_\Pi$$

$$\mathsf{case}_{u.C}\ ((z : \hat{A}) \mathbin{\hat{\times}} \hat{B})\ [\cdots] = [A, \lambda z.B/u, v]c_\Sigma$$
$$\mathsf{case}_{u.C}\ (\widehat{\mathsf{path}}_{i.\hat{A}}(a,b))\ [\cdots] = [[0/i]\hat{A}, [1/i]\hat{A}, \lambda i.\hat{A}, a, b/u_0, u_1, u_p, x_0, x_1]c_p$$
$$\mathsf{case}_{u.C}\ \widehat{\mathsf{bool}}\ [\cdots] = c_b$$

3. Categorical preliminaries

All the categorical machinery we assume can be found in standard introductory textbooks and references [ML98, Bor94a, Bor94b, Bor10, Awo10, Joh02]; in order to fix notations and render our presentation as self-contained as possible, however, we have included a number of definitions.

3.1. **Basic categorical definitions.**

Notation 3.1. Given a category $\mathcal{C}$ and objects $C, D : \mathcal{C}$, we write $\mathcal{C}[C, D]$ for the collection of arrows between C and D. We will also write $[\mathcal{C}, \mathcal{D}]$ for the *category* of functors $\mathcal{C} \longrightarrow \mathcal{D}$ and natural transformations between them. ❦

Convention 3.2. Conventionally, we write **Set** and **Cat** for the categories of sets and categories respectively; of course, to be more precise, we should instead refer to $\mathbf{Set}_\alpha$ and $\mathbf{Cat}_\alpha$ for some strongly inaccessible cardinal α, or equivalently a Grothendieck universe $\mathcal{U}$. We leave the resolution of these universes implicit, noting them explicitly in sensitive places.

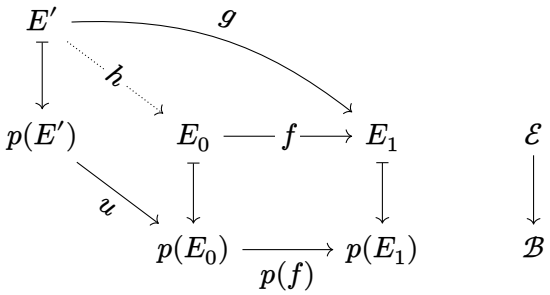
Notation 3.3. We will write Δ^n for the n-simplex regarded as a category; in particular, Δ^0 is the terminal category $\{*\}$, and Δ^1 is the category $\{\bullet \longrightarrow \circ\}$ of the walking arrow. Therefore $[\Delta^1, \mathcal{C}]$ is the category of arrows and commutative squares in $\mathcal{C}$. ❦

Definition 3.4 (Cartesian arrow category)**.** We write $[\Delta^1, \mathcal{C}]_{cart} \hookrightarrow [\Delta^1, \mathcal{C}]$ for the wide subcategory of arrows and *cartesian* squares between them. Concretely, given $f, g : [\Delta^1, \mathcal{C}]_{cart}$ a morphism between them exhibits f as a pullback of g:

$$\begin{array}{ccc} \partial_0 f & \longrightarrow & \partial_0 f \\ {\scriptstyle f}\downarrow & \lrcorner & \downarrow{\scriptstyle g} \\ \partial_1 f & \longrightarrow & \partial_1 g \end{array}$$

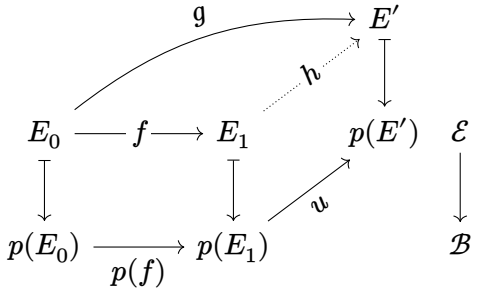
❦

Definition 3.5 (Cartesian morphism)**.** Given a functor $\mathcal{E} \xrightarrow{p} \mathcal{B}$, a morphism $E_0 \xrightarrow{f} E_1$ is *cartesian* if for every $E' \xrightarrow{g} E_1$ and $p(E') \xrightarrow{u} p(E_0)$ such that $p(g) = p(f) \circ u$, there exists a unique $E' \xrightarrow{h} E_0$ over u such that $f \circ h = g$. Diagrammatically:



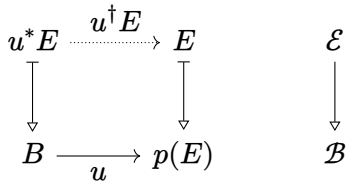
❦

Definition 3.6 (Opcartesian morphism)**.** Dually, given a functor $\mathcal{E} \xrightarrow{p} \mathcal{B}$, a morphism $E_0 \xrightarrow{f} E_1$ is *opcartesian* if for every $E_0 \xrightarrow{g} E'$ and $p(E_1) \xrightarrow{u} p(E')$ such that $p(g) = u \circ p(f)$, there exists a unique factor $E_1 \xrightarrow{h} E'$ lying over u such that $h \circ f = g$:



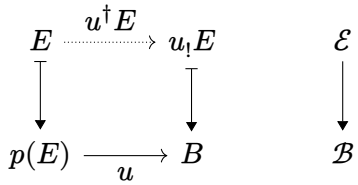
❦

Definition 3.7 (Fibration)**.** A *fibration* is a functor $\mathcal{E} \xrightarrow{p} \mathcal{B}$ such that for each morphism $B \xrightarrow{u} p(E)$ there exists a cartesian morphism $u^*E \xrightarrow{u^\dagger E} E$ lying over u:



We emphasize the property of an arrow being a fibration by using an open triangular tip, e.g. $\mathcal{E} \xrightarrow{p} \mathcal{B}$. ❦

Definition 3.8 (Opfibration)**.** Dually, an *opfibration* is a functor $\mathcal{E} \xrightarrow{p} \mathcal{B}$ such that for each morphism $p(E) \xrightarrow{u} B$ there exists an opcartesian morphism $E \xrightarrow{u^\dagger E} u_! E$ lying over u:



We emphasize the property of an arrow being an opfibration by using a filled triangular tip, e.g. $\mathcal{E} \xrightarrow{p} \mathcal{B}$. ❦

Fact 3.9. The codomain functor $[\Delta^1, \mathcal{C}] \xrightarrow{\partial_1} \mathcal{C}$ which sends $X \xrightarrow{f} Y$ to Y is always an opfibration, with opcartesian lifts implemented by postcomposition (dependent sum). ❦

Exercise 3.10. In a category $\mathcal{C}$ with pullbacks, the codomain functor $[\Delta^1, \mathcal{C}] \xrightarrow{\partial_1} \mathcal{C}$ is a fibration. Show that the cartesian lifts may be implemented by pullbacks in $\mathcal{C}$. ❦

Definition 3.11 (Comma category)**.** Given a pair of functors $\mathcal{D} \xrightarrow{F} \mathcal{C}$, $\mathcal{E} \xrightarrow{G} \mathcal{C}$, the *comma category* $F \downarrow G$ has as objects arrows $F(D) \xrightarrow{X} G(E)$ and commutative squares of the

following kind for arrows:

$$\begin{array}{ccc} F(D) & \xrightarrow{F(d)} & F(D') \\ {\scriptstyle X}\downarrow & & \downarrow{\scriptstyle X'} \\ G(E) & \xrightarrow[G(e)]{} & G(E') \end{array}$$

The comma category $F \downarrow G$ may be constructed more abstractly in terms of the following (1-categorical) pullback in **Cat**, the category of categories and functors:

$$\begin{array}{ccc} F \downarrow G & \longrightarrow & [\Delta^1, \mathcal{C}] \\ \downarrow & \lrcorner & \downarrow{\scriptstyle (\partial_0, \partial_1)} \\ \mathcal{D} \times \mathcal{E} & \xrightarrow[(F,G)]{} & \mathcal{C} \times \mathcal{C} \end{array} \tag{3.12}$$

Notation 3.13. Let $X : \mathcal{C}$; we will write $\Delta^0 \xrightarrow{\{X\}} \mathcal{C}$ for the constant functor $* \mapsto X$.

Notation 3.14. A common abuse of notation in the comma construction is that, when either F or G is the identity functor $\mathcal{C} \xrightarrow{\mathbf{id}_\mathcal{C}} \mathcal{C}$, they shall be written simply $\mathcal{C}$. For instance, $\mathcal{C} \downarrow G$ is written for $\mathbf{id}_\mathcal{C} \downarrow G$.

An important instance of the comma construction is the *slice* category.

Definition 3.15 (Slice category)**.** Given an object $X : \mathcal{C}$, the *slice* or "over-category" of $\mathcal{C}$ at X is the comma category $\mathcal{C}_{/X} = \mathcal{C} \downarrow \{X\}$. The objects of $\mathcal{C}_{/X}$ can be seen to be arrows $Y \longrightarrow X$; morphisms in the slice are commutative triangles.

Fact 3.16. In a category with pullbacks $\mathcal{C}$, a morphism $X \xrightarrow{f} Y$ induces a functor $\mathcal{C}_{/Y} \xrightarrow{f^*} \mathcal{C}_{/X}$ sending $Z \xrightarrow{g} Y$ to $Z \times_Y X \xrightarrow{f^* g} X$.

Notation 3.17. We will adopt a common abuse of notation and suppress the *weakening* functor $\mathcal{C} \xrightarrow{!_X^*} \mathcal{C}_{/X}$ when it is unambiguous.

Definition 3.18. Let κ be a regular cardinal; we say that a category $\mathcal{C}$ is κ-(co)complete when $\mathcal{C}$ has all (co)limits of κ-small diagrams; a functor that preserves these (co)limits is called κ-(co)continuous. When κ is omitted, a sufficiently large strongly inaccessible cardinal is assumed.

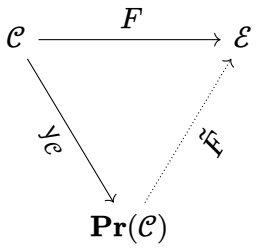
Definition 3.19. A finitely complete category, i.e. a category having all finite limits, is often called *left exact* or *lex*; likewise, a finitely continuous functor between lex categories is also called lex.

3.1.1. *Presheaves, representability, and discrete fibrations.*

Definition 3.20. A *presheaf* on $\mathcal{C}$ is a functor $\mathcal{C}^{\mathrm{op}} \xrightarrow{F} \mathbf{Set}$; the category of presheaves $[\mathcal{C}^{\mathrm{op}}, \mathbf{Set}]$ is written $\mathbf{Pr}(\mathcal{C})$. ❦

Presheaves capture the geometric intuition of probing a space or other object by small figures: the role of contexts and substitutions in (strict) type theory supplies type theorists and logicians with a useful concrete intuition for presheaves. A more structural perspective on presheaves is, however, essential: the category $\mathbf{Pr}(\mathcal{C})$ may be characterized universally as the *free cocompletion* of $\mathcal{C}$, equipping $\mathcal{C}$ with new colimits. When $\mathcal{C}$ already has some colimits, it is important to note that the new ones do *not* coincide with the old ones.

Construction 3.21 (The Yoneda embedding)**.** To be more precise, there is a universal functor $\mathcal{C} \xrightarrow{\mathsf{y}_{\mathcal{C}}} \mathbf{Pr}(\mathcal{C})$, called the *Yoneda embedding*, taking each object $C : \mathcal{C}$ to a "formal colimit" $\mathsf{y}_{\mathcal{C}}(C) = \mathcal{C}[\bullet, C]$, such that every cocontinuous functor $\mathcal{C} \longrightarrow \mathcal{E}$ with $\mathcal{E}$ cocomplete factors through $\mathsf{y}_{\mathcal{C}}$ in an essentially unique way:



❦

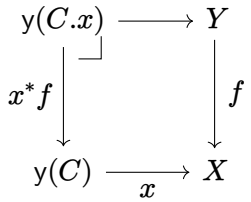
Lemma 3.22 (Yoneda lemma)**.** *As its name suggests, the Yoneda embedding is both full and faithful; this follows from an even stronger fact, called the* Yoneda lemma*; for each presheaf $X : \mathbf{Pr}(\mathcal{C})$, we have the following isomorphism:*

$$\mathbf{Pr}(\mathcal{C})[\mathsf{y}(C), X] \cong X(C)$$

Definition 3.23. A presheaf $X : \mathbf{Pr}(\mathcal{C})$ is called *representable* when it lies in the essential image of $\mathsf{y}_{\mathcal{C}}$, i.e. X is isomorphic to $\mathsf{y}_{\mathcal{C}}(C)$ for some $C : \mathcal{C}$. ❦

The notion of representable object is extended to maps in a canonical way, by considering fibers over representable objects.

Definition 3.24 (Representable natural transformation)**.** A representable natural transformation is a map $Y \xrightarrow{f} X : \mathbf{Pr}(\mathcal{C})$ whose every fiber over a representable object is representable. In other words, the fiber product of f with any $\mathsf{y}(C) \xrightarrow{x} X$ is representable:



❦

Representable natural transformations are a prime example of Grothendieck's "relative point of view", extending a notion that is first defined on objects to have sense on *morphisms*. It is useful to remark that the slice $\mathbf{Pr}(\mathcal{C})_{/X}$ is itself the category of presheaves $\mathbf{Pr}(\mathcal{C}/X)$ on the *category of elements* $\mathcal{C}/X$ of X, and that the representability of the map $Y \xrightarrow{f} X$ agrees with the representability of the object $f : \mathbf{Pr}(\mathcal{C})_{/X}$.

Construction 3.25 (Category of elements)**.** The *category of elements* $\mathcal{C}/X$ of a presheaf $X : \mathbf{Pr}(\mathcal{C})$ has as objects pairs C/x with $x \in X(C)$ and morphisms $D/f^*x \xrightarrow{f\dagger x} C/x$ for each $D \xrightarrow{f} C$ and $x \in X(C)$. ❦

In fact, the category of elements of a presheaf $X : \mathbf{Pr}(\mathcal{C})$ is the total category of a *discrete fibration* over $\mathcal{C}$.

Definition 3.26. A fibration $p : \mathcal{E} \to \mathcal{B}$ is discrete if $p(f) = \mathbf{id}_{\mathcal{B}}$ implies that $f = \mathbf{id}_{\mathcal{E}}$; equivalently, if the fibers of p are discrete categories. The collection of discrete fibrations over $\mathcal{C}$ forms a full subcategory $\mathbf{DF}_{\mathcal{C}} \subseteq \mathbf{Fib}_{\mathcal{C}} \subseteq \mathbf{Cat}_{/\mathcal{C}}$, with morphisms given by commuting triangles. ❦

Lemma 3.27. *Let $X : \mathbf{Pr}(\mathcal{C})$ be a presheaf; the functor $\mathcal{C}/X \xrightarrow{p_X} \mathcal{C}$ that takes each C/x to C is a discrete fibration. Likewise, letting $\mathcal{E} \xrightarrow{F} \mathcal{C}$ be a discrete fibration, we may define a presheaf $F_\bullet : \mathbf{Pr}(\mathcal{C})$ in which each F_C is the pullback of F along $\Delta^0 \xrightarrow{!_C} \mathcal{C}$.*

As might be expected, the assignment $X \mapsto p_X$ extends to a functor $\mathbf{Pr}(\mathcal{C}) \xrightarrow{p_\bullet} \mathbf{DF}_{\mathcal{C}}$ which is full, faithful, and essentially surjective (i.e. an equivalence of categories). Therefore $\mathbf{DF}_{\mathcal{C}}$ may be used as an alternative to $\mathbf{Pr}(\mathcal{C})$, and has its own Yoneda embedding $\mathcal{C} \xhookrightarrow{y_{\mathcal{C}}} \mathbf{DF}_{\mathcal{C}}$.

In the context of discrete fibrations, there is an alternative characterization of representable maps in addition to the Grothendieck-style extension of the essential image of the Yoneda embedding $\mathcal{C} \xhookrightarrow{y_{\mathcal{C}}} \mathbf{DF}_{\mathcal{C}}$ to maps via pullbacks.

Lemma 3.28 (Representability in discrete fibrations [ABSS14, Awo18b])**.** *A map $G \xrightarrow{f} F : \mathbf{DF}_{\mathcal{C}}$ is representable iff the* upstairs *functor $\partial_0(G) \xrightarrow{\partial_0(f)} \partial_0(F) : \mathbf{Cat}$ has a (non-fibered) right adjoint, which we may write $\partial_0(f) \dashv \mathbf{q}_f$.*

In contrast with presheaves, it is natural to simultaneously work with discrete fibrations over different base categories: we may write $\mathbf{DF} \subseteq \mathbf{Fib} \subseteq [\Delta^1, \mathbf{Cat}]$ for the category of fibrations over arbitrary base, rendered a full subcategory of the category of arrows of $\mathbf{Cat}$. Therefore, a morphism of discrete fibrations is a commuting square:

$$\begin{array}{ccc} \mathcal{E}_0 & \xrightarrow{\alpha_1} & \mathcal{E}_1 \\ {\scriptstyle p_0}\downarrow & & \downarrow{\scriptstyle p_1} \\ \mathcal{B}_0 & \xrightarrow[\alpha_0]{} & \mathcal{B}_1 \end{array}$$

Lemma 3.29. *The map sending a discrete fibration to its base category $\mathbf{DF} \xrightarrow{\partial_1} \mathbf{Cat}$ is a fibration. We will write $\mathbf{DF}_{\mathcal{C}}$ for the fiber $\partial_1[\mathcal{C}]$.*

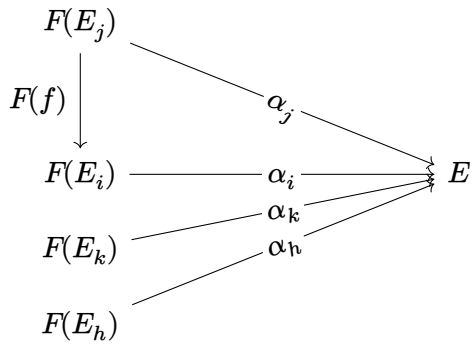
3.1.2. *Density and the dual Yoneda lemma.* The Yoneda lemma (Lemma 3.22) is indispensible, but the following dual form more deeply exposes the character of $\mathbf{Pr}(\mathcal{C})$ as free cocompletion.

Lemma 3.30 (Dual Yoneda lemma)**.** *The Yoneda embedding $\mathcal{C} \xhookrightarrow{y_{\mathcal{C}}} \mathbf{Pr}(\mathcal{C})$ is a dense functor.*

Of course, to understand Lemma 3.30 we must first have an understanding of density in the category theoretic sense. Every functor $\mathcal{C} \xrightarrow{F} \mathcal{E}$ generates a *canonical cocone* over each object $E : \mathcal{E}$ for the following diagram $F \downarrow \{E\} \xrightarrow{\mathsf{D}_F^E} \mathcal{E}$:

$$\mathsf{D}_F^E \overset{def}{=\!=} F \downarrow \{E\} \xrightarrow{\partial_0} \mathcal{C} \xrightarrow{F} \mathcal{E}$$

The canonical cocone $\mathsf{D}_F^E \xrightarrow{\delta_F^E} \{E\}$ takes each $\alpha_i : F \downarrow \{E\}$ to the underlying map $F(E_i) \xrightarrow{\alpha_i} E$. Visually, the cocone might look something like this:

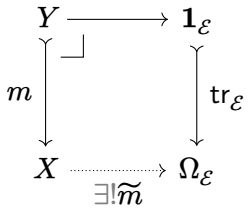


(3.31)

Definition 3.32 (Density)**.** A functor $\mathcal{C} \xrightarrow{F} \mathcal{E}$ is called *dense* if for each $E : \mathcal{E}$, the canonical cocone $\mathsf{D}_F^E \xrightarrow{\delta_F^E} \{E\}$ is universal (i.e. a colimit). ❦

Therefore, Lemma 3.30 (which might better be called the "density lemma") says exactly that every presheaf is a formal colimit of representable objects in a canonical way.

3.1.3. *Philo-logie and Diaconescu's theorem.* Categories of presheaves $\mathcal{E} = \mathbf{Pr}(\mathcal{C})$ are complete, cocomplete, locally cartesian closed, and exhibit certain non-trivial compatibilities between certain limits and colimits which may be boiled down to the existence of a classifying family for subobjects (monomorphisms). This *subobject classifier* is a monomorphism $\mathbf{1}_{\mathcal{E}} \xrightarrowtail{\mathsf{tr}_{\mathcal{E}}} \Omega_{\mathcal{E}}$ that is universal in the sense that all monomorphisms arise *in a unique way* from $\mathsf{tr}_{\mathcal{E}}$ by pullback:



(3.33)

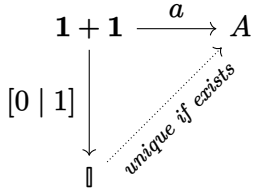
A category with these properties may be referred to as a *logos* following the terminology of Anel and Joyal [AJ19], though we defer the actual definition of logoi until slightly later. Recalling our characterization of $\mathbf{Pr}(\mathcal{C})$ as the *free* cocompletion of $\mathcal{C}$, we argue that the correct way to understand categories of presheaves is as a class of logoi that are generated in a specific way: namely, all colimits are added freely without imposing any relations.

We may also generate logoi in which the added colimits satisfy some relations; historically, the most common way to do so is to augment the category $\mathcal{C}$ with the data of a *coverage*. The canonical motivating example arises when considering presheaves on the frame of open sets $\mathcal{O}(X)$ of a topological space X. The logos of presheaves $\mathbf{Pr}(\mathcal{O}(X))$ does *not* have the

geometrically correct colimits corresponding to the gluing together of components of an open cover $\{U_i \longrightarrow U\}$, but some presheaves treat $\mathsf{y}(U)$ "as if" it were the appropriate colimit of the covering diagram to varying degrees:

(1) A presheaf X that has *no more than* one section $x \in X(U)$ compatible with a family of sections $\{x_i \in X(U_i)\}$ defined on the cover is called *separated*.
(2) A presheaf X that has exactly one such section $x \in X(U)$ for each such compatible family of sections is called *local*.

Separated presheaves will play an important role in the algebraic syntax and semantics of XTT (Section 4), in which we wish to ensure that there is at most one path $\mathbb{I} \longrightarrow A$ compatible with a boundary $\mathbf{1} + \mathbf{1} \longrightarrow A$ in the following sense:



(3.34)

While not all presheaves are local, it is possible to correct this defect by *quotienting* or *localizing* the logos, forcing certain maps out of colimits to become isomorphisms — achieved in this case by restricting to only the local presheaves. The property of the localization that enables this particularly simple computation is that it is left exact (preserves finite limits). We may therefore define a logos to be a left exact localization of the category of presheaves on a small category $\mathcal{C}$.

Definition 3.35 (Logos)**.** A logos is a left exact localization of the category of presheaves on a small category $\mathcal{C}$; a morphism $\mathcal{E} \longrightarrow \mathcal{F}$ between logoi is a functor between the underlying categories that is both left exact and cocontinuous. Such a morphism is often called an *algebraic morphism*. We will write $\mathfrak{Logos}$ for the 2-category of logoi, with 2-cells given by natural transformations between the underlying functors. ❦

Remark 3.36 (Direct image)**.** Because logoi are locally presentable as well as complete and cocomplete, every algebraic morphism $\mathcal{E} \xrightarrow{f^*} \mathcal{F}$ has a right adjoint $\mathcal{F} \xrightarrow{f_*} \mathcal{E}$. ❦

Two different base categories $\mathcal{C}, \mathcal{D}$ may yet generate the same presheaf logos; however, presheaf logoi enjoy a special relationship with their base categories embodied in Diaconescu's Theorem below [Bor94b].

Theorem 3.37 (Diaconescu's Theorem)**.** *A presheaf logos* $\mathbf{Pr}(\mathcal{C})$ *classifies* flat functors *out of* $\mathcal{C}$ *in the sense that there is an equivalence of categories* $\mathfrak{Logos}[\mathbf{Pr}(\mathcal{C}), \mathcal{E}] \simeq [\mathcal{C}, \mathcal{E}]_{\mathit{flat}}$*. In particular, each algebraic morphism* $\mathbf{Pr}(\mathcal{C}) \xrightarrow{f^*} \mathcal{E}$ *corresponds to an essentially unique flat functor* $\mathcal{C} \xrightarrow{f^* \circ \mathsf{y}_{\mathcal{C}}} \mathcal{E}$*.*

A flat functor is a generalization of the notion of left exact functor which may be used in case the domain category $\mathcal{C}$ is not finitely complete: in other words, a flat functor $\mathcal{C} \longrightarrow \mathcal{E}$ is one that preserves "even the finite limits that don't exist". Flat functors play an essential role in the general gluing theorem for models of Martin-Löf type theory into logoi developed by Sterling and Angiuli [SA20] which we have applied in this paper.

3.1.4. *Topos–logos duality.* Following the philosophy of Anel and Joyal [AJ19], we have (perhaps surprisingly to some readers) not referred to categories of presheaves as "topoi". This is because we prefer to think of a topos as a *geometrical* object, whereas a logos is *algebraic* in nature: for instance, Diaconescu's Theorem (Theorem 3.37) shows that the category of presheaves is an invariant form of the *theory* of flat functors in the style of Lawvere's functorial semantics [Law63].

It is instructive to start from the prototype of geometry–algebra duality embodied in the relationship between (sober) topological spaces and their frames of open sets; in this case, the frame of opens is an algebraic object, and the corresponding space is its geometric dual. By an analogy that may be substantiated in a precise way, a logos is the algebraic object corresponding to a topos, which is in contrast a kind of generalized space.

The duality between topoi and logoi is captured in a formal equivalence of categories $\mathrm{Sh} : \mathfrak{Topos}^{\mathrm{op}} \xrightarrow{\simeq} \mathfrak{Logos}$, taking a topos to its "logos of sheaves". While historically, many authors have thought of sheaves as local presheaves for a specific generators-and-relations presentation of a logos, Grothendieck insisted that this presentation is not at all the main object of study [AGV72, Gro86]: sheaves should be thought of as algebraic data varying over a (generalized) space, including both topological spaces *and* topoi. Indeed, it is enough to consider sheaves on topoi, recalling that any sober space corresponds to an essentially unique *enveloping topos* [AJ19].

On the other hand, the algebraic perspective of *logoi* and their algebraic morphisms is the most germane to this paper, so we do not refer to topoi in subsequent sections.

3.1.5. *Elementary logoi and logical morphisms.* While the logos is algebraic in nature, there is a different incarnation of the concept (due to Lawvere and Tierney) emphasizing the *higher-order logic* of subobjects. The difference between logic and algebra is at the root a question of morphisms: the appropriate morphisms between logoi *qua* algebra are the ones corresponding under functorial semantics to models of the classified theories, but these morphisms do not even preserve logical implication, much less other important logical notions like universal quantification and the subobject classifier.

In contrast to the needs of geometry, higher-order logic is usually defined with finitary disjunction only; moreover, in the study of higher-order logic, the presentability of a logos by generators and relations is a limiting factor (with realizability logoi furnishing the most well-known counterexample). For this reason, Lawvere and Tierney introduced the *elementary topos* (elementary logos), a generalization of *logos* requiring neither cocompleteness nor presentability in the sense described. To avoid confusion, many authors use the term *Grothendieck topos/logos* to distinguish the geometrical/algebraical concept from the one of Lawvere and Tierney.

The appropriate kind of morphism between these elementary logoi is the *logical morphism*, which preserves everything in sight (including the subobject classifier). While logical morphisms only very rarely arise in nature, they play an important role in the theory of gluing; in this paper, we will consider both algebraic and logical morphisms between *Grothendieck* logoi.

3.2. **Functorial semantics of dependent type theory à la Uemura.** Classically, the notion of an "algebraic theory" was understood in terms of sets of operations and equations. For instance, the theory of monoids may be written in terms of two operations: ε of arity

[0] and $\odot$ of arity [2]. Then, a set of equations is imposed on the set of trees generated by $\{\varepsilon, \odot\}$ to express the associativity and unit laws of a monoid. A model of a theory in this old-fashioned sense was then a *structure* comprising the following data:

(1) a carrier set A,
(2) a map $\varepsilon_A : 1 \to A$,
(3) a map $\odot_A : A \times A \to A$,
(4) subject to the following equations:

$$\odot_A(\varepsilon_A, x) = x$$
$$\odot_A(x, \varepsilon_A) = x$$
$$\odot_A(\odot_A(x, y), z) = \odot_A(x, \odot_A(y, z))$$

Models in this sense arrange themselves into a category, with morphisms given by functions between carrier sets that commute with the specified maps; of course, this is nothing more than the category of monoids in **Set**. Generally, one considers models in categories other than **Set**, a notion that makes sense for any category $\mathcal{C}$ having the requisite structure (in this case, finite products).

However, there are many other collections of operations and axioms that equally well express the concept of *monoid*, evidently exhibiting the same categories of models. For instance, one may use the set of operations $\{\mathrm{list}_n \mid n \in \mathbb{N}\} \cup \{\odot\}$ where each operation list_n has arity $[n]$. Lawvere famously observed that none of the important computations in universal algebra actually depend on which specific operations and axioms are used to encode a theory, advocating a perspective that regards the presentations above *not* as the theories themselves, but as structures lying over the theories.

A *theory* for Lawvere is a category $\mathbb{T}$ closed under certain structures (e.g. finite products, finite limits, etc.); a model of a theory in a category $\mathcal{C}$ is a functor $\mathbb{T} \longrightarrow \mathcal{C}$ preserving (finite products, finite limits, etc.). A collection of operations and axioms that generates a theory is called a *equational presentation* of that theory.

The approach of Lawvere, in which theories and their models are rendered functorially, is called the *functorial semantics* [Law63]. Lawvere has observed that many of the fundamental operations by which new theories are constructed from old theories are unnatural to describe in terms of presentations, but are simple at the level of categories and functors. One may continue to *present* theories $\mathbb{T}$ by sets of operations and axioms as before, but the spirit of the functorial method is to freely adopt whichever presentation is most useful in a specific context.

3.2.1. *Natural models, the judgmental essence of strict type theory.* We recall from Definition 3.24 the notion of a *representable natural transformation* of presheaves: it is a family of presheaves that, at every fiber over a representable object, is a representable object. This notion, which first arose in the context of algebraic geometry in the Grothendieck school [AGV72], plays a fundamental role in the semantics of dependent type theory as well as the categorical study of set theory and universes [Awo18b, Awo08, Str05, Str14].

From the type theoretic perspective, the importance of representable maps is easy to explain. In type theory, the basic objects under consideration are contexts Γ, types in context $\Gamma \vdash A$, and typed terms in context $\Gamma \vdash a : A$. The collection of types carries an action for each substitution $\Delta \xrightarrow{\gamma} \Gamma$ of contexts, and so does the collection of elements: we have $\Delta \vdash \gamma^* A$ and $\Delta \vdash \gamma^* a : \gamma^* A$. Moreover, while there is no context that represents the

collection of all types, for any specific type $\Gamma \vdash A$, we have a context $\Gamma.A$ that represents the *elements* of the type A.

This type theoretic situation can be captured mathematically in three steps:

(1) First of all, the collection of contexts may be organized into a category $\mathcal{C}$ with morphisms given by simultaneous substitutions.
(2) Next, the collection of types may be viewed as a presheaf $\mathbf{T}_{\mathcal{C}} : \mathbf{Pr}(\mathcal{C})$: a section $A \in \mathbf{T}_{\mathcal{C}}(\Gamma)$ is exactly a type $\Gamma \vdash A$, and the functorial action implements substitution on types. Likewise, the collection of typed elements is a presheaf indexed in the presheaf of types, i.e. a family $\widetilde{\mathbf{T}}_{\mathcal{C}} \xrightarrow{\tau_{\mathcal{C}}} \mathbf{T}_{\mathcal{C}}$.
(3) From the perspective of the Yoneda lemma, one may think of contexts as representable presheaves $\Gamma : \mathbf{Pr}(\mathcal{C})$, and a type in context Γ is a morphism $\Gamma \xrightarrow{A} \mathbf{T}_{\mathcal{C}}$. We therefore obtain representing contexts $\Gamma.A$ for each such type A by requiring that the family $\widetilde{\mathbf{T}}_{\mathcal{C}} \xrightarrow{\tau_{\mathcal{C}}} \mathbf{T}_{\mathcal{C}}$ be a representable natural transformation:

$$\begin{array}{ccc} \Gamma.A & \xrightarrow{q_A} & \widetilde{\mathbf{T}}_{\mathcal{C}} \\ {\scriptstyle p_A}\downarrow & \lrcorner & \downarrow{\scriptstyle \tau_{\mathcal{C}}} \\ \Gamma & \xrightarrow[A]{} & \mathbf{T}_{\mathcal{C}} \end{array} \tag{3.38}$$

The map $\Gamma.A \xrightarrow{p_A} \Gamma$ is the *weakening* substitution, and the map $\Gamma.A \xrightarrow{q_A} \widetilde{\mathbf{T}}_{\mathcal{C}}$ is the *variable* term. By chasing Diagram 3.38, it is easy to see that the type of the term q_A is $p_A^* A$ as expected. The structure defined above captures the decisive judgmental aspects of dependent type theory, and has been referred to by Awodey as a *natural model*: natural in both the informal sense, and in the sense that it is defined in terms of a representable *natural* transformation.

Definition 3.39 [Awo18b]**.** A natural model is a category $\mathcal{C}$ with a terminal object, together with a representable natural transformation in $\mathbf{Pr}(\mathcal{C})$. ❦

3.2.2. *Representable map categories and the semantics of type theory.* A given type theory is more than just a natural model in the sense of Definition 3.39: one must also specify other generators, such as type connectives and their elements. Most type connectives can be specified by writing down a family in $\mathbf{Pr}(\mathcal{C})$ and then asking for a cartesian square between that family and the natural model. This raises some questions:

(1) What kinds of structures can be added to the notion of a natural model and still give rise to a type theory?
(2) What is a morphism between models of such a type theory?

Uemura proposes to answer this question by developing a notion of "general type theory" and associated functorial semantics [Uem19]; while Lawvere defines an algebraic theory to be (roughly) a category with finite products, Uemura defines a *type* theory $\mathbb{T}$ to be a *representable map category*, which is a lex category $\mathcal{C}$ together with a distinguished class $\mathcal{R}$ of "representable maps" which captures the decisive aspects of the class of representable natural transformations in presheaf logoi.

Definition 3.40. A class of representable maps in a lex category $\mathcal{C}$ is a collection of maps $\mathcal{R}$ with the following closure conditions:

(1) Each identity map is representable, and the composition of representable maps is representable.
(2) Every pullback of a representable map along an arbitrary map is representable.
(3) The pushforward of an arbitrary map along a representable map exists (but is not necessarily representable). ❦

Notation 3.41. In particular, if the terminal map $X \longrightarrow \mathbf{1}_{\mathcal{C}}$ is representable, then all exponentials (internal homs) out of X may be computed by pushforward, which we may write either $X \Rightarrow Y$ or $[\![X, Y]\!]$. ❦

Definition 3.42. A representable map category is a lex category $\mathcal{C}$ together with a class $\mathcal{R}$ is representable maps in $\mathcal{C}$; a *representable map functor* between two representable map categories is a functor between the underlying categories that takes representable maps to representable maps. ❦

Remark 3.43. Every lex category $\mathcal{C}$ can be thought of as a very weak kind of extensional type theory, in which dependent types are maps and substitutions are given by pullback. A representable map structure on $\mathcal{C}$ does nothing more than enrich this language as follows:

(1) There is a class of *small* types called "representable" types.
(2) Representable types are closed under dependent sums.
(3) Types are closed under dependent products with representable base.

The "type theory" of a representable map category $\mathbb{T} = (\mathcal{C}, \mathcal{R})$ can be thought of as a logical framework in the sense of [HHP93, NPS90], in which hypothetical judgments are represented by pushforward. *From this perspective, it is most appropriate to refer to arbitrary objects and families in $\mathbb{T}$ as judgments, following the "judgments as types" philosophy of Harper, Honsell, and Plotkin* [HHP93]. ❦

Example 3.44 (The walking natural model)**.** The type theory with judgments for types and terms is specified by the representable map category $\mathbb{T}$ generated by a single representable map $\widetilde{\mathbf{T}} \xrightarrow{\tau} \mathbf{T}$. ❦

Example 3.45. $\mathbf{Pr}(\mathcal{C})$ supports the structure of a representable map category with the class of representable maps given by representable natural transformations. ❦

The classic notion of a representable map in a category of presheaves is of course an instance, giving rise to the "canonical representable structures" on $\mathbf{Pr}(\mathcal{C})$ and $\mathbf{DF}_{\mathcal{C}}$.

Example 3.46. $\mathbf{DF}_{\mathcal{C}}$ supports the structure of a representable map category, with representable maps given by a functors between discrete fibrations that have a non-fibered right adjoint [Awo18b, ABSS14]. Explicitly, a morphism $X \xrightarrow{f} Y : \mathbf{DF}_{\mathcal{C}}$ is a representable morphism if there is a functor $Y \xrightarrow{\mathbf{q}_f} X : \mathbf{Cat}$ such that $f \dashv \mathbf{q}_f$.

Moreover, with this class of representable maps, the equivalence between $\mathbf{DF}_{\mathcal{C}}$ and $\mathbf{Pr}(\mathcal{C})$ induces an equivalence between representable map categories. In other words, viewing a representable natural transformation of presheaves as a functor between categories of elements, one has such a right adjoint, and vice versa. ❦

The functorial semantics of type theories à la Uemura is then given in terms of these canonical representable map categories:

Definition 3.47 (Models)**.** A model of a type theory $\mathbb{T} = (\mathcal{C}, \mathcal{R})$ is a representable map functor $\mathbb{T} \xrightarrow{\mathcal{M}_\mathcal{C}} \mathbf{DF}_\mathcal{C}$ where $\mathcal{C}$ is a category with a terminal object. ❦

In Definition 3.47, the base category $\mathcal{C}$ is the category of contexts, and each $\mathcal{M}_\mathcal{C}(J)$ is the interpretation of a "judgment" $J : \mathbb{T}$ as a presheaf over $\mathcal{C}$.

Notation 3.48. Given any object $J : \mathbb{T}$, we write $J_\mathcal{C}$ for $\mathcal{M}_\mathcal{C}(J)$; likewise, for each $J \xrightarrow{f} I : \mathbb{T}$, we write $J_\mathcal{C} \xrightarrow{f_\mathcal{C}} I_\mathcal{C} : \mathbf{DF}_\mathcal{C}$ for $\mathcal{M}_\mathcal{C}(f)$. ❦

To a first approximation, a morphism $\mathcal{M}_F$ between two $\mathbb{T}$-models $\mathcal{M}_\mathcal{C}, \mathcal{M}_\mathcal{D}$ should be a functor $\mathcal{C} \xrightarrow{F} \mathcal{D}$ together with a natural assignment of functors $X \mapsto X_F$ for each $X : \mathbb{T}$:

$$\begin{array}{ccc} X_\mathcal{C} & \xrightarrow{X_F} & X_\mathcal{D} \\ \downarrow & & \downarrow \\ \mathcal{C} & \xrightarrow[F]{} & \mathcal{D} \end{array} \qquad \begin{array}{c} \mathbf{DF} \\ \downarrow \\ \mathbf{Cat} \end{array} \tag{$*$}$$

In addition to these requirements, a morphism of models should also preserve *context extension* up to isomorphism; from a categorical perspective, there is not much meaning in asking for context extension to be preserved "on the nose", since contexts are *objects* of a category and therefore only well-defined up to isomorphism. Therefore, given a context $C : \mathcal{C}$ and a type $\mathsf{y}(C) \xrightarrow{A} \mathbf{T}_\mathcal{C}$, we would expect that the context extension $F(C).\mathbf{T}_F(A)$ shall be isomorphic to $F(C.A)$.

In Uemura's framework, as with natural models [Awo18a], context extension is modeled by the representability of $\widetilde{\mathbf{T}} \xrightarrow{\tau} \mathbf{T}$. In fact, calculation shows that the (non-fibered) right adjoint $\mathbf{q}_{\tau_\mathcal{C}}$ to $\tau_\mathcal{C}$ sends a type $\mathsf{y}_\mathcal{C}(C) \xrightarrow{A} \mathbf{T}_\mathcal{C}$ to the variable term $\mathsf{y}_\mathcal{C}(C.A) \longrightarrow \widetilde{\mathbf{T}}_\mathcal{C}$ in the extended context. We may therefore phrase the preservation of context extensions, called *Beck-Chevalley* by Uemura, in terms of $\mathbf{q}_{f_\mathcal{C}}$ and $\mathbf{q}_{f_\mathcal{D}}$ for *each* representable map f, including $\widetilde{\mathbf{T}} \xrightarrow{\tau} \mathbf{T}$.

First, observe that for each representable map $J \xrightarrow{f} I$, we have a canonical 2-cell in $\mathbf{Cat}$ with the following boundary:

$$\begin{array}{ccc} I_\mathcal{C} & \xrightarrow{I_F} & I_\mathcal{D} \\ {\scriptstyle \mathbf{q}_{f_\mathcal{C}}}\downarrow & \Rightarrow & \downarrow{\scriptstyle \mathbf{q}_{f_\mathcal{D}}} \\ J_\mathcal{C} & \xrightarrow[J_F]{} & J_\mathcal{D} \end{array} \tag{3.49}$$

The 2-cell of Diagram 3.49 may be computed as follows:

$$\dfrac{\dfrac{\dfrac{\overline{f_\mathcal{C} \circ \mathbf{q}_{f_\mathcal{C}} \longrightarrow \mathbf{id}_{I_\mathcal{C}}}}{I_F \circ f_\mathcal{C} \circ \mathbf{q}_{f_\mathcal{C}} \longrightarrow I_F}}{f_\mathcal{D} \circ J_F \circ \mathbf{q}_{f_\mathcal{C}} \longrightarrow I_F}}{J_F \circ \mathbf{q}_{f_\mathcal{C}} \longrightarrow \mathbf{q}_{f_\mathcal{D}} \circ I_F}$$

The Beck-Chevalley condition for $J \xrightarrow{f} I$ is, then, that the 2-cell in Diagram 3.49 is invertible.

Definition 3.50 (Morphism of Models)**.** A morphism between $\mathbb{T} \xrightarrow{\mathcal{M}_{\mathcal{C}}} \mathbf{DF}_{\mathcal{C}}$ and $\mathbb{T} \xrightarrow{\mathcal{M}_{\mathcal{D}}} \mathbf{DF}_{\mathcal{D}}$ is a functor $\mathcal{C} \xrightarrow{F} \mathcal{D}$ preserving the terminal object, together with an assignment of functors $X_{\mathcal{C}} \xrightarrow{X_F} X_{\mathcal{D}}$ lying over F, natural in $X : \mathbb{T}$, such that each representable map $X \xrightarrow{f} Y$ satisfies the Beck-Chevalley condition. ❦

Notation 3.51. Let $\mathcal{M}_{\mathcal{C}} \xrightarrow{\mathcal{M}_F} \mathcal{M}_{\mathcal{D}}$ be a morphism of $\mathbb{T}$-models; given $X : \mathbb{T}, x : X_{\mathcal{C}}$, we will abusively write $F(x)$ for $X_F(x)$. ❦

Because morphisms of models necessarily require preservation of context extensions only up to isomorphism, a higher level of morphism naturally arises, lending the collection of models $\mathbf{Mod}_{\mathbb{T}}$ with the structure of a 2-category.

Definition 3.52 (2-morphisms of models)**.** Given morphisms $\mathcal{M}_{\mathcal{C}} \xrightarrow{\mathcal{M}_F, \mathcal{M}_G} \mathcal{M}_{\mathcal{D}}$, a 2-morphism $\mathcal{M}_F \xrightarrow{\mathcal{M}_\alpha} \mathcal{M}_G$ is a natural transformation $F \xrightarrow{\alpha} G$ between the underlying functors such that for each $X : \mathbb{T}$, there exists a *necessarily unique* natural transformation $X_F \xrightarrow{X_\alpha} X_G$ lying over α. ❦

Remark 3.53. The uniqueness of the map $X_F \xrightarrow{X_\alpha} X_G$ lying over α is ensured by the discreteness of $X_{\mathcal{D}}$. Summarizing, the existence of X_α is nothing more than the *condition* that for each $x : X_{\mathcal{C}}$ lying over $\Gamma : \mathcal{C}$, the following equation obtains:

$$\alpha^*_\Gamma(G(x)) = F(x)$$ ❦

Theorem 3.54. *The 2-category of models* $\mathbf{Mod}_{\mathbb{T}}$ *has a* bi-initial *object: an object whose under-categories are contractible.*

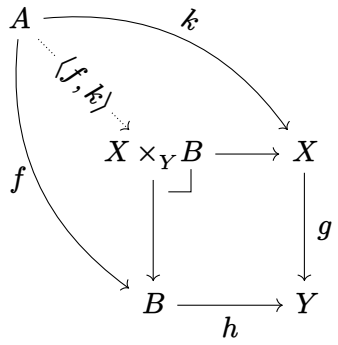
This bi-initial object is the *democratic heart* of the embedding $\mathbb{T} \xrightarrow{\mathsf{y}} \mathbf{DF}_{\mathbb{T}}$: the smallest full subcategory of $\mathbb{T}$ containing the terminal object and closed under context extension. The universal property of this bi-initial object ensures that there is at most one morphism $\mathcal{M}_{\mathcal{J}} \longrightarrow \mathcal{M}_{\mathcal{C}}$ for each $\mathcal{M}_{\mathcal{C}}$, up to a unique invertible 2-morphism.

3.3. **Left lifting structures and orthogonality.** In this section, let $\mathcal{E}$ be a category with finite limits. We will say that an object $X : \mathcal{E}$ is *exponentiable* when every exponential $[\![X, Y]\!]$ exists. We begin by recalling some basic definitions and facts from [Awo18b, ABFJ17, ABFJ17].

Definition 3.55 (Cartesian gap map)**.** Let the following square commute in $\mathcal{E}$:

$$\begin{array}{ccc} A & \xrightarrow{k} & X \\ {\scriptstyle f}\downarrow & & \downarrow{\scriptstyle g} \\ B & \xrightarrow[h]{} & Y \end{array} \tag{3.56}$$

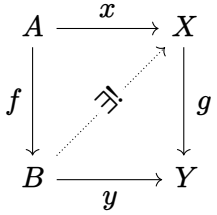
The universal property of the pullback of g along h states that the span f, k induces a map $\langle f, k \rangle$, which we call the *cartesian gap map* of Diagram 3.56:



Definition 3.57 (Internal pullback hom)**.** Let $A \xrightarrow{f} B$ and $X \xrightarrow{g} Y$ be morphisms in $\mathcal{E}$ such that A, B are both exponentiable. Then, the *internal pullback hom* of f and g is the cartesian gap map $[\![B, X]\!] \xrightarrow{\langle\!\langle f, g \rangle\!\rangle} [\![A, X]\!] \times_{[\![A,Y]\!]} [\![B, Y]\!]$ of the following square:

$$\begin{array}{ccc} [\![B, X]\!] & \xrightarrow{[\![f, X]\!]} & [\![A, X]\!] \\ {\scriptstyle [\![B,g]\!]}\downarrow & & \downarrow{\scriptstyle [\![A,g]\!]} \\ [\![B, Y]\!] & \xrightarrow[[\![f,Y]\!]]{} & [\![A, Y]\!] \end{array}$$

Definition 3.58 (External orthogonality)**.** A map $A \xrightarrow{f} B$ is called *left orthogonal* to $X \xrightarrow{g} Y$, written $f \perp g$, when there exists a unique lift for each square of the following kind:



Definition 3.59 (Internal orthogonality)**.** A map $A \xrightarrow{f} B$ is *internally left orthogonal* to $X \xrightarrow{g} Y$, written $f \pitchfork g$, if we have $(Z \times f) \perp g$ for every $Z : \mathcal{E}$.

Lemma 3.60 [ABFJ17, Definition 3.2.5]**.** *Fix $A \xrightarrow{f} B$ and $X \xrightarrow{g} Y$ with A, B exponentiable; then $f \pitchfork g$ iff the internal pullback hom $\langle\!\langle f, g \rangle\!\rangle$ is an isomorphism.* □

Definition 3.61 [Awo18b, Definition 18]**.** Fix $A \xrightarrow{f} B$ and $X \xrightarrow{g} Y$ with A, B exponentiable; a *left lifting structure* for f against g (written $f \Uparrow g$) is a section j of the internal

pullback hom $\langle\!\langle f, g \rangle\!\rangle$:

$$\begin{array}{ccc} & & [\![B, X]\!] \\ & j \curvearrowright & \downarrow \\ & & \langle\!\langle f, g \rangle\!\rangle \\ & & \downarrow \\ [\![A, X]\!] & \times_{[\![A, Y]\!]} & [\![B, Y]\!] \end{array}$$

❦

Example 3.62 (Intensional identity types)**.** In order to formulate the elimination of intensional identity types in a natural model, Awodey [Awo18a] uses a left lifting structure. More generally, most "pattern-matching" style elimination rules in type theory can be formulated as a left lifting structure [GKNB20]. ❦

Lemma 3.63 [Awo18b, Lemma 19]**.** *Given maps $A \xrightarrow{f} B$ and $X \xrightarrow{g} Y$, a left lifting structure $j : f \pitchfork g$ is equivalent to a choice of lifts $j_Z(y, x)$ natural in Z for any diagram of the following kind:*

$$\begin{array}{ccc} Z \times A & \xrightarrow{x} & X \\ {\scriptstyle Z \times f}\downarrow & \overset{j_Z(y,x)}{\nearrow} & \downarrow{\scriptstyle g} \\ Z \times B & \xrightarrow[y]{} & Y \end{array}$$

□

Remark 3.64. A left lifting structure $j : f \pitchfork g$ exhibits f as internally left orthogonal to g when j is simultaneously a retraction. ❦

Because both left lifting structures and orthogonality conditions may be expressed in the language of finite limits as above, it is justified to freely extend a representable map category $\mathbb{T}$ by either $f \doteq g$ or $j : f \pitchfork g$. In many cases, however, an orthogonality condition or lifting structure will need to be expressed in the free cocompletion $\mathbf{Pr}(\mathbb{T})$ because it may involve colimits that don't exist in $\mathbb{T}$; when defining a representable map category by a sequence of clauses, it is not *a priori* clear that this move is legitimate.

We will therefore characterize a useful class of orthogonality and lifting conditions on $\mathbf{Pr}(\mathcal{C})$ which may be unravelled into a suitable condition on a lex category $\mathcal{C}$, expressible using the language of finite limits.

Lemma 3.65. *Let $\mathcal{I}$ be a small category such that limits of $\mathcal{I}$-diagrams exist in $\mathcal{C}$; let $\mathcal{I} \xrightarrow{\Phi_\bullet} \mathcal{C}$ be a diagram such that for each $i : \mathcal{I}$, the object Φ_i is exponentiable. We will write $\hat{\Phi}_\infty$ for the colimit of $\Phi_\bullet$ taken in $\mathbf{Pr}(\mathcal{C})$, i.e. $\hat{\Phi}_\infty = \operatorname{colim}_{\mathcal{I}}(\mathsf{y}(\Phi_\bullet))$. Now let $\hat{\Phi}_\infty \xrightarrow{f} \mathsf{y}(B) : \mathbf{Pr}(\mathcal{C})$ with B exponentiable, and let $X \xrightarrow{g} Y$ be an arbitrary map in $\mathcal{C}$. Then, there exists a left lifting structure $j : f \pitchfork \mathsf{y}(g)$ in $\mathbb{T}$ iff there exists a section of the cartesian gap map for the canonical Diagram 3.66 in $\mathcal{C}$ below:*

$$\begin{array}{ccc} [\![B, X]\!] & \longrightarrow & \lim_{\mathcal{I}} [\![\Phi_\bullet, X]\!] \\ \downarrow & & \downarrow \\ [\![B, Y]\!] & \longrightarrow & \lim_{\mathcal{I}} [\![\Phi_\bullet, Y]\!] \end{array} \tag{3.66}$$

Moreover, the left lifting structure j exhibits f as internally left orthogonal to $\mathsf{y}(g)$ iff the cartesian gap map of Diagram 3.66 is an isomorphism.

Before proving Lemma 3.65, we first clarify the construction of Diagram 3.66. For each $i : \mathcal{I}$, we have the following composite map:

$$\begin{array}{ccc} \mathrm{y}(\Phi_i) & \xrightarrow{\mathrm{in}_i} & \hat{\Phi}_\infty \\ & \searrow^{y(f_i)} & \downarrow f \\ & & \mathrm{y}(B) \end{array} \tag{3.67}$$

Therefore, letting Z range over X, Y, we may construct a map into $\lim_{\mathcal{I}} [\![\Phi_\bullet, Z]\!]$ from a cone defined as follows:

$$\begin{array}{ccc} [\![B, Z]\!] & \xrightarrow{[i.[\![f_i, Z]\!]]} & \lim_{\mathcal{I}} [\![\Phi_\bullet, Z]\!] \\ & \searrow^{[\![f_i, Z]\!]} & \downarrow \pi_i \\ & & [\![\Phi_i, Z]\!] \end{array} \tag{3.68}$$

The force of Lemma 3.65 is therefore to assert that the existence of a left lifting structure $j : f \pitchfork \mathrm{y}(g)$ in $\mathbf{Pr}(\mathcal{C})$ is equivalent to the existence of a section to the cartesian gap map of Diagram 3.69 below:

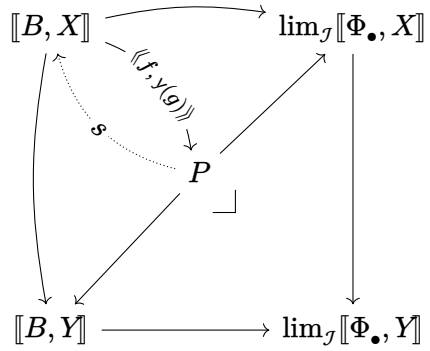
$$\begin{array}{ccc} [\![B, X]\!] & \xrightarrow{[i.[\![f_i, X]\!]]} & \lim_{\mathcal{I}} [\![\Phi_\bullet, X]\!] \\ {\scriptstyle [\![B, g]\!]}\downarrow & & \downarrow{\scriptstyle \lim_{\mathcal{I}} [\![\Phi_\bullet, g]\!]} \\ [\![B, Y]\!] & \xrightarrow[[i.[\![f_i, Y]\!]]]{} & \lim_{\mathcal{I}} [\![\Phi_\bullet, Y]\!] \end{array} \tag{3.69}$$

Proof of Lemma 3.65. The condition of the internal pullback hom $\langle\!\langle f, \mathrm{y}(g) \rangle\!\rangle$ may be portrayed as follows:

$$\begin{array}{ccc} \mathrm{y}([\![B, X]\!]) & \longrightarrow & [\![\hat{\Phi}_\infty, \mathrm{y}(X)]\!] \\ \downarrow & {\scriptstyle \langle\!\langle f, \mathrm{y}(g) \rangle\!\rangle} \swarrow \; \nwarrow^{s} & \uparrow \\ & P \; \lrcorner & \downarrow \\ \mathrm{y}([\![B, Y]\!]) & \longrightarrow & [\![\hat{\Phi}_\infty, \mathrm{y}(Y)]\!] \end{array} \tag{3.70}$$

But for each $Z : \mathcal{C}$, the presheaf $[\![\hat{\Phi}_\infty, \mathrm{y}(Z)]\!]$ is canonically represented by the object $\lim_{\mathcal{I}} [\![\Phi_\bullet, Z]\!] : \mathcal{C}$, using the universality of colimits in the logos $\mathbf{Pr}(\mathcal{C})$ and the fact that the Yoneda embedding commutes with limits and exponentials. Therefore, Diagram 3.71 below

faithfully translates the existence of the dotted map in Diagram 3.70 into the language of $\mathcal{C}$:

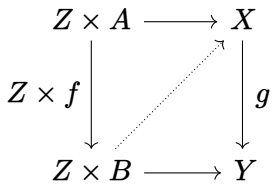


(3.71)

But the above is exactly the existence of a section to the cartesian gap map for Diagram 3.69. It is likewise easy to see that one section is a retraction iff the other is. □

3.4. **Separation.** We will have need of an orthogonality-like notion in which lifts may not exist, but when they do, they are unique. It is most appropriate to call this condition *separation*, by analogy with coverages and Grothendieck topologies.

Definition 3.72. Let $A \xrightarrow{f} B$ and $X \xrightarrow{g} Y$ be maps in a category $\mathcal{C}$; then we say that g is *separated* with respect to f when, for every object $Z : \mathcal{C}$, there is at most one lift for any square of the following shape:



❦

4. Functorial semantics of XTT

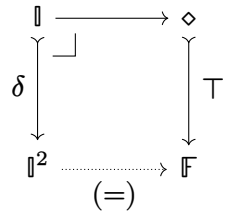
In this section, we define the classifying representable map category $\mathbb{T}$ of XTT and write $\diamond : \mathbb{T}$ for its terminal object.

Specification 4.1 (Judgmental structure)**.** The basic judgmental structure of XTT is specified below.

(1) A representable map $\widetilde{\mathbf{T}} \xrightarrow{\tau} \mathbf{T}$ which encodes the collection of typed elements lying over typed terms; the representability of τ allows the (abstract) context to be extended by an element $x : A$ of a type A.
(2) A representable map $\mathbb{I} \longrightarrow \diamond$, implementing the interval and its context extension.
(3) A representable map $\diamond \xrightarrow{\top} \mathbb{F}$, implementing the face formula judgment and its context extension. $\top$ is the "true" face formula. As a map out of the terminal object, $\top$ is automatically monomorphic: this means that two proofs of the same face formula are identical. ❦

Specification 4.2 (The interval)**.** We require only minimal structure on the interval, the two endpoints $\diamond \xrightarrow{0,1} \mathbb{I}$. ❦

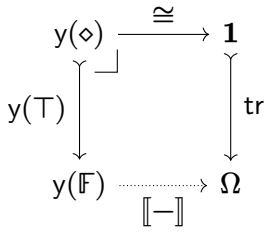
Specification 4.3 (Dimension equality)**.** On the face formulas we place the following structure, writing $\mathbb{I} \xrightarrow{\delta} \mathbb{I} \times \mathbb{I}$ for the diagonal map:



❦

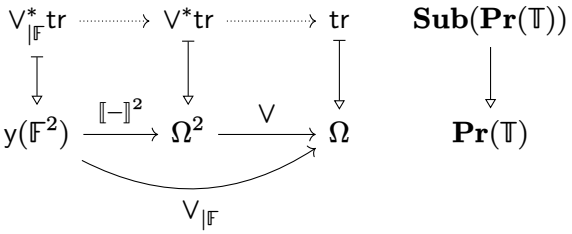
We must be careful when specifying disjunction of face formulas; the "true" disjunction of $\top[\phi]$ and $\top[\psi]$ ought to be a pushout, but we don't expect to have pushouts in $\mathbb{T}$, and moreover, we do not wish to require the ability to "split" on a disjunction in all the syntactic sorts of our type theory. The situation can be dealt with by making $\widetilde{\mathbf{T}} \xrightarrow{\tau} \mathbf{T}$ *think* that this pushout exists, in a way that we will make precise forthwith.

We begin by formulating the "true" disjunction in the free cocompletion $\mathbf{Pr}(\mathbb{T})$. First, we have a characteristic map that decodes a face condition $\phi : \mathsf{y}(\mathbb{F})$ to a proposition $[\![\phi]\!] : \Omega$.



(4.4)

As a first cut toward disjunction, we may define the *non-representable* subobject $\bigvee^*_{|\mathbb{F}}\mathsf{tr}$ of true disjunctions of face conditions:



(4.5)

We will then define the disjunction of face conditions to be a representable approximation of $\bigvee^*_{|\mathbb{F}}\mathsf{tr}$ that is respected by certain judgments of XTT.

Specification 4.6 (Disjunction)**.** We require a face condition formation map $\mathbb{F}^2 \xrightarrow{\vee} \mathbb{F} : \mathbb{T}$ satisfying some conditions which we will describe forthwith. We then require an "introduction" rule $\bigvee^*_{|\mathbb{F}}\mathsf{tr} \xrightarrow{\mathsf{in}_\vee} \mathsf{y}(\vee^*\top)$ in the slice $\mathbf{Pr}(\mathbb{T})_{/\mathsf{y}(\mathbb{F}^2)}$. The "elimination" rules for the disjunction are then expressed as a pair of internal orthogonality conditions in $\mathbf{Pr}(\mathbb{T})_{/\mathsf{y}(\mathbb{F}^2)}$:

(1) We require $\mathsf{in}_\vee \doteq \mathsf{y}(\mathbb{F}^2)^*\mathsf{y}(\top)$ in $\mathbf{Pr}(\mathbb{T})_{/\mathsf{y}(\mathbb{F}^2)}$, ensuring that the truth of a face condition may be established by eliminating a disjunction.
(2) We must have $\mathsf{in}_\vee \doteq \mathsf{y}(\mathbb{F}^2)^*\mathsf{y}(\tau)$ in $\mathbf{Pr}(\mathbb{T})_{/\mathsf{y}(\mathbb{F}^2)}$, ensuring that a "matching family" for a term on two disjuncts shall be a term under a disjunction. ❦

Specification 4.7 (Falsehood)**.** By Specification 4.3, we have a map $\diamond \xrightarrow{\bot \overset{def}{=\!=} (=) \circ \langle 0, 1 \rangle} \mathbb{F}$. We may therefore test the truth of this "false" equation:

$$\begin{array}{ccc} \bot^*\top & \rightarrowtail & \top \\ \downarrow & & \downarrow \\ \diamond & \xrightarrow[\bot]{} & \mathbb{F} \end{array} \qquad\qquad \begin{array}{c} \mathbf{Sub}(\mathbb{T}) \\ \downarrow \\ \mathbb{T} \end{array} \tag{4.8}$$

There is a universal comparison map $\emptyset \xrightarrow{\mathrm{in}_\bot} \mathsf{y}(\bot^*\top)$ in $\mathbf{Pr}(\mathbb{T})$ given by the universal property of the initial object; to support the "elimination rule" of the inconsistent face condition, then, we require orthogonalities $\mathrm{in}_\bot \mathrel{\mathop:}\perp \mathsf{y}(\top)$ and $\mathrm{in}_\bot \mathrel{\mathop:}\perp \mathsf{y}(\tau)$ in $\mathbf{Pr}(\mathbb{T})$. ❦

Remark 4.9. The orthogonality conditions from Specification 4.6 may be restated in the language of $\mathbb{T}_{/\mathbb{F}^2}$. Let $\mathbf{Span}$ be the walking span, and let $\mathbf{Span} \xrightarrow{\Phi_\bullet} \mathbb{T}_{/\mathbb{F}^2}$ be the following diagram in $\mathbb{T}_{/\mathbb{F}^2}$:

$$\begin{array}{ccc} \top \times \top & \to & \pi_2^*\top \\ \downarrow & & \\ \pi_1^*\top & & \end{array}$$

The following canonical squares must be cartesian:

$$\begin{array}{ccc} [\![\vee^*\top, (\mathbb{F}^2)^*\diamond]\!] & \longrightarrow & \lim_{\mathbf{Span}} [\![\Phi_\bullet, (\mathbb{F}^2)^*\diamond]\!] \\ \downarrow & & \downarrow \\ [\![\vee^*\top, (\mathbb{F}^2)^*\mathbb{F}]\!] & \longrightarrow & \lim_{\mathbf{Span}} [\![\Phi_\bullet, (\mathbb{F}^2)^*\diamond]\!] \end{array} \qquad \begin{array}{ccc} [\![\vee^*\top, (\mathbb{F}^2)^*\widetilde{\mathbf{T}}]\!] & \to & \lim_{\mathbf{Span}} [\![\Phi_\bullet, (\mathbb{F}^2)^*\widetilde{\mathbf{T}}]\!] \\ \downarrow & & \downarrow \\ [\![\vee^*\top, (\mathbb{F}^2)^*\mathbf{T}]\!] & \to & \lim_{\mathbf{Span}} [\![\Phi_\bullet, (\mathbb{F}^2)^*\mathbf{T}]\!] \end{array} \quad ❦$$

Remark 4.10. The orthogonality condition of Specification 4.7 may be restated in the language of $\mathbb{T}$ as the requirement that the following canonical squares be cartesian:

$$\begin{array}{ccc} [\![\bot^*\top, \diamond]\!] & \longrightarrow & \diamond \\ \downarrow & & \downarrow \\ [\![\bot^*\top, \mathbb{F}]\!] & \longrightarrow & \diamond \end{array} \qquad \begin{array}{ccc} [\![\bot^*\top, \widetilde{\mathbf{T}}]\!] & \longrightarrow & \diamond \\ \downarrow & & \downarrow \\ [\![\bot^*\top, \mathbf{T}]\!] & \longrightarrow & \diamond \end{array} \quad ❦$$

The following notation will often be used in the internal language of $\mathbb{T}$.

Notation 4.11 (Restriction to a partial element)**.** Let $\phi : \mathbb{F}$ be a face condition, $A : \mathbf{T}$ a type, and $a : \top[\phi] \Rightarrow \tau[A]$ a partial element; we will write $\tau[A \mid \phi \to a]$ for the collection of elements of A which restrict to a on ϕ, i.e. the elements $a' : \tau[A]$ where $\lambda\_.a' = a : \top[\phi] \Rightarrow A$. ❦

Notation 4.12 (The boundary of a dimension)**.** In the internal language of $\mathbb{T}$, we will write $\partial(r) : \mathbb{F}$ for the *boundary* of a dimension $r : \mathbb{I}$, defined as the disjunction $\partial(r) \overset{def}{=\!=} (r = 0) \vee (r = 1)$. ❦

The boundary of a dimension will be used in specifying the closure under *path types*.

Specification 4.13 (Closure under connectives)**.** We will require that typing judgment $\widetilde{\mathbf{T}} \xrightarrow{\tau} \mathbf{T}$ is closed under dependent sum, dependent product, and dependent *path* types.

We first express the generic map underlying each connective, and then force it to be representable.

$$[\Delta^1, \mathbb{T}]_{cart} \xrightarrow{\bullet^\Sigma} [\Delta^1, \mathbb{T}]_{cart} \qquad [\Delta^1, \mathbb{T}]_{cart} \xrightarrow{\bullet^\Pi} [\Delta^1, \mathbb{T}]_{cart} \qquad [\Delta^1, \mathbb{T}]_{cart} \xrightarrow{\bullet^{\mathrm{P}}} [\Delta^1, \mathbb{T}]_{cart}$$

Let $Y \xrightarrow{f} X$, i.e. f an object of $[\Delta^1, \mathbb{T}]_{cart}$; we first define the bases X^Q of the functorial action f^Q where $Q \in \{\Pi, \Sigma, \mathrm{P}\}$:

$$\begin{aligned} X^\Sigma &= [A : X; B : f[A] \Rightarrow X] \\ X^\Pi &= [A : X; B : f[A] \Rightarrow X] \\ X^{\mathrm{P}} &= [A : \mathbb{I} \Rightarrow X; a : (i : \mathbb{I}, \_ : \partial(i)) \Rightarrow f[A(i)]] \end{aligned}$$

Then, we define the rest of the action in the internal language of $\mathcal{E}_{/(X^Q)}$:

$$\begin{aligned} f^\Sigma[A; B] &= [a : f[A]; b : f[B(a)]] \\ f^\Pi[A; B] &= (a : f[A]) \Rightarrow f[B(a)] \\ f^{\mathrm{P}}[A; a] &= \{p : (i : \mathbb{I}) \Rightarrow f[A(i)] \mid \lambda i.\lambda\_.p(i) = a\} \end{aligned}$$

The closure of the type theory under these connectives is then accomplished by requiring in $[\Delta^1, \mathbb{T}]_{cart}$ algebras $\tau^\Sigma \longrightarrow \tau$, $\tau^\Pi \longrightarrow \tau$, and $\tau^{\mathrm{P}} \longrightarrow \tau$. ❦

Remark 4.14. Unfolding Specification 4.13 into the language of $\mathbb{T}$, this means that we have the following cartesian squares:

$$\begin{array}{ccc} \widetilde{\mathbf{T}}^\Sigma & \xrightarrow{\text{pair}} & \widetilde{\mathbf{T}} \\ {\scriptstyle \tau^\Sigma}\downarrow & & \downarrow{\scriptstyle \tau} \\ \mathbf{T}^\Sigma & \xrightarrow[\text{sg}]{} & \mathbf{T} \end{array} \qquad \begin{array}{ccc} \widetilde{\mathbf{T}}^\Pi & \xrightarrow{\text{lam}} & \widetilde{\mathbf{T}} \\ {\scriptstyle \tau^\Pi}\downarrow & & \downarrow{\scriptstyle \tau} \\ \mathbf{T}^\Pi & \xrightarrow[\text{pi}]{} & \mathbf{T} \end{array} \qquad \begin{array}{ccc} \widetilde{\mathbf{T}}^{\mathrm{P}} & \xrightarrow{\text{abs}} & \widetilde{\mathbf{T}} \\ {\scriptstyle \tau^{\mathrm{P}}}\downarrow & & \downarrow{\scriptstyle \tau} \\ \mathbf{T}^{\mathrm{P}} & \xrightarrow[\text{path}]{} & \mathbf{T} \end{array}$$ ❦

We will add a type of *booleans*; as usual in type theory, the boolean type is *not* the coproduct of the point with itself, but a weak version thereof. The simplest way to specify a weak coproduct is by means of a left lifting structure (Definition 3.61) internal to the *free cocompletion* $\mathbf{Pr}(\mathbb{T})$ of $\mathbb{T}$.

Specification 4.15 (Booleans)**.** To specify the type of booleans, we then require a type $\diamond \xrightarrow{\text{bool}} \mathbf{T}$ together with the following objects in $\mathbf{Pr}(\mathbb{T})$:

(1) A morphism $\mathbf{2}_{\mathbf{Pr}(\mathbb{T})} \xrightarrow{i_{\text{bool}}} \mathrm{y}(\text{bool}^*\tau) : \mathbf{Pr}(\mathbb{T})$, where $\mathbf{2}_{\mathbf{Pr}(\mathbb{T})}$ is the coproduct $\mathbf{1}_{\mathbf{Pr}(\mathcal{C})} + \mathbf{1}_{\mathbf{Pr}(\mathcal{C})}$ in the presheaf category.
(2) A left lifting structure for i_{bool} with respect to $\mathrm{y}(\widetilde{\mathbf{T}} \xrightarrow{\tau} \mathbf{T})$, i.e. $\text{ind}_{\text{bool}} : i_{\text{bool}} \pitchfork \mathrm{y}(\tau)$. ❦

Remark 4.16. We may unravel Specification 4.15 into the language of $\mathbb{T}$ using Lemma 3.65. First, the morphism i_{bool} corresponds to exactly two introduction forms $\diamond \xrightarrow{\text{tt}, \text{ff}} \text{bool}^*\tau$ in $\mathbb{T}$; the left lifting structure ind_{bool} amounts to a choice of section for the cartesian gap map of

the following canonical square:

$$\begin{array}{ccc} [\![\mathrm{bool}^*\tau, \widetilde{\mathbf{T}}]\!] & \longrightarrow & \widetilde{\mathbf{T}} \times \widetilde{\mathbf{T}} \\ \downarrow & & \downarrow \\ [\![\mathrm{bool}^*\tau, \mathbf{T}]\!] & \longrightarrow & \mathbf{T} \times \mathbf{T} \end{array} \tag{4.17}$$

Translated into the type theoretic internal language of $\mathbb{T}$, this corresponds to a dependent eliminator of the following form:

$$\begin{aligned} \mathrm{ind}_{\mathrm{bool}} : (C : \tau[\mathrm{bool}] \Rightarrow \mathbf{T}, c_0 : \tau[C(\mathrm{tt})], c_1 : \tau[C(\mathrm{ff})], b : \tau[\mathrm{bool}]) &\longrightarrow \tau[C(b)] \\ C, c_0, c_1, b \mid b = \mathrm{tt} &\vdash \mathrm{ind}_{\mathrm{bool}}(C, c_0, c_1, b) = c_0 \\ C, c_0, c_1, b \mid b = \mathrm{ff} &\vdash \mathrm{ind}_{\mathrm{bool}}(C, c_0, c_1, b) = c_1 \end{aligned}$$ ❦

4.1. **Universe à la Tarski.** In this section, we will specify the closure of $\mathbb{T}$ under a universe à la Tarski of Bishop sets. First, we must define what it means semantically to be a Bishop set, in the style of Coquand [Coq17]; following previous work, we refer to this condition as *boundary separation* [SAG19].

Definition 4.18. A family $X \xrightarrow{f} Y : \mathbb{T}$ is said to be *boundary separated* when $\mathbb{I}^* f$ is separated with respect to the boundary inclusion $[i : \mathbb{I} \vdash \partial(i)] \rightarrowtail [i : \mathbb{I} \vdash \diamond]$ in the slice $\mathbb{T}_{/\mathbb{I}}$, in the sense of Definition 3.72. Unfolding, this means that for any map $Z \xrightarrow{r} \mathbb{I}$, any square of the following shape in $\mathbb{T}$ has *at most* one lift:

$$\begin{array}{ccc} [Z \vdash \partial(r)] & \longrightarrow & X \\ \downarrow & \nearrow & \downarrow f \\ [Z \vdash \diamond] & \longrightarrow & Y \end{array} \tag{4.19}$$

The concept of boundary separation can be expressed in the language of $\mathbb{T}$ as follows:

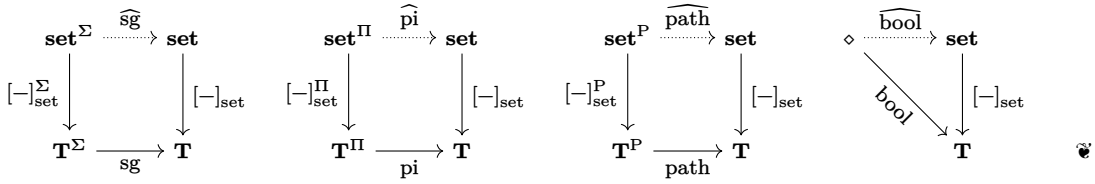
$$i : \mathbb{I}; x : X; a, b : f[x] \mid (\lambda\alpha.a) =_{(\partial(i) \Rightarrow f[x])} (\lambda\alpha.b) \vdash a =_{f[x]} b$$ ❦

Specification 4.20 (Universe à la Tarski)**.** To specify a universe à la Tarski, we require a type $\diamond \xrightarrow{\mathrm{set}} \mathbf{T}$ together with a decoding map $\mathrm{set}^*\tau \xrightarrow{[-]_{\mathrm{set}}} \mathbf{T}$. We will write **set** for the fiber $\mathrm{set}^*\tau : \mathbb{T}$. Pulling back $\widetilde{\mathbf{T}} \xrightarrow{\tau} \mathbf{T}$ along the decoding map, we have a new representable map $\tilde{\mathbf{set}} \xrightarrow{\mathbf{el}} \mathbf{set}$:

$$\begin{array}{ccc} \tilde{\mathbf{set}} & \longrightarrow & \widetilde{\mathbf{T}} \\ {\scriptstyle \mathbf{el}}\downarrow & \lrcorner & \downarrow{\scriptstyle \tau} \\ \mathbf{set} & \xrightarrow[[-]_{\mathrm{set}}]{} & \mathbf{T} \end{array} \tag{4.21}$$

We then require that the family $\tilde{\mathbf{set}} \xrightarrow{\mathbf{el}} \mathbf{set}$ be boundary separated. ❦

Specification 4.22 (Codes for connectives)**.** The closure of **set** under various connectives of type theory is accomplished as follows:



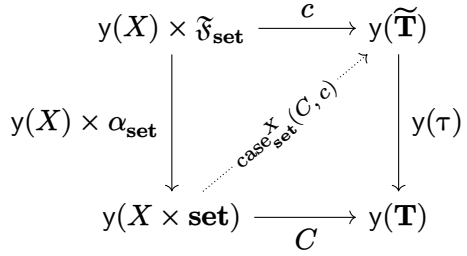
Specification 4.23 (Type-case)**.** The type-case construct of XTT's closed universe is implemented by a left lifting structure. First, we define the following coproduct in the free cocompletion $\mathbf{Pr}(\mathbb{T})$:

$$\mathfrak{F}_{\mathbf{set}} \overset{def}{=\!=} \mathsf{y}(\mathbf{set}^{\Sigma}) + \mathsf{y}(\mathbf{set}^{\Pi}) + \mathsf{y}(\mathbf{set}^{\mathrm{P}}) + \mathsf{y}(\diamond)$$

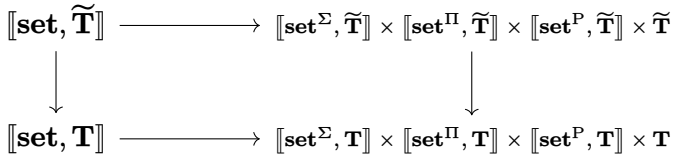
The codes for each type constructor can be arranged into an algebra $\mathfrak{F}_{\mathbf{set}} \xrightarrow{\alpha_{\mathbf{set}}} \mathsf{y}(\mathbf{set})$:

$$\mathfrak{F}_{\mathbf{set}} \xrightarrow{[\mathsf{y}(\widehat{\mathrm{sg}}) \mid \mathsf{y}(\widehat{\mathrm{pi}}) \mid \mathsf{y}(\widehat{\mathrm{path}}) \mid \mathsf{y}(\widehat{\mathrm{bool}})]} \mathsf{y}(\mathbf{set})$$

We then require a left lifting structure $\mathrm{case}_{\mathbf{set}} : \alpha_{\mathbf{set}} \pitchfork \mathsf{y}(\tau)$, which provides solutions to lifting problems of the following shape:



Remark 4.24. In the language of $\mathbb{T}$, the lifting structure from Specification 4.23 amounts to a section of the cartesian gap map for the following canonical square:



The types encoded by XTT's closed universe must be "fibrant" in the sense that they support transport along paths and composition of paths. We will express these as two separate left lifting structures, with suitable compatibility laws.

Specification 4.25. *Coercion* will be specified as a left lifting structure in the slice $\mathbb{T}_{/\mathbb{I}}$. First, observe that we have a diagonal map $[r : \mathbb{I} \vdash \diamond] \xrightarrow{[r : \mathbb{I} \vdash r]} [r : \mathbb{I} \vdash \mathbb{I}] : \mathbb{T}_{/\mathbb{I}}$. We then require a left lifting structure $\mathrm{coe} : [r : \mathbb{I} \vdash r] \pitchfork [r : \mathbb{I} \vdash \mathbf{el}]$ in $\mathbb{T}_{/\mathbb{I}}$.

Specification 4.26. *Composition* is specified by a left lifting structure in the slice $\mathbb{T}_{/\mathbb{I}^2}$: the first dimension $[r, s : \mathbb{I} \vdash r]$ plays the same role as the generic dimension in Specification 4.25, and the second dimension $[r, s : \mathbb{I} \vdash s]$ generates the boundary cofibration $[r, s : \mathbb{I} \vdash \partial(s)]$ along which we are extending a partial line.

Consider the map $[r, s : \mathbb{I} \vdash \{i : \mathbb{I} \mid \top[i = r \vee \partial(s)]\}] \rightarrowtail^{\iota} [r, s : \mathbb{I} \vdash \mathbb{I}] : \mathbb{T}_{/\mathbb{I}^2}$; we then require a left lifting structure $\mathrm{comp} : \iota \pitchfork [r, s : \mathbb{I} \vdash \mathbf{el}]$ in $\mathbb{T}_{/\mathbb{I}^2}$. ❦

Remark 4.27. Unfolding Specifications 4.25 and 4.26 into the language of $\mathbb{T}$, we have the following constants:

$$\frac{r : \mathbb{I} \qquad \hat{A} : \mathbb{I} \Rightarrow \mathbf{set} \qquad a : \mathbf{el}[\hat{A}(r)]}{\mathrm{coe}^{r \rightsquigarrow \bullet}_{\hat{A}} a : (i : \mathbb{I}) \Rightarrow \mathbf{el}[\hat{A}(i) \mid i = r \to a]}$$

$$\frac{r, s : \mathbb{I} \qquad \hat{A} : \mathbb{I} \Rightarrow \mathbf{set} \qquad a : (i : \mathbb{I}, \alpha : \top[i = r \vee \partial(s)]) \Rightarrow \mathbf{el}[\hat{A}(i)]}{\mathrm{com}\langle s \rangle^{r \rightsquigarrow \bullet}_{\hat{A}} a : (i : \mathbb{I}) \Rightarrow \mathbf{el}[\hat{A}(i) \mid i = r \vee \partial(s) \to a(i)]}$$

Homogeneous composition is the special case of composition where the type code is constant:

$$\mathrm{hcom}\langle s \rangle^{r \rightsquigarrow \bullet}_{\hat{A}} a \stackrel{def}{=\!=} \mathrm{com}\langle s \rangle^{r \rightsquigarrow \bullet}_{\lambda\_.\hat{A}} a$$ ❦

Specification 4.28 (Regularity)**.** We will require the following regularity laws:

$$\mathrm{coe}^{r \rightsquigarrow r'}_{\lambda\_.\hat{A}} a = a$$

$$\mathrm{hcom}\langle s \rangle^{r \rightsquigarrow r'}_{\hat{A}} (\lambda\_\ \_.a) = a$$ ❦

Lemma 4.29. *As a consequence of boundary separation, the following compatibility law between homogeneous and heterogeneous equality holds:*

$$\mathrm{com}\langle s \rangle^{r \rightsquigarrow r'}_{\hat{A}} a = \mathrm{coe}^{r \rightsquigarrow r'}_{\hat{A}} (\mathrm{hcom}\langle s \rangle^{r \rightsquigarrow r'}_{\hat{A} r} (\lambda i, \alpha. \mathrm{coe}^{i \rightsquigarrow r}_{\hat{A}} a(i, \alpha)))$$

Specification 4.30 (Coercion at connectives)**.** In order to satisfy the *canonicity* property, our theory must further constrain the behavior of the coercion operation at each type.

$$\pi_1(\mathrm{coe}^{r \rightsquigarrow r'}_{\lambda i. \widehat{\mathrm{sg}}(\hat{A} i, \hat{B} i)} p) = \mathrm{coe}^{r \rightsquigarrow r'}_{\hat{A}} \pi_1(p)$$

$$\pi_2(\mathrm{coe}^{r \rightsquigarrow r'}_{\lambda i. \widehat{\mathrm{sg}}(\hat{A} i, \hat{B} i)} p) = \mathrm{coe}^{r \rightsquigarrow r'}_{\lambda i. \hat{B} i (\mathrm{coe}^{r \rightsquigarrow i}_{\hat{A}} \pi_1(p))} \pi_2(p)$$

$$(\mathrm{coe}^{r \rightsquigarrow r'}_{\lambda i. \widehat{\mathrm{pi}}(\hat{A} i, \hat{B} i)} p)(a) = \mathrm{coe}^{r \rightsquigarrow r'}_{\lambda i. \hat{B} i (\mathrm{coe}^{r' \rightsquigarrow i}_{\hat{A}} a)} p(\mathrm{coe}^{r' \rightsquigarrow r}_{\hat{A}} a)$$

$$(\mathrm{coe}^{r \rightsquigarrow r'}_{\lambda i. \widehat{\mathrm{path}}(\hat{A} i, a i)} p)(s) = \mathrm{com}\langle s \rangle^{r \rightsquigarrow r'}_{\hat{A} s} (\lambda\_. p(s))$$ ❦

Lemma 4.31 (Composition at connectives)**.** *The behavior of the homogeneous composition operations at each connective is completely determined by Specifications 4.28 and 4.30. In particular, we have:*

$$\pi_1(\mathrm{hcom}\langle s \rangle^{r \rightsquigarrow r'}_{\widehat{\mathrm{sg}}(\hat{A}, \hat{B})} p) = \mathrm{hcom}\langle s \rangle^{r \rightsquigarrow r'}_{\hat{A}} (\lambda i, \alpha. \pi_1(p(i, \alpha)))$$

$$\pi_2(\mathrm{hcom}\langle s \rangle^{r \rightsquigarrow r'}_{\widehat{\mathrm{sg}}(\hat{A}, \hat{B})} p) = \mathrm{com}\langle s \rangle^{r \rightsquigarrow r'}_{\lambda i. \hat{B}(\mathrm{hcom}\langle s \rangle^{r \rightsquigarrow i}_{\hat{A}} (\lambda i, \alpha. \pi_1(p(i, \alpha))))} (\lambda i, \alpha. \pi_2(p(i, \alpha)))$$

$$(\mathrm{hcom}\langle s \rangle^{r \rightsquigarrow r'}_{\widehat{\mathrm{pi}}(\hat{A}, \hat{B})} p)(a) = \mathrm{hcom}\langle s \rangle^{r \rightsquigarrow r'}_{\hat{B}(a)} (\lambda i, \alpha. p(i, \alpha, a))$$

$$(\mathrm{hcom}\langle s \rangle^{r \rightsquigarrow r'}_{\widehat{\mathrm{path}}(\hat{A}, a)} p)(s') = \mathrm{hcom}\langle s \rangle^{r \rightsquigarrow r'}_{\hat{A} s'} (\lambda i, \alpha. p(i, \alpha, s'))$$

$$\mathrm{hcom}\langle s \rangle^{r \rightsquigarrow r'}_{\widehat{\mathrm{bool}}} p = p(r, *)$$

Proof. By boundary separation, pivoting on $s : \mathbb{I}$. □

5. Cubical computability families

Definition 5.1. The category $\square$ of *Cartesian cubes* is the free category with finite products generated by a bi-pointed object, i.e. the Lawvere category of the theory of bi-pointed objects [Awo15]. We will write $\mathbf{Set}_{\square}$ for the category $\mathbf{Pr}(\square)$ of *Cartesian cubical sets.* ❦

Let $\mathcal{M}_{\mathcal{C}}$ be a model of XTT whose category of contexts is $\mathcal{C}$. Letting $\square$ be the category of Cartesian cubes, a finite product preserving functor $\square \xrightarrow{\square^\bullet} \mathcal{C}$ is determined by an *interval object* (i.e. a bi-pointed object) in $\mathcal{C}$, which we construct in Construction 5.2 below.

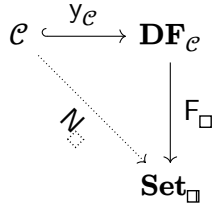
Construction 5.2 (An interval object in $\mathcal{C}$)**.** Because the Yoneda embedding reflects limits, the terminal discrete fibration $\diamond_{\mathcal{C}} : \mathbf{DF}_{\mathcal{C}}$ is represented by the terminal object of $\mathcal{C}$, which we may write $[\diamond_{\mathcal{C}}]$. We have required that the terminal map $(\mathbb{I} \longrightarrow \diamond)_{\mathcal{C}} : \mathbf{DF}_{\mathcal{C}}$ be a representable family; therefore, the discrete fibration $\mathbb{I}_{\mathcal{C}} : \mathbf{DF}_{\mathcal{C}}$ is represented by some object $[\mathbb{I}_{\mathcal{C}}] : \mathcal{C}$. Since the Yoneda embedding is fully faithful, the constants $(\diamond \xrightarrow{0,1} \mathbb{I})_{\mathcal{C}}$ are represented by two maps $[\diamond_{\mathcal{C}}] \xrightarrow{[0],[1]} [\mathbb{I}_{\mathcal{C}}] : \mathcal{C}$. ❦

From Construction 5.2 we obtain a suitable functor $\square \xrightarrow{\square^\bullet} \mathcal{C}$. By change of base, we obtain a reindexing functor $\mathbf{Pr}(\mathcal{C}) \xrightarrow{\square^*} \mathbf{Set}_{\square}$ which has both left and right adjoints by Kan extension. Composing with the equivalence $\mathbf{DF}_{\mathcal{C}} \simeq \mathbf{Pr}(\mathcal{C})$, we define a functor $\mathsf{F}_{\square}$ to glue along below:

$$\mathbf{DF}_{\mathcal{C}} \xrightarrow{\simeq} \mathbf{Pr}(\mathcal{C}) \xrightarrow{\square^*} \mathbf{Set}_{\square} \qquad \mathsf{F}_{\square} : \mathbf{DF}_{\mathcal{C}} \to \mathbf{Set}_{\square} \tag{5.3}$$

Thus the functor $\mathbf{DF}_{\mathcal{C}} \xrightarrow{\mathsf{F}_{\square}} \mathbf{Pr}(\square)$ is in fact an algebraic morphism of logoi, therefore a good candidate for type theoretic gluing [SA20].

Construction 5.4 (A cubical nerve)**.** By composing the change of base $\mathbf{DF}_{\mathcal{C}} \xrightarrow{\mathsf{F}_{\square}} \mathbf{Set}_{\square}$ with the Yoneda embedding, we obtain a *nerve* functor from the category of contexts of $\mathcal{M}_{\mathcal{C}}$ into cubical sets:



❦

In 2015, Awodey suggested the idea of gluing along a *cubical nerve* like Construction 5.4 to develop the metatheory of cubical type theory; to many researchers, it seemed as though Huber's operational proof of canonicity for cubical type theory [Hub18] could be reconstructed in a mathematical way. Since then, Awodey and Fiore have used this cubical gluing technique to study a version of intensional type theory with an interval in unpublished joint work; in 2019, the present authors applied cubical gluing to prove canonicity for an earlier version of XTT [SAG19].

Lemma 5.5. *The cubical nerve is* internally flat.

Proof. By Diaconescu's theorem [Bor10], we may check that the extension $\mathbf{DF}_{\mathcal{C}} \xrightarrow{\mathrm{Lan}_{\mathsf{y}_{\mathcal{C}}} \mathsf{N}_\square} \mathbf{Set}_\square$ is an algebraic morphism of logoi. But this Yoneda extension is just $\mathsf{F}_\square \simeq \square^*_\bullet$, which has both left and right adjoints. □

The classical gluing construction. Indeed, we may obtain a category of *glued contexts* by taking the comma category $\mathbf{Set}_\square \downarrow \mathsf{N}_\square$. One may think of a glued context as a "cubically computable context", with morphisms given by natural transformations of cubical sets which are tracked by substitutions from the model $\mathcal{M}_{\mathcal{C}}$. Then, the collections of types and elements (and the rest of a model of XTT) must be developed in the discrete fibrations/presheaves over the gluing category. This is the perspective pursued in previous work on gluing for strict type theory [SAG19, KHS19, CHS19, GKNB20]; in recent joint work, the first two authors of this paper have argued that it is considerably simpler to first glue over $\mathbf{DF}_{\mathcal{C}}$ rather than $\mathcal{C}$, and then restrict further to $\mathcal{C}$ by pulling back along $\mathcal{C} \xhookrightarrow{\mathsf{y}_{\mathcal{C}}} \mathbf{DF}_{\mathcal{C}}$ [SA20].

Following [SA20], we prefer to first glue along $\mathbf{DF}_{\mathcal{C}} \xrightarrow{\mathsf{F}_\square} \mathbf{Set}_\square$ rather than $\mathcal{C} \xrightarrow{\mathsf{N}_\square} \mathbf{Set}_\square$, because the comma category $\mathcal{G} = \mathbf{Set}_\square \downarrow \mathsf{F}_\square$ has more regular properties than $\mathcal{G}_{\mathsf{K}} = \mathbf{Set}_\square \downarrow \mathsf{N}_\square$, being a Grothendieck logos [AGV72]. The resulting two-step gluing process amounts to enlarging the collection of computability families to include families over arbitrary discrete fibrations $X : \mathbf{DF}_{\mathcal{C}}$, in addition to the familiar ones that lie directly over contexts $\Gamma : \mathcal{C}$. Computability families lying over a context $\Gamma : \mathcal{C}$ will be referred to as "compact"; the fundamental example of a computability family which is not compact is the computability family of computable types, lying over the non-representable discrete fibration $\mathbf{T}_{\mathcal{C}} : \mathbf{DF}_{\mathcal{C}}$.

The general computability families are connected a *model* of type theory (which must take place in discrete fibrations over *compact* computability families) via a nerve–realization adjunction. The nerve functor $\mathcal{G} \longrightarrow \mathbf{DF}_{\mathcal{G}_{\mathsf{K}}}$ is fully faithful and even locally cartesian closed, and may therefore be used to transform general computability families into constituents of a model of type theory over $\mathcal{G}_{\mathsf{K}}$ which are otherwise vastly more difficult to compute.

5.1. **General and compact computability families.** The picture painted at the end of the previous section is depicted in Diagram 5.6 below:

$$\begin{array}{ccccc}
\mathcal{G}_{\mathsf{K}} & \xhookrightarrow{\mathsf{K}} & \mathcal{G} & \longrightarrow & [\Delta^1, \mathbf{Set}_\square] \\
{\scriptstyle \mathsf{gl}_{\mathsf{K}}}\downarrow & \lrcorner & {\scriptstyle \mathsf{gl}}\downarrow & \lrcorner & \downarrow{\scriptstyle \partial_1} \\
\mathcal{C} & \xhookrightarrow{\mathsf{y}_{\mathcal{C}}} & \mathbf{DF}_{\mathcal{C}} & \xrightarrow{\mathsf{F}_\square} & \mathbf{Set}_\square \\
 & {\scriptstyle \mathsf{N}_\square}\searrow & & \nearrow & \\
 & & \mathbf{Set}_\square & &
\end{array} \tag{5.6}$$

We call $\mathcal{G}$ the category of (general) computability families, whereas the full subcategory $\mathcal{G}_{\mathsf{K}}$ is the category of *compact* computability families. This terminology is justified by Theorem 5.7 below.

Theorem 5.7 [SA20, Theorem 4.8]**.** *The subcategory inclusion* $\mathcal{G}_{\mathsf{K}} \xhookrightarrow{\mathsf{K}} \mathcal{G}$ *is* dense *in the sense that every object* $X : \mathcal{G}$ *is canonically the colimit of the following diagram of compact objects:*

$$\mathsf{K} \downarrow \{X\} \xrightarrow{\pi_{\mathsf{K}}} \mathcal{G}_{\mathsf{K}} \xhookrightarrow{\mathsf{K}} \mathcal{G} \tag{5.8}$$

We reproduce a variant of the proof given by Sterling and Angiuli [SA20].

Proof. Using the universality of colimits in a Grothendieck logos and the fact that $\mathsf{F}_{\square}$ is cocontinuous, we will show that X is the colimit of a particular canonical diagram in $\mathcal{G}$ which is final for Diagram 5.8. First, we use the dual Yoneda lemma (that $\mathcal{C}$ is dense in $\mathbf{DF}_{\mathcal{C}}$) to observe that $\mathsf{gl}(X)$ is the colimit of the following diagram:

$$\mathsf{y}_{\mathcal{C}} \downarrow \{\mathsf{gl}(X)\} \xrightarrow{\pi_{\mathcal{C}}} \mathcal{C} \xhookrightarrow{\mathsf{y}_{\mathcal{C}}} \mathbf{DF}_{\mathcal{C}} \tag{5.9}$$

Each leg $\mathsf{y}_{\mathcal{C}}(C_i) \xrightarrow{\alpha_i} \mathsf{gl}(X)$ of the colimiting cone for Diagram 5.9 induces a cartesian lift at X in the gluing fibration:

$$\begin{array}{ccc} \alpha_i^* X & \xrightarrow{\alpha_i^\dagger X} & X \\ \downarrow & & \downarrow \\ \mathsf{y}_{\mathcal{C}}(C_i) & \xrightarrow[\alpha_i]{} & \mathsf{gl}(X) \end{array} \tag{5.10}$$

We will see that the resulting cocone $\{\alpha_i^* X \xrightarrow{\alpha_i^\dagger X} X\}$ in $\mathcal{G}$ is universal for X. Because colimits in the comma category may be computed pointwise, we may reason as follows. Cartesian lifts in the gluing fibration are computed as pullbacks in $\mathbf{Set}_{\square}$; because colimits in the presheaf logos $\mathbf{Set}_{\square}$ are universal, it suffices to check that the cone $\{\mathsf{F}_{\square}(\mathsf{y}_{\mathcal{C}}(C_i)) \xrightarrow{\mathsf{F}_{\square}(\alpha_i)} \mathsf{F}_{\square}(\mathsf{gl}(X))\}$ is universal in $\mathbf{Set}_{\square}$. But $\mathbf{DF}_{\mathcal{C}} \xrightarrow{\mathsf{F}_{\square}} \mathbf{Set}_{\square}$ is cocontinuous, so it is enough that $\{\mathsf{y}_{\mathcal{C}}(C_i) \xrightarrow{\alpha_i} \mathsf{gl}(X)\}$ is universal. Finally, the universality of Cartesian lifts ensures that the collection $\{\alpha_i^* X\}$ is final for Diagram 5.8. □

Definition 5.11. An object $X : \mathcal{G}$ will be called *compact* when it lies in the essential image of K; equivalently, when $\mathsf{gl}(X) : \mathbf{DF}_{\mathcal{C}}$ is representable. ❦

We may impose the structure of a representable map category on $\mathcal{G}$, based on generalizing the notion of compactness from objects to families.

Definition 5.12. A family $Y \xrightarrow{p} X : \mathcal{G}$ is called *compact* when every change of base to a compact object $\mathsf{K}(\Gamma)$ is compact in the sense of the following diagram:

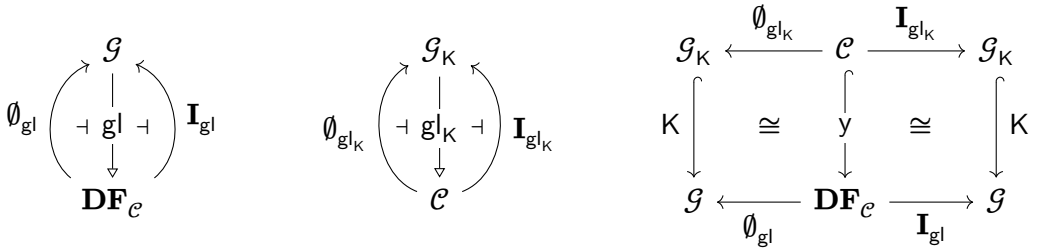
$$\begin{array}{ccc} \mathsf{K}(Y_x) & \longrightarrow & Y \\ \downarrow & \lrcorner & \downarrow p \\ \mathsf{K}(Z) & \xrightarrow[x]{} & X \end{array}$$

❦

Then, we say that the representable families in $\mathcal{G}$ are exactly the compact ones. It is simple to verify that this class of maps satisfies the axioms of a representable map

category: they are clearly closed under change of base, and since $\mathcal{G}$ is locally cartesian closed, pushforwards along representable maps always exist.

Lemma 5.13. *The gluing fibrations $\mathcal{G} \xrightarrow{\mathsf{gl}} \mathbf{DF}_{\mathcal{C}}$ and $\mathcal{G}_{\mathsf{K}} \xrightarrow{\mathsf{gl}_{\mathsf{K}}} \mathcal{C}$ have both left and right adjoints, and are therefore both continuous and cocontinuous; moreover, both adjoints are sections.*



Proof. From the perspective of the left and right adjoints as sections of the fibration, it is particularly simple to explain their behavior: the left adjoint takes a discrete fibration X to the *initial* object of the fiber category $\mathsf{gl}[X]$, and the right adjoint takes X to the *terminal* object of the fiber category $\mathsf{gl}[X]$. Considering that $\mathsf{gl}[X]$ is just the slice $\mathbf{Set}_{\square}/\mathsf{F}_{\square}(X)$, we may compute the families as follows:

$$\emptyset_{\mathsf{gl}}(X) = (\emptyset_{\mathbf{Set}_{\square}} \xrightarrow{!_{\mathsf{F}_{\square}(X)}} \mathsf{F}_{\square}(X))$$
$$\mathbf{I}_{\mathsf{gl}}(X) = (\mathsf{F}_{\square}(X) \xrightarrow{\mathbf{id}_{\mathsf{F}_{\square}(X)}} \mathsf{F}_{\square}(X))$$

In fact, this characterization already describes the left and right adjoints to gl_{K}, considering that the fibers $\mathsf{gl}_{\mathsf{K}}[C] = \mathbf{Set}_{\square}/\mathsf{N}_{\square}(C)$ are equivalent to $\mathsf{gl}[\mathsf{y}_{\mathcal{C}}(C)] = \mathbf{Set}_{\square}/\mathsf{F}_{\square}(\mathsf{y}_{\mathcal{C}}(C))$. □

Lemma 5.14. *The gluing fibration $\mathcal{G} \xrightarrow{\mathsf{gl}} \mathbf{DF}_{\mathcal{C}}$ is a representable map functor.*

Proof. gl is a logical morphism, preserving in particular finite limits and all pushforwards. Therefore, it remains to check that it takes representable maps to representable maps. We fix a compact family $Y \xrightarrow{p} X$ to check that the map $\mathsf{gl}(Y \xrightarrow{p} X)$ is representable. In this case, it will be simplest to use the Grothendieck–style characterization of representable maps in $\mathbf{DF}_{\mathcal{C}}$ in terms of change of base to a representable object. Fixing $C : \mathcal{C}$ and a generalized element $\mathsf{y}_{\mathcal{C}}(C) \xrightarrow{x} \mathsf{gl}(X)$, we must check that the fiber product $\mathsf{gl}(Y) \times_{\mathsf{gl}(X)} \mathsf{y}_{\mathcal{C}}(C)$ is representable:

$$\begin{array}{ccc} \mathsf{gl}(Y) \times_{\mathsf{gl}(X)} \mathsf{y}_{\mathcal{C}}(C) & \longrightarrow & \mathsf{gl}(Y) \\ \downarrow & \lrcorner & \downarrow{\scriptstyle \mathsf{gl}(p)} \\ \mathsf{y}_{\mathcal{C}}(C) & \xrightarrow[x]{} & \mathsf{gl}(X) \end{array} \tag{5.15}$$

By transposing along the adjunction $\emptyset_{\mathsf{gl}} \dashv \mathsf{gl}$, we have a map $\emptyset_{\mathsf{gl}}(\mathsf{y}_{\mathcal{C}}(C)) \xrightarrow{\tilde{x}} X : \mathcal{G}$; noting that $\emptyset_{\mathsf{gl}}(\mathsf{y}_{\mathcal{C}}(C)) \cong \mathsf{K}(\emptyset_{\mathsf{gl}_{\mathsf{K}}}(C))$, we therefore have the following cartesian square in $\mathcal{G}$

using the compactness of p:

$$\begin{array}{ccc} \mathsf{K}(Y_{\tilde{x}}) & \longrightarrow & Y \\ \downarrow & \lrcorner & \downarrow{\scriptstyle p} \\ \mathsf{K}(\emptyset_{\mathsf{gl}_{\mathsf{K}}}(C)) & \xrightarrow[\tilde{x}]{} & X \end{array} \tag{5.16}$$

The image of Diagram 5.16 under $\mathcal{G} \xrightarrow{\mathsf{gl}} \mathbf{DF}_{\mathcal{C}}$ has the same cospan as Diagram 5.15; but gl preserves pullbacks and $\mathsf{K}(Y_{\tilde{x}})$ must lie over a representable object, so we are finished. □

Lemma 5.17. *The gluing fibration $\mathcal{G} \xrightarrow{\mathsf{gl}} \mathbf{DF}_{\mathcal{C}}$ reflects representable maps.*

Proof. Fixing a family $Y \xrightarrow{p} X : \mathcal{G}$ such that $\mathsf{gl}(Y \xrightarrow{p} X)$ is representable, we must check that p is a compact family; in other words, fixing $\mathsf{K}(\Gamma) \xrightarrow{x} X$, we must check that the fiber product $Y \times_X \mathsf{K}(\Gamma)$ below is compact:

$$\begin{array}{ccc} Y \times_X \mathsf{K}(\Gamma) & \longrightarrow & Y \\ \downarrow & \lrcorner & \downarrow{\scriptstyle p} \\ \mathsf{K}(\Gamma) & \xrightarrow[x]{} & X \end{array} \tag{5.18}$$

It suffices to check that $\mathsf{gl}(Y \times_X \mathsf{K}(\Gamma))$ is representable; because $\mathsf{gl}(p)$ is representable, Diagram 5.18 lies over the square below:

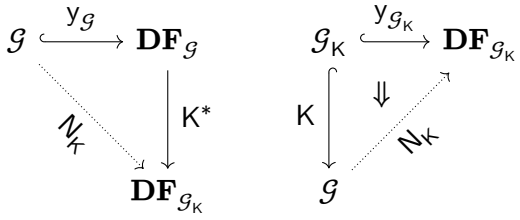
$$\begin{array}{ccc} \mathsf{y}(\mathsf{gl}(Y)_{\mathsf{gl}(x)}) & \longrightarrow & \mathsf{gl}(Y) \\ \downarrow & \lrcorner & \downarrow{\scriptstyle \mathsf{gl}(p)} \\ \mathsf{y}(\mathsf{gl}_{\mathsf{K}}(\Gamma)) & \xrightarrow[\mathsf{gl}(x)]{} & \mathsf{gl}(X) \end{array}$$

□

5.2. **Nerve and realization.** A computability model of XTT is given by a representable map functor $\mathbb{T} \xrightarrow{\mathcal{M}_{\mathcal{G}_{\mathsf{K}}}} \mathbf{DF}_{\mathcal{G}_{\mathsf{K}}}$; in particular, we must construct a natural model in $\mathbf{DF}_{\mathcal{G}_{\mathsf{K}}}$ which is closed under all the connectives of XTT and its universe of Bishop sets. These objects are, however, particularly difficult to construct from the perspective of discrete fibrations or presheaves; in previous work, it has accordingly been necessary to construct the constitutents of the computability model at the level of sets [SAG19, Coq19, KHS19], manually quantifying over computable contexts and computable substitutions.

In recent work, Sterling and Angiuli have shown that it is simpler to construct these objects *internally* to the logos $\mathcal{G}$ of general computability families, and then transfer them in a single motion to $\mathbf{DF}_{\mathcal{G}_{\mathsf{K}}}$. This is accomplished by means of a *nerve* functor $\mathcal{G} \xrightarrow{\mathsf{N}_{\mathsf{K}}} \mathbf{DF}_{\mathcal{G}_{\mathsf{K}}}$ which, by virtue of the density of $\mathcal{G}_{\mathsf{K}} \xhookrightarrow{\mathsf{K}} \mathcal{G}$ (Theorem 5.7), is not only fully faithful but also locally cartesian closed. Crucially, N_{K} will also turn out to be a representable map functor.

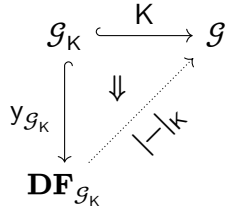
Construction 5.19 (Nerve)**.** Let $X : \mathcal{G}$ be a general computability family; we may define a discrete fibration $\mathsf{N}_\mathsf{K}(X) : \mathbf{DF}_{\mathcal{G}_\mathsf{K}}$ whose fiber at a compact computability family $\Gamma : \mathcal{G}_\mathsf{K}$ is the hom set $\mathcal{G}[\mathsf{K}(\Gamma), X]$. This assignment extends to a functor $\mathcal{G} \xrightarrow{\mathsf{N}_\mathsf{K}} \mathbf{DF}_{\mathcal{G}_\mathsf{K}}$, which may be viewed either as a restriction Yoneda embedding of $\mathcal{G}$, or a left Kan extension of the Yoneda embedding of $\mathcal{G}_\mathsf{K}$:



❦

Lemma 5.20 (Realization)**.** *The nerve functor has a left adjoint* $\mathbf{DF}_{\mathcal{G}_\mathsf{K}} \xrightarrow{|-|_\mathsf{K}} \mathcal{G}$*, the* realization *of a discrete fibration on compact computability families; consequently,* N_K *preserves small limits.*

Proof. The realization functor is obtained by left Kan extension:



□

The realization of a specific discrete fibration may be computed as a coend, using the general formula for pointwise Kan extensions. Letting $F : \mathbf{DF}_{\mathcal{G}_\mathsf{K}}$, we calculate:

$$\begin{aligned} |F|_\mathsf{K} &\cong \left(\mathrm{Lan}_{\mathsf{y}_{\mathcal{G}_\mathsf{K}}} \mathsf{K}\right)(F) \\ &\cong \textstyle\int^{\Gamma : \mathcal{G}_\mathsf{K}} \mathbf{DF}_{\mathcal{G}_\mathsf{K}}[\mathsf{y}_{\mathcal{G}_\mathsf{K}}(\Gamma), F] \cdot \mathsf{K}(\Gamma) \\ &\cong \textstyle\int^{\Gamma : \mathcal{G}_\mathsf{K}} F_\Gamma \cdot \mathsf{K}(\Gamma) \end{aligned}$$

Writing $\mathcal{G}_\mathsf{K}^{\mathsf{op}} \xrightarrow{F_\bullet} \mathbf{Set}$ for the presheaf corresponding to the discrete fibration F, we may package the computation of F's realization in terms of the tensor calculus of functors:

$$|F|_\mathsf{K} \cong F_\bullet \otimes_{\mathcal{G}_\mathsf{K}} \mathsf{K} \tag{5.21}$$

From Lemma 5.22 below, we can see that the nerve exhibits a true "Yoneda embedding" of general computability families into discrete fibrations.

Lemma 5.22. *The nerve functor* $\mathcal{G} \xrightarrow{\mathsf{N}_\mathsf{K}} \mathbf{DF}_{\mathcal{G}_\mathsf{K}}$ *is fully faithful.*

Proof. This is equivalent to the density of the inclusion K. □

Lemma 5.23 (Equal and opposite subcategories)**.** *The nerve–realization adjunction restricts to an equivalence (of categories) between the subcategories of generating objects (compact computability families and representable presheaves respectively).*

Proof. We first check that the nerve of any compact computability family $X \cong \mathsf{K}(\Gamma)$ is a representable discrete fibration. It suffices to compute in terms of the corresponding presheaves:

$$\begin{aligned}\mathsf{N}_{\mathsf{K}}(\mathsf{K}(\Gamma))_\bullet &\cong \mathcal{G}[\mathsf{K}(\bullet), \mathsf{K}(\Gamma)] \\ &\cong \mathcal{G}_{\mathsf{K}}[\bullet, \Gamma] \\ &\cong \mathsf{y}_{\mathcal{G}_{\mathsf{K}}}(\Gamma) \\ &\cong (\mathsf{y}_{\mathcal{G}_{\mathsf{K}}}(\Gamma))_\bullet\end{aligned}$$

Therefore, we have $\mathsf{N}_{\mathsf{K}}(\mathsf{K}(\Gamma)) \cong \mathsf{y}_{\mathcal{G}_{\mathsf{K}}}(\Gamma)$. Next, we check that the realization of any representable discrete fibration $F \cong \mathsf{y}_{\mathcal{G}_{\mathsf{K}}}(\Gamma)$ is compact; by the above, we have $|\mathsf{y}_{\mathcal{G}_{\mathsf{K}}}(\Gamma)|_{\mathsf{K}} \cong |\mathsf{N}_{\mathsf{K}}(\mathsf{K}(\Gamma))|_{\mathsf{K}}$; because the nerve is fully faithful (Lemma 5.22), the counit to the adjunction $|-|_{\mathsf{K}} \dashv \mathsf{N}_{\mathsf{K}}$ is an isomorphism, so we further have $|\mathsf{y}_{\mathcal{G}_{\mathsf{K}}}(\Gamma)|_{\mathsf{K}} \cong |\mathsf{N}_{\mathsf{K}}(\mathsf{K}(\Gamma))|_{\mathsf{K}} \cong \mathsf{K}(\Gamma)$.

We have shown that the nerve–realization adjunction restricts to a pair of functors between subcategories. It is easy to see that this is in fact an equivalence, since we have shown that $|\mathsf{N}_{\mathsf{K}}(\mathsf{K}(\Gamma))|_{\mathsf{K}} \cong \mathsf{K}(\Gamma)$ and $\mathsf{N}_{\mathsf{K}}(|\mathsf{y}_{\mathcal{G}_{\mathsf{K}}}(\Gamma)|_{\mathsf{K}}) \cong \mathsf{N}_{\mathsf{K}}(\mathsf{K}(\Gamma)) \cong \mathsf{y}_{\mathcal{G}_{\mathsf{K}}}(\Gamma)$. □

To establish the behavior of the nerve on pushforwards, it will be useful to choose a good dense subcategory of each slice $(\mathbf{DF}_{\mathcal{G}_{\mathsf{K}}})_{/F}$.

Remark 5.24. First we recall that in presheaves, each slice $\mathbf{Pr}(\mathcal{G}_{\mathsf{K}})_{/F_\bullet}$ may be reconstructed equivalently as the category of presheaves $\mathbf{Pr}(\int F_\bullet) = \mathbf{Pr}(F)$ on the total category of F; therefore, by the Yoneda lemma, each slice $(\mathbf{DF}_{\mathcal{G}_{\mathsf{K}}})_{/F}$ is densely generated by the functor $F \longrightarrow (\mathbf{DF}_{\mathcal{G}_{\mathsf{K}}})_{/F}$ which sends every element $x \in F_\Gamma$ to the corresponding map $\mathsf{y}_{\mathcal{G}_{\mathsf{K}}}(\Gamma) \xrightarrow{x} F$.

Furthermore, by Lemma 5.23 each generating object $\mathsf{y}_{\mathcal{G}_{\mathsf{K}}}(\Gamma) \longrightarrow F$ may be written equivalently as $\mathsf{N}_{\mathsf{K}}(\mathsf{K}(\Gamma)) \longrightarrow F$. If $F = \mathsf{N}_{\mathsf{K}}(X)$, we may observe that the slice $(\mathbf{DF}_{\mathcal{G}_{\mathsf{K}}})_{/F}$ is in fact densely generated by the comma category $\mathsf{K} \downarrow \{X\}$ under the functor which sends each $\mathsf{K}(\Gamma) \xrightarrow{x} X$ to $\mathsf{N}_{\mathsf{K}}(x) : (\mathbf{DF}_{\mathcal{G}_{\mathsf{K}}})_{/F}$, a direct consequence of the fully faithfulness of the nerve (Lemma 5.22). ❦

Lemma 5.25 [SA20, Lemma 4.2]**.** *The nerve functor preserves all pushforwards $f^* \dashv f_* : \mathcal{G}_{/X} \longrightarrow \mathcal{G}_{/Y}$ for $X \xrightarrow{f} Y : \mathcal{G}$.*

Proof. Letting $g : \mathcal{G}_{/X}$, we intend to check that $\mathsf{N}_{\mathsf{K}}(f_* g)$ is the pushforward $\mathsf{N}_{\mathsf{K}}(f)_* \mathsf{N}_{\mathsf{K}}(g)$. By Remark 5.24, it suffices to check the universal property at generators $\mathsf{N}_{\mathsf{K}}(\mathsf{K}(\Gamma)) \xrightarrow{\mathsf{N}_{\mathsf{K}}(x)} \mathsf{N}_{\mathsf{K}}(Y)$ of the slice $(\mathbf{DF}_{\mathcal{G}_{\mathsf{K}}})_{/\mathsf{N}_{\mathsf{K}}(Y)}$.

$$\begin{aligned}&[\mathsf{N}_{\mathsf{K}}(x), \mathsf{N}_{\mathsf{K}}(f)_* \mathsf{N}_{\mathsf{K}}(g)] \\ &\quad\cong [\mathsf{N}_{\mathsf{K}}(f)^* \mathsf{N}_{\mathsf{K}}(x), \mathsf{N}_{\mathsf{K}}(g)] \\ &\quad\cong [\mathsf{N}_{\mathsf{K}}(f^* x), \mathsf{N}_{\mathsf{K}}(g)] \\ &\quad\cong [f^* x, g] \\ &\quad\cong [x, f_* g] \\ &\quad\cong [\mathsf{N}_{\mathsf{K}}(x), \mathsf{N}_{\mathsf{K}}(f_* g)]\end{aligned}$$

□

Corollary 5.26. *The nerve functor is locally cartesian closed.*

Lemma 5.27. *The nerve functor preserves representable maps.*

Proof. We fix a representable map $Y \xrightarrow{f} X : \mathcal{G}$, to check that its nerve $\mathsf{N}_{\mathsf{K}}(Y) \xrightarrow{\mathsf{N}_{\mathsf{K}}(f)} \mathsf{N}_{\mathsf{K}}(X) : \mathbf{DF}_{\mathcal{G}_{\mathsf{K}}}$ is representable; it will be simplest to check this condition formulated in the classical Grothendieck–style, considering fiber products with representable objects:

$$\begin{array}{ccc} \mathsf{N}_{\mathsf{K}}(Y) \times_{\mathsf{N}_{\mathsf{K}}(X)} \mathsf{y}_{\mathcal{G}_{\mathsf{K}}}(\Gamma) & \to & \mathsf{N}_{\mathsf{K}}(Y) \\ \downarrow & \lrcorner & \downarrow{\scriptstyle \mathsf{N}_{\mathsf{K}}(f)} \\ \mathsf{y}_{\mathcal{G}_{\mathsf{K}}}(\Gamma) & \xrightarrow[x]{} & \mathsf{N}_{\mathsf{K}}(X) \end{array} \tag{5.28}$$

First of all, we may replace $\mathsf{y}_{\mathcal{G}_{\mathsf{K}}}(\Gamma)$ with the isomorphic $\mathsf{N}_{\mathsf{K}}(\mathsf{K}(\Gamma))$; since N_{K} is fully faithful and left exact, the entire square lies in the image of the nerve:

$$\begin{array}{ccc} \mathsf{N}_{\mathsf{K}}(Y \times_X \mathsf{K}(\Gamma)) & \longrightarrow & \mathsf{N}_{\mathsf{K}}(Y) \\ \downarrow & \lrcorner & \downarrow{\scriptstyle \mathsf{N}_{\mathsf{K}}(f)} \\ \mathsf{N}_{\mathsf{K}}(\mathsf{K}(\Gamma)) & \xrightarrow[\mathsf{N}_{\mathsf{K}}(\ulcorner x \urcorner)]{} & \mathsf{N}_{\mathsf{K}}(X) \end{array} \qquad \begin{array}{ccc} Y \times_X \mathsf{K}(\Gamma) & \longrightarrow & Y \\ \downarrow & \lrcorner & \downarrow{\scriptstyle f} \\ \mathsf{K}(\Gamma) & \xrightarrow[\ulcorner x \urcorner]{} & X \end{array} \tag{5.29}$$

Therefore, it suffices to check that the fiber product $Y \times_X \mathsf{K}(\Gamma)$ is taken by the nerve to a representable object; by Lemma 5.23, this follows from the compactness of $Y \times_X \mathsf{K}(\Gamma)$ by virtue of the representability of f. □

Corollary 5.30. *The nerve functor is a representable map functor.*

Proof. By Corollary 5.26 and Lemma 5.27. □

5.3. **Universes in the gluing fibration.** A type theoretic fibration category [Shu15] is a categorical notion for modeling weak (intensional) type theories analogous to Joyal's clans and tribes [Joy17]. While locally cartesian closed categories capture the style of type theory in which all morphisms express types (and thence have pullback and pushforward), type theoretic fibration categories analogously explain the more common case where only some morphisms (called "fibrations") correspond to types, and therefore only some pullbacks and pushforwards exist. In particular, any locally cartesian closed category (such as a logos) is a type theoretic fibration category, in which the class of fibrations includes all morphisms.

Uemura has generalized Shulman's type theoretic fibration categories to account for fiberwise structure in a fibration [Uem17]: a fibered type theoretic fibration category is a fibration between type theoretic fibration categories which enjoys certain closure and preservation properties. In particular, the gluing fibration $\mathcal{G} \xrightarrow{\mathsf{gl}} \mathbf{DF}_{\mathcal{C}}$ can be seen to be a fibered type theoretic fibration category, in which the fibrations in the base and the total category are *all* maps.

While type theoretic fibration categories are models of *weak* type theory rather than strict type theory, they provide a suitable environment in which to develop the semantics of strict type theory. Uemura has that $\mathcal{G} \xrightarrow{\mathsf{gl}} \mathbf{DF}_{\mathcal{C}}$ possesses a *fibered universe* which may be used as the basis for a coherence theorem which exhibits the structure of a strict model of type theory in the gluing fibration, i.e. a representable map functor $\mathbb{T} \longrightarrow \mathcal{G}$ which is a weak section of the gluing fibration.

Following Uemura [Uem17], we exhibit a right adjoint to cartesian lift in the gluing fibration.

Lemma 5.31 (Pushforward in the gluing fibration)**.** *Let* $X \xrightarrow{f} Y : \mathbf{DF}_{\mathcal{C}}$*; then we have a pushforward functor* $\mathrm{gl}[X] \xrightarrow{f_*} \mathrm{gl}[Y]$ *with the universal property* $f^* \dashv f_*$*.*

Proof. We have $\mathbf{I}_{\mathrm{gl}}(X \xrightarrow{f} Y)$ lying over $X \xrightarrow{f} Y$; by pushforward in the codomain fibration of $\mathcal{G}$ we have $\mathcal{G}_{/\mathbf{I}_{\mathrm{gl}}(X)} \xrightarrow{\mathbf{I}_{\mathrm{gl}}(f)_*} \mathcal{G}_{/\mathbf{I}_{\mathrm{gl}}(Y)}$. But each slice $\mathcal{G}_{/\mathbf{I}_{\mathrm{gl}}(Z)}$ is equivalent to the fiber $\mathrm{gl}[Z]$. The universal property follows from the fact that the cartesian lift f^* is, modulo the same equivalence, the pullback $\mathbf{I}_{\mathrm{gl}}(f^*)$. □

We will fix a sufficiently large universe of discrete fibrations $\widetilde{\mathrm{U}} \xrightarrow{\mathrm{u}} \mathrm{U} : \mathbf{DF}_{\mathcal{C}}$, obtained from the standard lifting of Grothendieck universes to Grothendieck logoi [HS97, Str05]. This universe is closed under dependent product, dependent sum, quotients, subobject comprehension, and even induction-recursion (assuming a strong background set theory). We will pick u to be large enough to classify at least $\left(\widetilde{\mathbf{T}} \xrightarrow{\tau} \mathbf{T}\right)_{\mathcal{C}}$.

Construction 5.32 (Universe of cubical sets)**.** A Hofmann–Streicher universe of cubical sets $\widetilde{\mathrm{V}} \xrightarrow{\mathrm{v}} \mathrm{V} : \mathbf{Set}_{\square}$ is considered "sufficiently large" if it regards the family $\mathsf{F}_{\square}\left(\widetilde{\mathrm{U}} \xrightarrow{\mathrm{u}} \mathrm{U}\right)$ as small. *We then will write* $\widetilde{\mathrm{V}} \xrightarrow{\mathrm{v}} \mathrm{V}$ *for the smallest sufficiently large universe.* ❦

Notation 5.33. The universe $\widetilde{\mathrm{V}} \xrightarrow{\mathrm{v}} \mathrm{V}$ may be viewed as a universe in the fiber $\mathrm{gl}[\diamond_{\mathcal{C}}]$. By change of base along the terminal maps $X \longrightarrow \diamond_{\mathcal{C}}$, we obtain universe objects $\widetilde{\mathrm{V}}_{[X]} \xrightarrow{\mathrm{v}_{[X]}} \mathrm{V}_{[X]} : \mathrm{gl}[X]$ in each fiber of the gluing fibration. ❦

We may see that each $\widetilde{\mathrm{V}}_{[X]} \xrightarrow{\mathrm{v}_{[X]}} \mathrm{V}_{[X]}$ is likewise a suitable universe object in the fiber $\mathrm{gl}[X]$, considering that each change of base $\mathrm{gl}[X] \longrightarrow \mathrm{gl}[Y]$ is a logical morphism of logoi.

Construction 5.34 (Universe of small computability families)**.** We define a computability family $\mathrm{G} : \mathrm{gl}[\mathrm{U}]$ by pushing forward $\mathrm{V}_{[\widetilde{\mathrm{U}}]}$ along $\widetilde{\mathrm{U}} \xrightarrow{\mathrm{u}} \mathrm{U}$:

$$\mathrm{G} \overset{def}{=\!=} \mathrm{u}_* \mathrm{V}_{[\widetilde{\mathrm{U}}]}$$

Writing $\mathrm{u}^*\mathrm{u}_* X \xrightarrow{\varepsilon_X} X$ for the counit of the adjunction $\mathrm{u}^* \dashv \mathrm{u}_*$, we may define $\widetilde{\mathrm{G}}$ by pullback in $\mathrm{gl}[\widetilde{\mathrm{U}}]$ as follows:

$$\begin{array}{ccc} \widetilde{\mathrm{G}} & \longrightarrow & \widetilde{\mathrm{V}}_{[\widetilde{\mathrm{U}}]} \\ \downarrow & \lrcorner & \downarrow{\scriptstyle \mathrm{v}_{[\widetilde{\mathrm{U}}]}} \\ \mathrm{u}^*\mathrm{G} & \xrightarrow[\varepsilon_{\mathrm{V}_{[\widetilde{\mathrm{U}}]}}]{} & \mathrm{V}_{[\widetilde{\mathrm{U}}]} \end{array}$$

Finally, we obtain the generic family $\widetilde{\mathrm{G}} \xrightarrow{\mathrm{g}} \mathrm{G}$ lying over $\widetilde{\mathrm{U}} \xrightarrow{\mathrm{u}} \mathrm{U}$ by composing with the cartesian lift:

$$\widetilde{\mathrm{G}} \xrightarrow{(vert)} \mathrm{u}^*\mathrm{G} \xrightarrow{\mathrm{u}^\dagger \mathrm{G}} \mathrm{G} \qquad \text{(with } \mathrm{g} \text{ : } \widetilde{\mathrm{G}} \to \mathrm{G}\text{)}$$

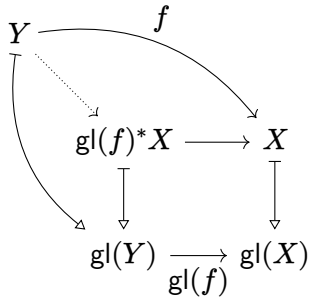
❦

Uemura has characterized the class of small maps for which $\widetilde{\mathrm{G}} \xrightarrow{\mathrm{g}} \mathrm{G}$ is generic.

Lemma 5.35 [Uem17, Lemma 4.4]**.** *A map $Y \xrightarrow{f} X : \mathcal{G}$ is small for* g *iff the following conditions hold:*

(1) $\mathsf{gl}(f)$ *is small for* u*.*

(2) *The following induced vertical map $Y \longrightarrow \mathsf{gl}(f)^* X$ is small for $\widetilde{\mathrm{V}}_{[\mathsf{gl}(Y)]} \xrightarrow{\mathrm{V}_{[\mathsf{gl}(Y)]}} \mathrm{V}_{[\mathsf{gl}(Y)]}$ when viewed as a map in the fiber $\mathsf{gl}[\mathsf{gl}(Y)]$:*



□

Using the characterization of Lemma 5.35, Uemura shows that $\widetilde{\mathrm{G}} \xrightarrow{\mathrm{g}} \mathrm{G}$ is closed under dependent product and dependent sum. In addition, we may see in our case that $\widetilde{\mathrm{G}} \xrightarrow{\mathrm{g}} \mathrm{G}$ is closed under extensional equality types as well as inductive-recursive definitions.

Lemma 5.36. *All monomorphisms $Y \xrightarrow{m} X : \mathcal{G}$ are small.*

Proof. To see that $\mathsf{gl}(m)$ is small for $\widetilde{\mathrm{U}} \xrightarrow{\mathrm{u}} \mathrm{U}$, note that gl preserves pullbacks and therefore monos; but monos are always small relative to Hofmann-Streicher universes. But the vertical map $Y \longrightarrow \mathsf{gl}(f)^*$ is also a monomorphism, so the same observation shows that this too is small. □

In other words, $\widetilde{\mathrm{G}} \xrightarrow{\mathrm{g}} \mathrm{G}$ is closed under subobject comprehension. We may even show that the *object* part of the subobject classifier $\Omega_{\mathcal{G}}$ is small, using the fact that Hofmann-Streicher universes are additionally impredicative [Str05].

Definition 5.37. Given a universe $\widetilde{\mathrm{U}} \xrightarrow{\mathrm{u}} \mathrm{U}$, we say that two codes $X \xrightarrow{A,\,B} \mathrm{U}$ are *isomorphic* when they have the same extensions, i.e. the pullbacks $A^*\mathrm{u}, B^*\mathrm{u}$ are isomorphic. In other words, the two codes are characteristic for the same family. ❦

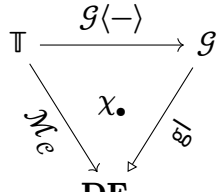
Lemma 5.38 (Realignment)**.** *Let $X \xrightarrow{\mathfrak{A}} \mathrm{G} : \mathcal{G}$ be a code for a small computability family in the following configuration:*

$$\begin{array}{ccc} X & \xrightarrow{\mathfrak{A}} & \mathrm{G} \\ \downarrow & & \downarrow \\ \mathsf{gl}(X) & \xrightarrow[A]{} & \mathrm{U} \end{array} \tag{5.39}$$

Let $\mathsf{gl}(X) \xrightarrow{B} \mathrm{U}$ be a code isomorphic to $\mathsf{gl}(X) \xrightarrow{A} \mathrm{U}$; then, we have a code $X \xrightarrow{\mathfrak{A}_B} \mathrm{G}$ lying strictly over B which is isomorphic to $\mathfrak{A}$. □

6. XTT in computability families

The essence of the canonicity argument for XTT lies in constructing a representable map functor $\mathbb{T} \xrightarrow{\mathcal{G}\langle -\rangle} \mathcal{G}$ together with a natural isomorphism $\mathcal{M}_{\mathcal{C}} \xrightarrow{\chi_\bullet} \mathsf{gl} \circ \mathcal{G}\langle -\rangle$ in the sense of Diagram 6.1 below:



(6.1)

The informal definition of $\mathbb{T}$ in Section 4 may be encoded as an equational presentation, i.e. a sequence of generating families, representable families, operations, and equations in Uemura's syntactic (meta-)logical framework [Uem19]; this free generation of $\mathbb{T}$ induces a (bi-categorical) universal property among representable map categories equipped with the structure of XTT. Therefore, to see that the canonical natural isomorphism $\chi_\bullet$ exists, it will suffice to choose suitable isomorphisms χ_I for just the *generating* objects $I : \mathbb{T}$; in all cases, the isomorphism χ_I will be canonical (or even the identity), because we will always define $\mathcal{G}\langle I \longrightarrow J\rangle$ to lie essentially over $(I \longrightarrow J)_{\mathcal{C}}$.

6.1. Glued cubical structure.

Construction 6.2 (The interval)**.** We must construct a computability family $\mathcal{G}\langle \mathbb{I}\rangle$ lying over $\mathbb{I}_{\mathcal{C}}$; at the level of cubical sets, we will use the "generic dimension" $\mathsf{y}_{\square}([1]) \xrightarrow{\mathcal{G}\langle \mathbb{I}\rangle} \mathsf{F}_{\square}(\mathbb{I}_{\mathcal{C}})$ determined essentially by the functorial action of $\square \xrightarrow{\square_\bullet} \mathcal{C}$. Fixing an element $[m] \xrightarrow{r} [1]$ of $\mathsf{y}_{\square}([1])$, we have by functoriality an element $\square_m \xrightarrow{\square_r} \square_1$ of $\mathsf{F}_{\square}(\mathbb{I}_{\mathcal{C}})$. We likewise have appropriate endpoints as follows:

$$\begin{array}{ccc} \mathbf{1}_{\mathbf{Set}_{\square}} & \xrightarrow{0,1} & \mathsf{y}_{\square}([1]) \\ \cong \downarrow & & \downarrow \mathcal{G}\langle \mathbb{I}\rangle \\ \mathsf{F}_{\square}(\diamond_{\mathcal{C}}) & \xrightarrow[\mathsf{F}_{\square}(0_{\mathcal{C}}),\mathsf{F}_{\square}(1_{\mathcal{C}})]{} & \mathsf{F}_{\square}(\mathbb{I}_{\mathcal{C}}) \end{array} \tag{6.3}$$

Diagram 6.3 can be seen to commute, considering the definition of $\square_\bullet$ as the finite product preserving functor corresponding to the interval-algebra in $\mathcal{C}$. ❦

Construction 6.4 (The face formula classifier)**.** We recall that the gluing category $\mathcal{G}$ is a logos [AGV72], and moreover, the gluing fibration $\mathcal{G} \xrightarrow{\mathsf{gl}} \mathbf{DF}_{\mathcal{C}}$ is a *logical morphism* and therefore preserves the subobject classifier and its first-order logic [Joh02]. Therefore, we may obtain a glued face formula classifier in a conceptual way.

Because $\Omega_{\mathbf{DF}_\mathcal{C}} : \mathbf{DF}_\mathcal{C}$ classifies monomorphisms, we obtain a unique cartesian classifying square in $\mathbf{DF}_\mathcal{C}$ for the generic face formula of $\mathcal{M}_\mathcal{C}$:

$$\begin{array}{ccc} \diamond_\mathcal{C} & \xrightarrow{\cong} & \mathbf{1}_{\mathbf{DF}_\mathcal{C}} \\ {\scriptstyle \top_\mathcal{C}}\downarrow & & \downarrow{\scriptstyle \mathsf{tr}_{\mathbf{DF}_\mathcal{C}}} \\ \mathbb{F}_\mathcal{C} & \xrightarrow[[\![-]\!]]{} & \Omega_{\mathbf{DF}_\mathcal{C}} \end{array} \tag{6.5}$$

Therefore, we may construct a suitable glued face formula classifier by taking a cartesian lift in the gluing fibration of the subobject classifier $\Omega_\mathcal{G}$ along $[\![-]\!]$ from Diagram 6.5, considering that $\Omega_\mathcal{G}$ lies over $\Omega_{\mathbf{DF}_\mathcal{C}}$:

$$\begin{array}{ccc} \mathcal{G}\langle \mathbb{F} \rangle & \xrightarrow{[\![-]\!]} & \Omega_\mathcal{G} \\ \downarrow & & \downarrow \\ \mathbb{F}_\mathcal{C} & \xrightarrow[[\![-]\!]]{} & \Omega_{\mathbf{DF}_\mathcal{C}} \end{array} \qquad \begin{array}{c} \mathcal{G} \\ \downarrow \\ \mathbf{DF}_\mathcal{C} \end{array} \tag{6.6}$$

The generic face formula $\mathcal{G}\langle \top \rangle$ is then obtained by pullback in $\mathcal{G}$:

$$\begin{array}{ccc} \mathcal{G}\langle \diamond \rangle & \dashrightarrow & \mathbf{1}_\mathcal{G} \\ {\scriptstyle \mathcal{G}\langle \top \rangle}\downarrow & & \downarrow{\scriptstyle \mathsf{tr}_\mathcal{G}} \\ \mathcal{G}\langle \mathbb{F} \rangle & \xrightarrow[[\![-]\!]]{} & \Omega_\mathcal{G} \end{array} \tag{6.7}$$

Because $\mathcal{G}|$ preserves finite limits and pullback cones are unique up to unique isomorphism, we see that $\mathcal{G}\langle \top \rangle$ lies over $\top_\mathcal{C}$ as required. ❦

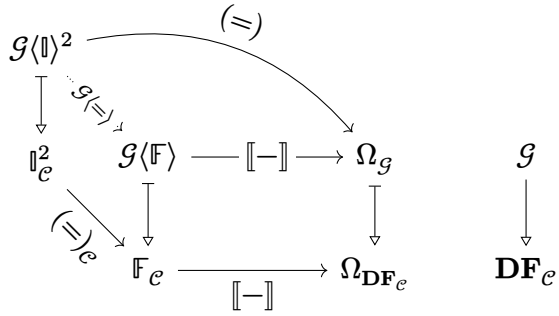
We may prove a Beck-Chevalley lemma for dimension equality in $\mathbf{DF}_\mathcal{C}$:

Lemma 6.8 (Beck-Chevalley)**.** *The following diagram commutes in* $\mathbf{DF}_\mathcal{C}$*.*

$$\begin{array}{ccc} \mathbb{I}^2_\mathcal{C} & & \\ {\scriptstyle (=)_\mathcal{C}}\downarrow & \swarrow_{\equiv} & \\ \mathbb{F}_\mathcal{C} & \xrightarrow[[\![-]\!]]{} & \Omega_{\mathbf{DF}_\mathcal{C}} \end{array}$$

Proof. Recalling the diagram from Specification 4.3, we observe that both maps are characteristic of the same subobject, and thence equal. □

Construction 6.9 (Glued dimension equality)**.** Dimension equality is lifted from $\mathcal{M}_{\mathcal{C}}$ to the gluing category by the universal property of the cartesian lift below, using the Beck-Chevalley triangle of Lemma 6.8 and the fact that gl is a logical morphism.



(6.10)

We must check that the square below from Specification 4.3 is cartesian:

$$\begin{array}{ccc} \mathcal{G}\langle\mathbb{I}\rangle & \longrightarrow & \mathcal{G}\langle\diamond\rangle \\ {\scriptstyle\delta}\downarrow & & \downarrow{\scriptstyle\mathcal{G}\langle\top\rangle} \\ \mathcal{G}\langle\mathbb{I}\rangle^2 & \xrightarrow[\mathcal{G}\langle=\rangle]{} & \mathcal{G}\langle\mathbb{F}\rangle \end{array} \tag{6.11}$$

By the pullback lemma, it would suffice to check that the outer square below is cartesian.

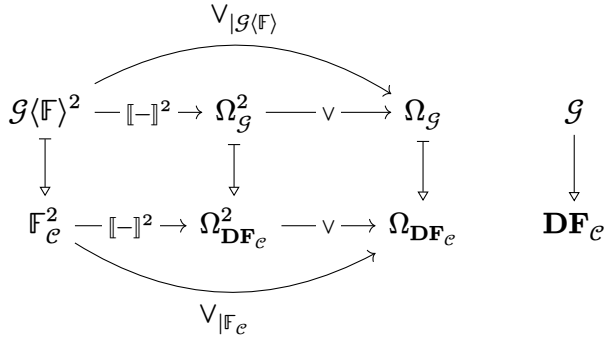
$$\begin{array}{ccccc} \mathcal{G}\langle\mathbb{I}\rangle & \longrightarrow & \mathcal{G}\langle\diamond\rangle & \longrightarrow & \mathbf{1}_{\mathcal{G}} \\ {\scriptstyle\delta}\downarrow & & \downarrow & & \downarrow{\scriptstyle\mathsf{tr}_{\mathcal{G}}} \\ \mathcal{G}\langle\mathbb{I}\rangle^2 & \xrightarrow[\mathcal{G}\langle=\rangle]{} & \mathcal{G}\langle\mathbb{F}\rangle & \xrightarrow[[\![-]\!]]{} & \Omega_{\mathcal{G}} \end{array} \tag{6.12}$$

Using the upstairs triangle of Diagram 6.10, it suffices to observe that the following classification square is cartesian:

$$\begin{array}{ccc} \mathcal{G}\langle\mathbb{I}\rangle & \longrightarrow & \mathbf{1}_{\mathcal{G}} \\ {\scriptstyle\delta}\downarrow & & \downarrow{\scriptstyle\mathsf{tr}_{\mathcal{G}}} \\ \mathcal{G}\langle\mathbb{I}\rangle^2 & \xrightarrow[(=)]{} & \Omega_{\mathcal{G}} \end{array}$$

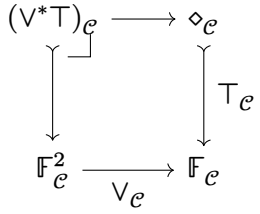
We do not expect a Beck-Chevalley lemma for disjunction analogous to Lemma 6.8, since $\phi \vee \psi$ is not (and cannot be) the "true" disjunction of $\mathbf{DF}_{\mathcal{C}}$: instead, we imposed orthogonality conditions in Specification 4.6 to ensure that certain judgments of XTT (typehood, typing, and formula satisfaction) treat $\phi \vee \psi$ as if it were a disjunction.

Construction 6.13 (Glued disjunction)**.** We may test a pair of glued face conditions for truth of disjunction as follows:



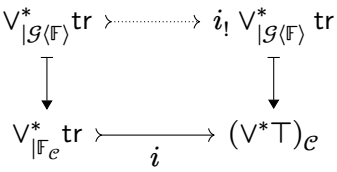
(6.14)

Unfortunately, the subobject $\bigvee^*_{|\mathbb{F}_{\mathcal{C}}}\mathsf{tr} \overset{def}{=\!=} \{\phi, \psi \mid \top_{\mathcal{C}}[\phi] \vee \top_{\mathcal{C}}[\psi]\}$ corresponding to the downstairs map of Diagram 6.14 is *not* classified by $\mathbb{F}_{\mathcal{C}}$! This is because such a subobject must have representable fibers, and but the real disjunction is a colimit in $\mathbf{DF}_{\mathcal{C}}$ and therefore not representable. We need something that lies instead over the following pullback:



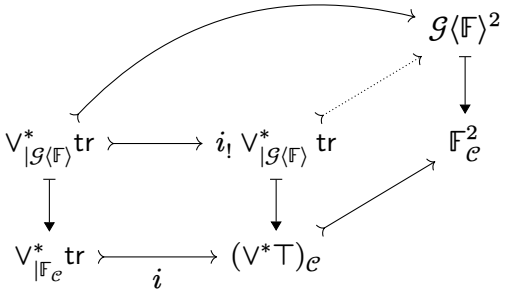
(6.15)

Because the "ideal" disjunction $\bigvee^*_{|\mathbb{F}_{\mathcal{C}}}\mathsf{tr}$ is more universal than the disjunction of $\mathbb{F}_{\mathcal{C}}$, we obtain a unique map $\bigvee^*_{|\mathbb{F}_{\mathcal{C}}}\mathsf{tr} \overset{i}{\rightarrowtail} (\vee^*\top)_{\mathcal{C}}$; taking an opcartesian lift, we may shift $\bigvee^*_{|\mathcal{G}\langle\mathbb{F}\rangle}\mathsf{tr}$ (the subobject corresponding to the upstairs map of Diagram 6.14) to lie over $(\vee^*\top)_{\mathcal{C}}$:



(6.16)

This lift can be seen to be a subobject of $\mathcal{G}\langle\mathbb{F}\rangle^2$ using the universal property of the opcartesian lift:



(6.17)

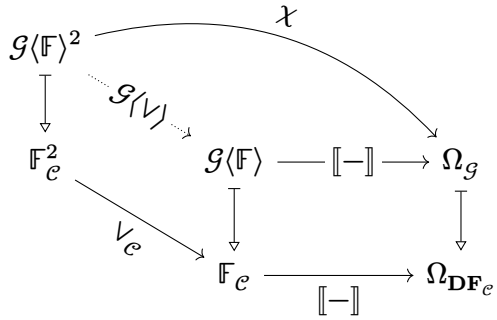
Consider the characteristic map of the dotted monomorphism from Diagram 6.17:

$$\begin{array}{ccc} i_! \vee^*_{|\mathcal{G}\langle\mathbb{F}\rangle|}\mathsf{tr} & \longrightarrow & \mathbf{1}_{\mathcal{G}} \\ \downarrow & & \downarrow \\ \mathcal{G}\langle\mathbb{F}\rangle^2 & \xrightarrow[\chi]{} & \Omega_{\mathcal{G}} \end{array} \tag{6.18}$$

Because gl is a logical functor and $\Omega_{\mathbf{DF}_{\mathcal{C}}}$ classifies subobjects strictly, Diagram 6.18 must lie over the following square:

$$\begin{array}{ccccc} (\vee^*\top)_{\mathcal{C}} & & \longrightarrow & & \mathbf{1}_{\mathbf{DF}_{\mathcal{C}}} \\ \downarrow & & & & \downarrow \\ \mathbb{F}^2_{\mathcal{C}} & \xrightarrow[\vee_{\mathcal{C}}]{} & \mathbb{F}_{\mathcal{C}} & \xrightarrow[[\![-]\!]]{} & \Omega_{\mathbf{DF}_{\mathcal{C}}} \end{array} \tag{6.19}$$

Therefore, we may use the universal property of the cartesian lift to obtain a code for disjunction of glued face conditions:



❦

Lemma 6.20 (Disjunction elimination / truth)**.** *The glued disjunction satisfies the internal orthogonality condition with respect to $\mathcal{G}\langle\top\rangle$ written in Specification 4.6.*

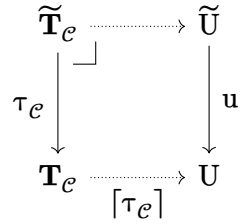
Proof. Fixing $X \xrightarrow{(\phi,\psi)} \mathcal{G}\langle\mathbb{F}\rangle^2$, we must find a unique lift for the following square, lying over the corresponding unique lift in $\mathbf{DF}_{\mathcal{C}}$:

$$\begin{array}{ccc} (\phi,\psi)^* \vee^*_{|\mathcal{G}\langle\mathbb{F}\rangle|}\mathsf{tr} & \longrightarrow & \mathcal{G}\langle\diamond\rangle \\ \downarrow & \nearrow & \downarrow \\ (\phi,\psi)^*\mathcal{G}\langle\vee^*\top\rangle & \longrightarrow & \mathcal{G}\langle\mathbb{F}\rangle \end{array} \tag{6.21}$$

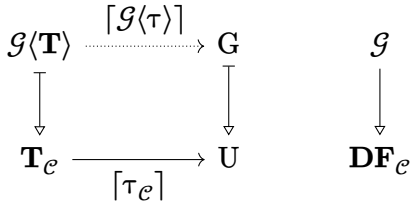
Because the lift in $\mathbf{DF}_{\mathcal{C}}$ is assumed to exist and is unique, it suffices to find a unique lift for the image of Diagram 6.21 under $\mathcal{G} \xrightarrow{\mathsf{E}_\bullet} \mathbf{Set}_{\square}$; but $\mathcal{G}\langle\vee^*\top\rangle$ is an opcartesian lift of $\vee^*_{|\mathcal{G}\langle\mathbb{F}\rangle|}\mathsf{tr}$, so the left-hand map becomes an identity in $\mathbf{Set}_{\square}$. □

6.2. **Glued type structure.** We will now show the sense in which the semantic constructions of Uemura summarized above suffice to develop the type structure of XTT in the gluing fibration.

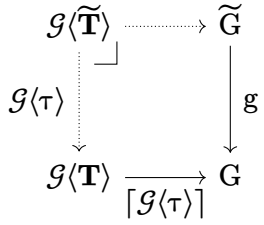
Construction 6.22 (Universe of glued types)**.** We will define a computability family $\mathcal{G}\langle \mathbf{T} \rangle : \mathcal{G}$ lying over $\mathbf{T}_{\mathcal{C}}$ in the gluing fibration $\mathcal{G} \xrightarrow{\mathsf{gl}} \mathbf{DF}_{\mathcal{C}}$. Because we have assumed $(\widetilde{\mathbf{T}} \xrightarrow{\tau} \mathbf{T})_{\mathcal{C}}$ is small for $\widetilde{\mathrm{U}} \xrightarrow{\mathrm{u}} \mathrm{U}$, we have a characteristic map:



We therefore obtain the base of a universe by cartesian lift:



Then, the rest of the universe $\mathcal{G}\langle \widetilde{\mathbf{T}} \xrightarrow{\tau} \mathbf{T} \rangle$ is obtained by pullback:

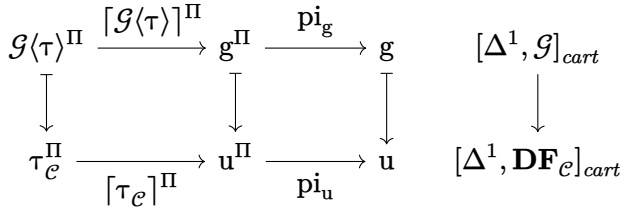


❦

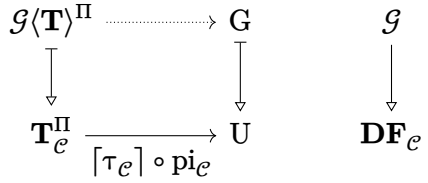
Lemma 6.23 (Disjunction elimination / elements)**.** *The glued disjunction satisfies the internal orthogonality condition with respect to $\mathcal{G}\langle \tau \rangle$ written in Specification 4.6.*

Proof. The proof is identical to that of Lemma 6.20. □

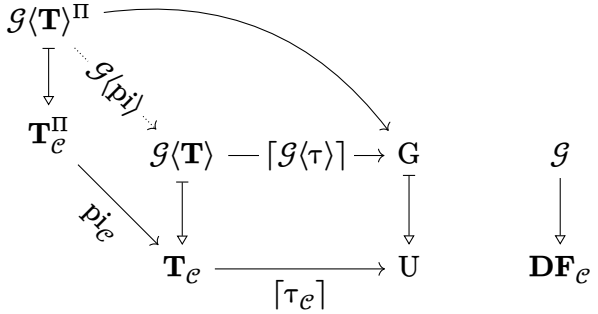
Construction 6.24 (Closure under dependent product)**.** We must show that $\mathcal{G}\langle \tau \rangle$ has a code for dependent products lying over the corresponding algebra $\tau^{\Pi}_{\mathcal{C}} \longrightarrow \tau_{\mathcal{C}}$. First of all, we have a potential code in $\mathcal{G}$ for the dependent product of $\mathcal{G}\langle \tau \rangle$-families in G, defined using functoriality and the closure of $\widetilde{\mathrm{G}} \xrightarrow{\mathrm{g}} \mathrm{G}$ under dependent products, and the fact that gl preserves dependent products:



By realignment (Lemma 5.38), using the fact $\mathrm{pi}_{\mathrm{U}} \circ \lceil \tau_{\mathcal{C}} \rceil^{\Pi}$ and $\lceil \tau_{\mathcal{C}} \rceil \circ \mathrm{pi}_{\mathcal{C}}$ are (different) characteristic maps for the same family, we obtain a new code in G for the same family in the following configuration:



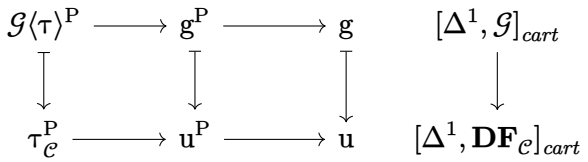
Therefore, we are in a position to define the type code using the universal property of the cartesian lift:



We have got the downstairs map aligned properly; to complete the algebra $\tau_{\mathcal{C}}^{\Pi} \longrightarrow \tau_{\mathcal{C}}$ with a properly aligned upstairs map, we may use the universal property of the pullback and the fact that gl is lex. ❦

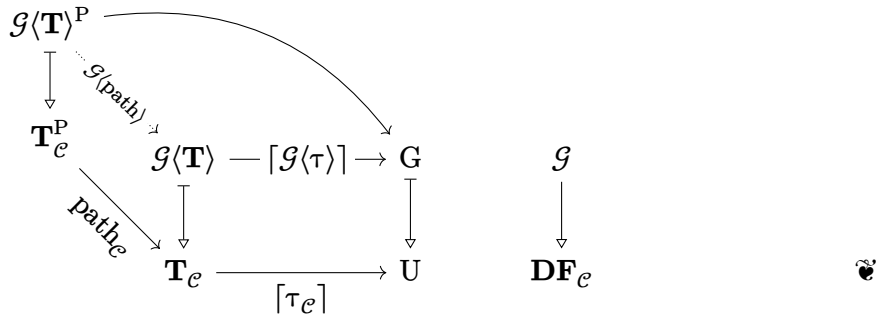
The closure under dependent sum works identically. We will, however, illustrate the closure under path types.

Construction 6.25 (Closure under path types)**.** First of all, the universe G is closed under path types because path types may be constructed (up to isomorphism) using the interval, dependent products, and subobject comprehension (all of which are small). Therefore, we have a cartesian map $\mathrm{g}^{\mathrm{P}} \longrightarrow \mathrm{g} : [\Delta^1, \mathcal{G}]_{cart}$. Using functoriality of $\bullet^{\mathrm{P}}$ and the fact that path types are preserved by gl, we have:

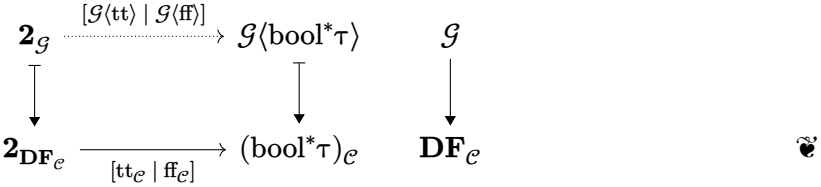


By Lemma 5.38, we may realign the upstairs map to lie over $\mathrm{path}_{\mathcal{C}} \circ \lceil \tau_{\mathcal{C}} \rceil$; therefore, we obtain a code for the glued path type lying over the original code by the universal property

of the cartesian lift:

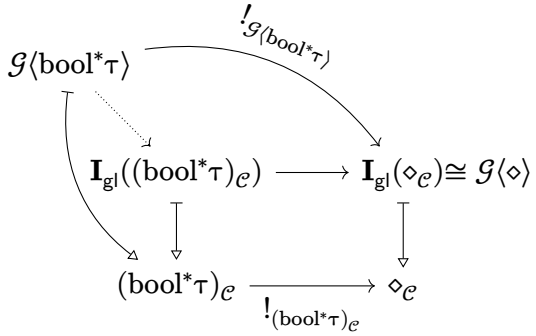


Construction 6.26. We may form a computability family over the booleans by *opcartesian* lift, using the fact that the gluing (op)fibration preserves colimits:



Lemma 6.27. *The computability family $\mathcal{G}\langle\mathrm{bool}^*\tau\rangle$ from Construction 6.26 is small.*

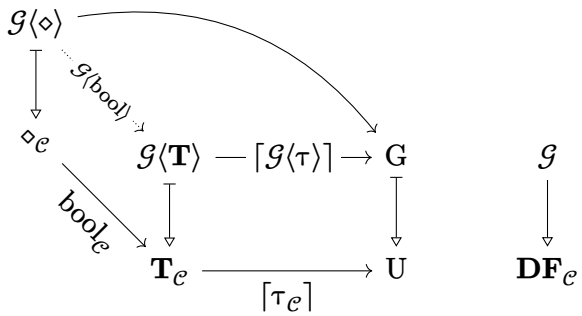
Proof. By the characterization theorem, it suffices to check that it lies over a small object (obvious), and that vertical map it induces is small in $\mathrm{gl}[(\mathrm{bool}^*\tau)_{\mathcal{C}}]$. To see that this is the case, we compute this vertical map as follows:



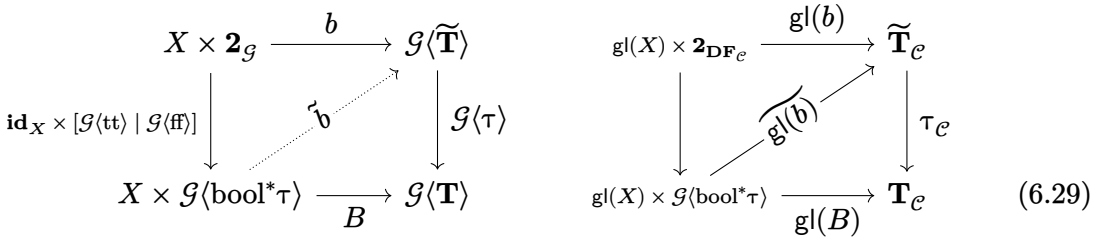
The vertical map above can be seen to be small using the fact that $\mathbf{2}_{\mathbf{Set}_{\square}} \longrightarrow \mathsf{F}_{\square}((\mathrm{bool}^*\tau)_{\mathcal{C}})$ is small. □

Construction 6.28 (Booleans)**.** By Lemma 6.27, there exists some characteristic map $\mathcal{G}\langle\diamond\rangle \xrightarrow{\lceil\mathrm{bool}^*\tau\rceil} \mathrm{G}$ for $\mathcal{G}\langle\mathrm{bool}^*\tau\rangle$ lying over a characteristic map for the object $(\mathrm{bool}^*\tau)_{\mathcal{C}}$. Therefore, again using the realignment lemma, we may define a suitable code for the booleans

using the universal property of the cartesian lift:

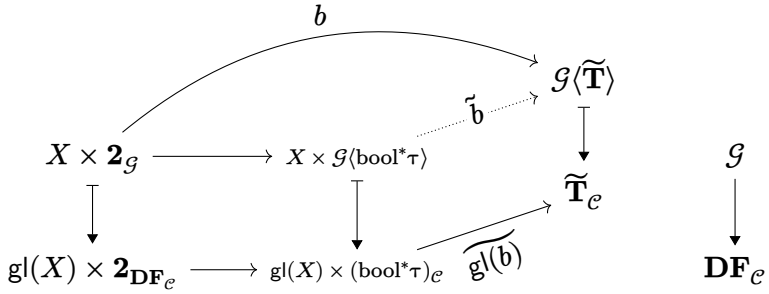


To define the elimination form, we must exhibit a choice of lifts of the following form natural in $X : \mathcal{G}$, lying over the corresponding lifts that we have fixed in $\mathbf{DF}_{\mathcal{C}}$:



(6.29)

For each $X : \mathcal{G}$, the existence of such a lift is guaranteed by the universal property of the induced opcartesian map $X \times \mathbf{2}_{\mathcal{G}} \longrightarrow X \times \mathcal{G}\langle\mathrm{bool}^*\tau\rangle$:



To see that the choice of $\tilde{b}$ is natural in X, we will observe the stronger property that it is the *unique* lift lying over $\widetilde{\mathsf{gl}(b)}$. Computing the opcartesian lift explicitly, we see that $\mathsf{E}_{\mathcal{G}\langle\mathrm{bool}^*\tau\rangle} = \mathsf{E}_{\mathbf{2}_{\mathcal{G}}}$; moreover $\mathsf{E}_\bullet$ preserves colimits because $\mathsf{F}_\square$ is left exact [Tay99], so in fact $\mathsf{E}_{\mathcal{G}\langle\mathrm{bool}^*\tau\rangle} = \mathbf{2}_{\mathbf{Set}_\square}$. Therefore, the (non-unique) lifting situation of Diagram 6.29 becomes a unique lifting situation in cubical sets. ❦

6.3. **Universe of Bishop sets.**

Construction 6.30 (Glued universe à la Tarski)**.** By induction-recursion, we may define a universe $\mathrm{U}_{\mathrm{IR}} : \mathcal{G}$ simultaneously with a decoding function $\mathrm{U}_{\mathrm{IR}} \xrightarrow{[-]_{\mathrm{IR}}} \mathcal{G}\langle\mathbf{T}\rangle$ closed under dependent product, dependent sum, path, and boolean.

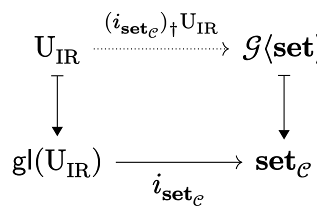
$$\frac{}{\texttt{bool} : \mathrm{U}_{\mathrm{IR}} \qquad [\texttt{bool}]_{\mathrm{IR}} = \mathcal{G}\langle \mathrm{bool} \rangle}$$

$$\frac{A : \mathrm{U}_{\mathrm{IR}} \qquad B : \mathcal{G}\langle \top \rangle[[A]_{\mathrm{IR}}] \Rightarrow \mathrm{U}_{\mathrm{IR}}}{\texttt{pi}(A, B) : \mathrm{U}_{\mathrm{IR}} \qquad [\texttt{pi}(A, B)]_{\mathrm{IR}} = \mathcal{G}\langle \mathrm{pi} \rangle([A]_{\mathrm{IR}}, [-]_{\mathrm{IR}} \circ B)}$$

$$\frac{A : \mathrm{U}_{\mathrm{IR}} \qquad B : \mathcal{G}\langle \top \rangle[[A]_{\mathrm{IR}}] \Rightarrow \mathrm{U}_{\mathrm{IR}}}{\texttt{sg}(A, B) : \mathrm{U}_{\mathrm{IR}} \qquad [\texttt{sg}(A, B)]_{\mathrm{IR}} = \mathcal{G}\langle \mathrm{sg} \rangle([A]_{\mathrm{IR}}, [-]_{\mathrm{IR}} \circ B)}$$

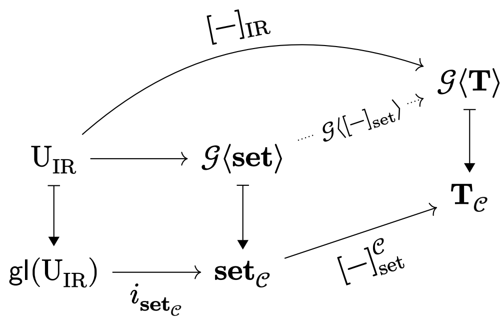
$$\frac{A : \mathcal{G}\langle \mathbb{I} \rangle \Rightarrow \mathrm{U}_{\mathrm{IR}} \qquad a : (i : \mathcal{G}\langle \mathbb{I} \rangle, \_ : \partial(i)) \Rightarrow \mathcal{G}\langle \top \rangle[[A(i)]_{\mathrm{IR}}]}{\texttt{path}(A, a) : \mathrm{U}_{\mathrm{IR}} \qquad [\texttt{path}(A, a)]_{\mathrm{IR}} = \mathcal{G}\langle \mathrm{path} \rangle([-]_{\mathrm{IR}} \circ A, a)}$$

U_{IR} lies *not* over the type-theoretic universe à la Tarski $\mathbf{set}_{\mathcal{C}} : \mathbf{DF}_{\mathcal{C}}$, but rather over a genuine inductive-recursive universe in $\mathbf{DF}_{\mathcal{C}}$. Because this $\mathsf{gl}(\mathrm{U}_{\mathrm{IR}})$ is the *least* universe closed under the mentioned connectives, we obtain a universal map $\mathsf{gl}(\mathrm{U}_{\mathrm{IR}}) \xrightarrow{i_{\mathbf{set}_{\mathcal{C}}}} \mathbf{set}_{\mathcal{C}}$ which automatically commutes with all connectives. We may therefore shift U_{IR} to lie over $\mathbf{set}_{\mathcal{C}}$ by opcartesian lift against this universal map:



(6.31)

A decoding map $\mathcal{G}\langle \mathbf{set} \rangle \xrightarrow{\mathcal{G}\langle [-]_{\mathrm{set}} \rangle} \mathcal{G}\langle \mathbf{T} \rangle$ is inherited using the universal property of the opcartesian lift:

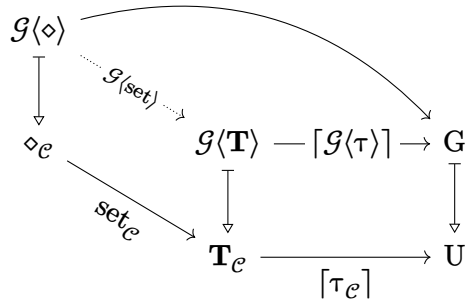


(6.32)

The map $[-]_{\mathrm{IR}}$ can be seen to lie strictly over the downstairs composite in Diagram 6.32 using the uniqueness of maps out of inductive-recursive universes. Moreover, the object $\mathcal{G}\langle \mathbf{set} \rangle$ is small, so we have a characteristic map $\mathcal{G}\langle \diamond \rangle \xrightarrow{\ulcorner \mathcal{G}\langle \mathbf{set} \rangle \urcorner} \mathrm{G}$.

The constructions above may be used as the basis for a universe à la Tarski $\mathcal{G}\langle \mathrm{set} \rangle : \mathcal{G}\langle \mathbf{T} \rangle$ lying over the type $\mathrm{set}_{\mathcal{C}} : \mathbf{T}_{\mathcal{C}}$. By Lemma 5.38, we may realign the characteristic map $\mathcal{G}\langle \diamond \rangle \xrightarrow{\ulcorner \mathcal{G}\langle \mathbf{set} \rangle \urcorner} \mathrm{G}$ to lie over the composite $\diamond_{\mathcal{C}} \xrightarrow{\ulcorner \top_{\mathcal{C}} \urcorner \circ \mathrm{set}_{\mathcal{C}}} \mathrm{U}$; in this way, we obtain an appropriate code for the universe à la Tarski by means of the universal property of the

cartesian lift below:



❦

We have *not* shown that the universe à la Tarski $\mathcal{G}\langle\mathbf{set}\rangle$ is closed under the appropriate connectives — we only know that U_{IR} is closed under those connectives. Prior to demonstrating this, however, we must record a few facts about the behavior of $\mathsf{F}_\square$ on pushfowards.

Lemma 6.33 [SA20, Lemma 3.5]**.** *Let $\mathcal{X} \xrightarrow{F} \mathcal{E}$ be any left exact functor, and let $f_*g : \mathcal{X}_{/Z}$ be the pushforward of $X \xrightarrow{g} Y$ along $Y \xrightarrow{f} Z$. Then we have a canonical (not usually invertible) comparison map $F(f_*g) \dashrightarrow F(f)_*F(g)$.* □

Corollary 6.34. *We a canonical comparison map commuting the generic dependent product family past $\mathsf{F}_\square \circ \mathcal{M}_{\mathcal{C}}$ in the following sense:*

$$\mathsf{F}_\square(\tau_{\mathcal{C}}^{\Pi}) \xrightarrow{(\!|-|\!)} (\mathsf{F}_\square(\tau_{\mathcal{C}}))^{\Pi}$$

Proof. By Lemma 6.33, using the fact that $\mathsf{F}_\square$ is left exact. □

Construction 6.35 (Constructors for the universe à la Tarski)**.** We will now show that $\mathcal{G}\langle\mathbf{set}\rangle$ is closed under the necessary connectives; we consider only the case of dependent products, since the remaining constructors work identically.

Recall that we must construct a morphism $\mathcal{G}\langle\mathbf{set}^{\Pi}\rangle \xrightarrow{\mathcal{G}\langle\widehat{\mathrm{pi}}\rangle} \mathcal{G}\langle\mathbf{set}\rangle$ that lies over $\mathbf{set}_{\mathcal{C}}^{\Pi} \xrightarrow{\widehat{\mathrm{pi}}_{\mathcal{C}}} \mathbf{set}_{\mathcal{C}}$. Unfolding this situation, we wish to construct the following dotted map in $\mathbf{Set}_\square$:

$$\begin{array}{ccc}
\mathsf{E}_{\mathcal{G}\langle\mathbf{set}\rangle^{\Pi}} & \overset{\mathsf{E}_{\mathcal{G}\langle\widehat{\mathrm{pi}}\rangle}}{\dashrightarrow} & \mathsf{E}_{\mathcal{G}\langle\mathbf{set}\rangle} \\
{\scriptstyle \mathcal{G}\langle\mathbf{set}\rangle^{\Pi}}\downarrow & & \downarrow{\scriptstyle \mathcal{G}\langle\mathbf{set}\rangle} \\
\mathsf{F}_\square(\mathbf{set}_{\mathcal{C}}^{\Pi}) & \xrightarrow[\mathsf{F}_\square(\widehat{\mathrm{pi}}_{\mathcal{C}})]{} & \mathsf{F}_\square(\mathbf{set}_{\mathcal{C}})
\end{array}$$

We will define this map in the language of cubical sets. Accordingly, we begin by computing the fibers of $\mathcal{G}\langle\mathbf{set}\rangle$ and $\mathcal{G}\langle\mathbf{set}^{\Pi}\rangle$:

$$A : \mathsf{F}_\square(\mathbf{set}_{\mathcal{C}}) \mid \mathcal{G}\langle\mathbf{set}\rangle[A] = \{\mathfrak{A} : \mathrm{U}_{\mathrm{IR}} \mid \mathsf{F}_\square(i_{\mathbf{set}_{\mathcal{C}}}(\mathsf{gl}(\mathfrak{A}))) = A\}$$

$$G : \mathsf{F}_\square(\mathbf{set}_{\mathcal{C}}^{\Pi}) \mid \mathcal{G}\langle\mathbf{set}^{\Pi}\rangle[G] = (\mathfrak{A} : \mathcal{G}\langle\mathbf{set}\rangle[(\!|G|\!)_0]) \times (a : \mathsf{F}_\square(\tau_{\mathcal{C}})[(\!|G|\!)_0])(\mathfrak{a} : (\mathcal{G}\langle\tau\rangle[[\mathfrak{A}]_{\mathrm{IR}}])[a]) \Rightarrow \mathcal{G}\langle\mathbf{set}\rangle[(\!|G|\!)_1\ a]$$

With these fibers in hand, we may define $\mathsf{E}_{\mathcal{G}\langle\widehat{\mathrm{pi}}\rangle}$:

$$\mathsf{E}_{\mathcal{G}\langle\widehat{\mathrm{pi}}\rangle}[G](\mathfrak{A}, \mathfrak{B}) = \mathtt{pi}(\mathfrak{A}, \lambda[a, \mathfrak{a}].\ \mathfrak{B}\ a\ \mathfrak{a})$$

❦

Lemma 6.36. *The foregoing construction of $\mathcal{G}\langle\tilde{\mathbf{set}} \xrightarrow{\mathbf{el}} \mathbf{set}\rangle$ is boundary separated.*

Proof. By induction, using the fact that $(\tilde{\mathbf{set}} \xrightarrow{\mathbf{el}} \mathbf{set})_{\mathcal{C}}$ is boundary separated. □

Construction 6.37 (Lifting structure)**.** The type-case lifting structure of Specification 4.23 may be constructed using the induction-recursion principle of $\mathcal{G}\langle\mathbf{set}\rangle$, and using the corresponding lifting structure for $\mathbf{set}_{\mathcal{C}}$. ❦

Lemma 6.38 (Coercion and composition)**.** *There are coercion and compositions operations defined on glued lines of sets $\mathcal{G}\langle\mathbb{I}\rangle \Rightarrow \mathcal{G}\langle\mathbf{set}\rangle$ which have the types given in Remark 4.27, satisfying the regularity law (Specification 4.28) as well as the connective-specific equations of Specification 4.30.*

Proof. Coercion and *homogeneous* composition operations are first defined using the induction principle of the inductive recursive universe of sets, taking the equations of Specification 4.30 and Lemma 4.31 respectively as definitions; general composition is defined by the standard reduction to coercion and homogeneous composition (Lemma 4.29) [ABC+19, AHH17]. □

7. Canonicity for XTT

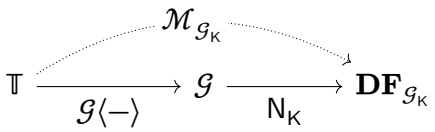
In Section 5, we introduced the logos $\mathcal{G}$ of *computability families*, along with a representable map functor $\mathbb{T} \xrightarrow{\mathcal{G}\langle -\rangle} \mathcal{G}$. These constructions contain the essence of the proof of canonicity for XTT, but in order to complete the proof we must assemble $\mathcal{G}$ and $\mathcal{G}\langle -\rangle$ into a model of $\mathbb{T}$ equipped with a morphism to $\mathcal{M}_{\mathcal{C}}$. In this section, we construct this *gluing model* of XTT and prove the following canonicity theorem:

Proposition 7.1 (Canonicity)**.** *Given a term $\cdot \vdash M : \mathsf{bool}$, either $\cdot \vdash M = \mathsf{tt} : \mathsf{bool}$ or $\cdot \vdash M = \mathsf{ff} : \mathsf{bool}$.*

7.1. The canonicity model of XTT.

Recall from Section 4 that a model of $\mathbb{T}$ is a category with a terminal object $\mathcal{C}$ paired with a representable map functor $\mathbb{T} \xrightarrow{\mathcal{M}_{\mathcal{C}}} \mathbf{DF}_{\mathcal{C}}$. The category of computability families $\mathcal{G}$ is not a category of discrete fibrations, and therefore we cannot directly take $\mathcal{G}, \mathcal{G}\langle -\rangle$ as the gluing model. Instead, following [SA20] we will shift to working with discrete fibrations on *compact* computability families $\mathbf{DF}_{\mathcal{G}_{\mathsf{K}}}$ and use the representable map functor $\mathcal{G} \xrightarrow{\mathsf{N}_{\mathsf{K}}} \mathbf{DF}_{\mathcal{G}_{\mathsf{K}}}$ to uniformly transfer the computability families from Section 5 from $\mathcal{G}$ to $\mathbf{DF}_{\mathcal{G}_{\mathsf{K}}}$.

Construction 7.2 (Gluing model)**.** The *gluing model* $\mathbb{T} \xrightarrow{\mathcal{M}_{\mathcal{G}_{\mathsf{K}}}} \mathbf{DF}_{\mathcal{G}_{\mathsf{K}}}$ is defined as the composition $\mathsf{N}_{\mathsf{K}} \circ \mathcal{G}\langle -\rangle$. Diagrammatically:



❦

It remains to construct the morphism $\mathcal{M}_{\mathcal{G}_{\mathsf{K}}} \longrightarrow \mathcal{M}_{\mathcal{C}} : \mathbf{Mod}_{\mathbb{T}}$. We will begin by constructing the data of this morphism, checking the Beck-Chevalley condition afterward.

Construction 7.3 (Gluing homomorphism data)**.** We expect a morphism of models tracked by the gluing fibration $\mathcal{G}_{\mathsf{K}} \xrightarrow{\mathsf{gl}_{\mathsf{K}}} \mathcal{C}$ at the level of contexts; it remains to make a choice of functors $J_{\mathcal{G}_{\mathsf{K}}} \longrightarrow J_{\mathcal{C}} : \mathbf{DF}$ lying over gl_{K} in $\mathbf{DF} \longrightarrow \mathbf{Cat}$ (natural in $J : \mathbb{T}$) to exhibit the

action of the homomorphism on judgments. Fixing a judgment $J : \mathbb{T}$, we construct each of components as follows:

$$\begin{array}{ccccccc} J_{\mathcal{G}_\mathsf{K}} & \xrightarrow{\cong} & \mathsf{N}_\mathsf{K}(\mathcal{G}\langle J\rangle) & \xrightarrow{[\Delta^1,\mathsf{gl}]} & \mathsf{gl}(\mathcal{G}\langle J\rangle) & \xrightarrow{\chi_J} & J_{\mathcal{C}} \\ \downarrow & & \downarrow & & \downarrow & & \downarrow \\ \mathcal{G}_\mathsf{K} & \xrightarrow[\mathbf{id}_{\mathcal{G}_\mathsf{K}}]{} & \mathcal{G}_\mathsf{K} & \xrightarrow[\mathsf{gl}_\mathsf{K}]{} & \mathcal{C} & \xrightarrow[\mathbf{id}_{\mathcal{C}}]{} & \mathcal{C} \end{array} \qquad \begin{array}{c} \mathbf{DF} \\ \downarrow \\ \mathbf{Cat} \end{array} \tag{7.4}$$

Diagram 7.4 above exhibits a natural transformation, because χ_J is a component of the natural transformation $\mathsf{gl} \circ \mathcal{G}\langle -\rangle \xrightarrow{\chi_\bullet} \mathcal{M}_{\mathcal{C}}$. ❦

It remains to check that the naturality squares induced by Construction 7.3 at representable maps satisfy the Beck-Chevalley condition. First, we record a simple characterization of the right adjoint $\mathbf{q}_f$ of a representable map in $\mathbf{DF}_{\mathcal{C}}$:

Lemma 7.5 [Uem19, Corollary 3.11]**.** *If $J \xrightarrow{f} I : \mathbf{DF}_{\mathcal{C}}$ is a representable map, the right adjoint $\mathbf{q}_f$ sends an element $\mathsf{y}(C) \xrightarrow{y} I$ to the upstairs element x determined by the following pullback square:*

$$\begin{array}{ccc} \mathsf{y}(C.y) & \overset{x}{\dashrightarrow} & J \\ \downarrow & \lrcorner & \downarrow{\scriptstyle f} \\ \mathsf{y}(C) & \xrightarrow[y]{} & I \end{array}$$

□

Lemma 7.6. *If $J \xrightarrow{f} I : \mathbb{T}$ is a representable map, the following naturality square satisfies the Beck-Chevalley condition:*

$$\begin{array}{ccc} J_{\mathcal{G}_\mathsf{K}} & \xrightarrow{J_{\mathsf{gl}_\mathsf{K}}} & J_{\mathcal{C}} \\ {\scriptstyle f_{\mathcal{G}_\mathsf{K}}}\downarrow & & \downarrow{\scriptstyle f_{\mathcal{C}}} \\ I_{\mathcal{G}_\mathsf{K}} & \xrightarrow[I_{\mathsf{gl}_\mathsf{K}}]{} & I_{\mathcal{C}} \end{array}$$

Proof. We must show that the following square commutes up to canonical isomorphism:

$$\begin{array}{ccc} I_{\mathcal{G}_\mathsf{K}} & \xrightarrow{I_{\mathsf{gl}_\mathsf{K}}} & I_{\mathcal{C}} \\ {\scriptstyle \mathbf{q}_{f_{\mathcal{G}_\mathsf{K}}}}\downarrow & & \downarrow{\scriptstyle \mathbf{q}_{f_{\mathcal{C}}}} \\ J_{\mathcal{G}_\mathsf{K}} & \xrightarrow[J_{\mathsf{gl}_\mathsf{K}}]{} & J_{\mathcal{C}} \end{array} \tag{7.7}$$

To begin with, we fix an element $\mathsf{K}(\Gamma) \xrightarrow{y} \mathcal{G}\langle J\rangle : J_{\mathcal{G}_\mathsf{K}}$. Inspecting the definition of $I_{\mathsf{gl}_\mathsf{K}}$ from Construction 7.3, we see that $I_{\mathsf{gl}_\mathsf{K}}(y) = \chi_I \circ \mathsf{gl}(y)$. Therefore, by Lemma 7.5,

$\mathbf{q}_{f_{\mathcal{C}}}(I_{\mathsf{gl}_{\mathsf{K}}}(y)) = \mathbf{q}_{f_{\mathcal{C}}}(\chi_I \circ \mathsf{gl}(y))$ is the top map of the following pullback square:

$$\begin{array}{ccc} \mathsf{y}(\cdots) & \dashrightarrow & J_{\mathcal{C}} \\ \downarrow & \lrcorner & \downarrow{\scriptstyle f_{\mathcal{C}}} \\ \mathsf{y}(\mathsf{gl}_{\mathsf{K}}(\Gamma)) & \xrightarrow[\chi_I \circ \mathsf{gl}(y)]{} & I_{\mathcal{C}} \end{array} \tag{7.8}$$

Similarly, we may compute $\mathbf{q}_{f_{\mathcal{C}}}(I_{\mathsf{gl}_{\mathsf{K}}}(y))$ as the top of the following composite square:

$$\begin{array}{ccccc} \mathsf{y}(\mathsf{gl}_{\mathsf{K}}(\cdots)) & \longrightarrow & \mathsf{gl}(\mathcal{G}\langle J\rangle) & \xrightarrow{\chi_J} & J_{\mathcal{C}} \\ \downarrow & \lrcorner & \downarrow{\scriptstyle \mathsf{gl}(\mathcal{G}\langle f\rangle)} & & \downarrow{\scriptstyle f_{\mathcal{C}}} \\ \mathsf{y}(\mathsf{gl}_{\mathsf{K}}(\Gamma)) & \xrightarrow[\mathsf{gl}(y)]{} & \mathsf{gl}(\mathcal{G}\langle I\rangle) & \xrightarrow[\chi_I]{} & I_{\mathcal{C}} \end{array} \tag{7.9}$$

To show that the top of Diagram 7.8 is isomorphic to Diagram 7.9, it suffices to check that the outer square of Diagram 7.9 is cartesian. Both χ_J and χ_I are isomorphisms so the right-hand square of Diagram 7.9 is cartesian, so the result follows immediately from the pullback pasting lemma. □

Corollary 7.10. *The natural transformation $\mathcal{M}_{\mathsf{gl}_{\mathsf{K}}}$ from Construction 7.3 is a morphism of models.* □

7.2. **The canonicity theorem.** Having constructed the gluing model and the projection onto $\mathcal{M}_{\mathcal{C}}$, we are now in a position to prove canonicity. First, we stop considering an arbitrary model $\mathcal{M}_{\mathcal{C}}$ and work exclusively with $\mathcal{M}_{\mathcal{J}}$, the bi-initial model of $\mathbb{T}$ (Theorem 3.54). Bi-initiality ensures that there is a morphism $\mathcal{M}_{\mathcal{J}} \xrightarrow{i_{\mathcal{G}_{\mathsf{K}}}} \mathcal{M}_{\mathcal{G}_{\mathsf{K}}}$ and, because $\mathbf{id}_{\mathcal{M}_{\mathcal{J}}}$ and $\mathcal{M}_{\mathsf{gl}_{\mathsf{K}}} \circ \mathcal{M}_{i_{\mathcal{G}_{\mathsf{K}}}}$ are both objects of $\mathbf{Model}_{\mathbb{T}}[\mathcal{M}_{\mathcal{J}}, \mathcal{M}_{\mathcal{J}}]$, there is a unique invertible 2-morphism $\mathbf{id}_{\mathcal{M}_{\mathcal{J}}} \xrightarrow{\iota} \mathcal{M}_{\mathsf{gl}_{\mathsf{K}}} \circ \mathcal{M}_{i_{\mathcal{G}_{\mathsf{K}}}}$.

In prior presentations of gluing with strict homomorphisms [SAG19, KHS19, CHS19, GKNB20], $\mathcal{M}_{\mathcal{J}}$ was initial in the *1-categorical* sense, so $\mathbf{id}_{\mathcal{M}_{\mathcal{J}}}$ and $\mathcal{M}_{\mathsf{gl}_{\mathsf{K}}} \circ \mathcal{M}_{i_{\mathcal{G}_{\mathsf{K}}}}$ were equal on this nose. This in turn implied that for every map $\diamond_{\mathcal{J}} \xrightarrow{f} (\mathrm{bool}^*\tau)_{\mathcal{J}}$, there existed a map $\diamond_{\mathcal{G}_{\mathsf{K}}} \xrightarrow{[\![f]\!]} (\mathrm{bool}^*\tau)_{\mathcal{G}_{\mathsf{K}}}$ such that $\mathsf{gl}_{\mathsf{K}}([\![f]\!]) = f$. Canonicity followed more-or-less immediately by inspection of $[\![f]\!]$.

In this work, we have used a weaker notion of morphism and as a consequence, $\mathcal{M}_{\mathcal{J}}$ is merely *bi*-initial. Accordingly, we cannot immediately conclude that f is in the image of $\mathcal{M}_{\mathsf{gl}_{\mathsf{K}}}$. In fact, while it is not generally the case that $f = f_{\mathsf{gl}_{\mathsf{K}} \circ i_{\mathcal{G}_{\mathsf{K}}}}$, it is possible to *realign* $x_{i_{\mathcal{G}_{\mathsf{K}}}}$ to an isomorphic arrow which does lie strictly over f.

Lemma 7.11 (Realignment)**.** *Given a morphism $\mathsf{y}(\Gamma) \xrightarrow{x} X_{\mathcal{J}} : \mathbf{DF}_{\mathcal{J}}$, there exists an object $[\![\Gamma]\!] : \mathcal{G}_{\mathsf{K}}$ and a morphism $\mathsf{y}([\![\Gamma]\!]) \xrightarrow{[\![x]\!]} X_{\mathcal{G}_{\mathsf{K}}} : \mathbf{DF}_{\mathcal{G}_{\mathsf{K}}}$ such that $\mathsf{gl}_{\mathsf{K}}([\![x]\!]) = x$ and $i_{\mathcal{G}_{\mathsf{K}}}(\Gamma) \cong [\![\Gamma]\!]$.*

Proof. First, we note that $i_{\mathcal{G}_\mathsf{K}}(x)$ lies over $\mathsf{gl}_\mathsf{K}(i_{\mathcal{G}_\mathsf{K}}(x))$ by definition. Moreover, because $\mathcal{G}_\mathsf{K} \xrightarrow{\mathsf{gl}_\mathsf{K}} \mathcal{C}$ is a fibration, we may construct a cartesian lift of $\Gamma \xrightarrow{\boldsymbol{\iota}} \mathsf{gl}_\mathsf{K}(i_{\mathcal{G}_\mathsf{K}}(\Gamma))$. Diagrammatically, there exists two squares:

$$\begin{array}{ccc} \boldsymbol{\iota}^*(i_{\mathcal{G}_\mathsf{K}}(\Gamma)) & \overset{\boldsymbol{\iota}^\dagger}{\dashrightarrow} & i_{\mathcal{G}_\mathsf{K}}(\Gamma) \\ \downarrow & & \downarrow \\ \Gamma & \xrightarrow[\boldsymbol{\iota}]{} & \mathsf{gl}_\mathsf{K}(i_{\mathcal{G}_\mathsf{K}}(\Gamma)) \end{array} \qquad \begin{array}{ccc} \mathsf{y}(i_{\mathcal{G}_\mathsf{K}}(\Gamma)) & \xrightarrow{i_{\mathcal{G}_\mathsf{K}}(x)} & X_{\mathcal{G}_\mathsf{K}} \\ \downarrow & & \downarrow \\ \mathsf{y}(\mathsf{gl}_\mathsf{K}(i_{\mathcal{G}_\mathsf{K}}(\Gamma))) & \xrightarrow[\mathsf{gl}_\mathsf{K}(i_{\mathcal{G}_\mathsf{K}}(x))]{} & X_{\mathcal{J}} \end{array}$$

Observe that $\boldsymbol{\iota}^\dagger$ is an isomorphism because it is cartesian over an isomorphism. Unfolding the definition of a 2-morphism, we see that $\mathsf{gl}_\mathsf{K}(i_{\mathcal{G}_\mathsf{K}}(x)) \circ \mathsf{y}(\boldsymbol{\iota}) = x$. Accordingly, we may paste together these two diagrams to obtain the following:

$$\begin{array}{ccccc} \mathsf{y}(\boldsymbol{\iota}^*(i_{\mathcal{G}_\mathsf{K}}(\Gamma))) & & \xrightarrow{i_{\mathcal{G}_\mathsf{K}}(x) \circ \mathsf{y}(\boldsymbol{\iota}^\dagger)} & & X_{\mathcal{G}_\mathsf{K}} \\ \downarrow & & & & \downarrow \\ \mathsf{y}(\Gamma) & \xrightarrow{\quad} & \mathsf{gl}_\mathsf{K}(i_{\mathcal{G}_\mathsf{K}}(x)) \circ \mathsf{y}(\boldsymbol{\iota}) & \xrightarrow{\quad} & X_{\mathcal{J}} \\ & & \underset{x}{\searrow} & & \nearrow \end{array}$$

Therefore, $[\![x]\!] \overset{def}{=\!=} i_{\mathcal{G}_\mathsf{K}}(x) \circ \mathsf{y}(\boldsymbol{\iota}^\dagger)$ is a morphism lying strictly over x. The final condition is immediate because $[\![\Gamma]\!] \cong i_{\mathcal{G}_\mathsf{K}}(\Gamma)$ by definition. □

Remark 7.12. One might hope that Lemma 7.11 implies the existence of a morphism $\mathcal{M}_\mathcal{J} \xrightarrow{\mathcal{M}_j} \mathcal{M}_{\mathcal{G}_\mathsf{K}}$ which satisfies the identity $\mathcal{M}_{\mathsf{gl}_\mathsf{K}} \circ \mathcal{M}_j = \mathbf{id}_\mathcal{J}$ on the nose. Simply applying the realignment procedure to every element of $\mathcal{M}_\mathcal{J}$ does not result in a morphism, however, because it is not functorial, merely pseudo-functorial. This should not be surprising: realignment relies on a choice of cartesian lift, which is only pseudo-functorial in general. ❦

Theorem 7.13 (Canonicity)**.** *Given a term* $\cdot \vdash M : \mathsf{bool}$*, either* $\cdot \vdash M = \mathsf{tt} : \mathsf{bool}$ *or* $\cdot \vdash M = \mathsf{ff} : \mathsf{bool}$*.*

Proof. A closed term $\cdot \vdash M : \mathsf{bool}$ is just a morphism $\mathsf{y}(\mathbf{1}_\mathcal{J}) \xrightarrow{x} (\mathsf{bool}^*\tau)_\mathcal{J} : \mathbf{DF}_\mathcal{J}$; it suffices to prove that $x = \mathsf{tt}_\mathcal{J}$ or $x = \mathsf{ff}_\mathcal{J}$. First, by Lemma 7.11 we obtain a morphism $\mathsf{y}([\![\diamond]\!]) \xrightarrow{[\![x]\!]} (\mathsf{bool}^*\tau)_{\mathcal{G}_\mathsf{K}} : \mathbf{DF}_{\mathcal{G}_\mathsf{K}}$ lying over x. Next, by definition we have $(\mathsf{bool}^*\tau)_{\mathcal{G}_\mathsf{K}} = \mathsf{N}_\mathsf{K}(\mathcal{G}\langle \mathsf{bool}^*\tau \rangle)$ and so $[\![x]\!]$ is uniquely determined by morphism $\mathbf{1}_\mathcal{G} \xrightarrow{\widetilde{[\![x]\!]}} \mathcal{G}\langle \mathsf{bool}^*\tau \rangle : \mathcal{G}$. Unfolding further, any morphism $\mathbf{1}_\mathcal{G} \xrightarrow{\widetilde{[\![x]\!]}} \mathcal{G}\langle \mathsf{bool}^*\tau \rangle$ must be a commuting square of the following shape in $\mathbf{Set}_\square$:

$$\begin{array}{ccc} \mathbf{1}_{\mathbf{Set}_\square} & \xrightarrow{\mathsf{E}_{\widetilde{[\![x]\!]}}} & \mathbf{2}_{\mathbf{Set}_\square} \\ \cong \downarrow & & \downarrow \\ \mathsf{F}_\square(\diamond_\mathcal{C}) & \xrightarrow[\mathsf{F}_\square(\mathsf{gl}(y))]{} & \mathsf{F}_\square((\mathsf{bool}^*\tau)_\mathcal{C}) \end{array} \tag{7.14}$$

It is immediate that $\mathsf{E}_{\widetilde{[\![x]\!]}} = \mathbf{inl}$ or $\mathsf{E}_y = \mathbf{inr}$; the fact that Diagram 7.14 commutes ensures that $\widetilde{[\![x]\!]}$ lies over either $\mathrm{tt}_{\mathcal{J}}$ or $\mathrm{ff}_{\mathcal{J}}$. □

8. Perspective and outlook

For decades now, the puzzle of Martin-Löf's intensional identity type has remained at the center of type theorists' minds. In 1994, Streicher showed that intensional type theory was independent of seemingly sensible principles (like function extensionality) by constructing extremely intensional counter-models [Str94]; in 1998, Hofmann and Streicher went a step further and demonstrated that the same type theory was independent of the uniqueness of identity proofs (UIP) principle by constructing a model of type theory in groupoids [HS98].

Hofmann and Streicher's contribution showed that it was possible for identity in a universe of sets to "mean" bijection, a precursor to the univalence principle of Voevodsky [Voe06, Voe10], later codified in the language of homotopy type theory (HoTT) [Uni13]. Later, it was discovered that the identity type conferred an infinite-dimensional structure already familiar in the context of homotopy theory [AW09, vdBG11, Lum10].

Cubical type theories were invented in order to repair several semantic and syntactic anomalies of the new homotopy type theory; homotopy type theory lacks *canonicity*, a property closely related to but distinct from the existence of a computational "proofs-as-programs" interpretation of the language. On the other hand, the standard model of homotopy type theory in *simplicial sets* [KL16] must be formulated in a boolean metatheory [BC15]. The discovery of a constructive model for univalent type theory in cubical sets [BCH14] sparked a flurry of work on explicitly cubical type theories [CCHM17, AHH17, ABC$^+$19] which resolved both the matters of canonicity and computational interpretation [Hub18, AHH17].

While the benefits of cubical ideas for solving problems in infinite-dimensional type theory are clear, we believed that it might be possible to bring the cubical perspective to bear on the problems of *traditional* one-dimensional type theory, in which the intensional identity type is augmented with enough uniqueness and extensionality principles for it to behave like classical mathematical equality. In the context of the strongest possible such uniqueness principle, *equality reflection*, it remains an open question whether it is possible to implement a usable proof assistant; on the other hand, extending type theories with axioms for function extensionality destroys canonicity and has significant usability problems.

Inspired by the work of Altenkirch, McBride, and Swierstra on *Observational Type Theory* [AM06, AMS07], which internalized aspects of the setoid model of type theory, we sought to internalize Coquand's semantic universe of Bishop sets [Coq17] as a type theory in its own right, XTT. We believe that XTT is an ideal language for *dependently typed programming*, in which it is very important that coercions may be erased prior to execution (a procedure that cannot be applied to coercions arising from the univalence principle). Unfortunately, there are several obstacles rendering both OTT and XTT unsuitable for use as languages for formalizing general mathematics, disadvantages not shared by homotopy type theory or its cubical variants.

8.1. **Lack of standard universes.**

In mathematics, a universe is a "family of (some) families", an object from which every family in some class arises by pullback; we cannot have a universe of all families for general reasons, but there are several restrictions of this

naïve idea that make sense, such as a universe of all monomorphisms (a subobject classifier), or a universe of κ-small families for some regular or inaccessible cardinal κ.

Universes in mathematics are important for two reasons: first, they tame subtle but essentially bureaucratic questions of size [AGV72, Exposé I, Ch. 0], and second, they reconstruct the rational aspects of the set-theoretical *axiom of replacement*, enabling concepts to be formulated in the convenient fiber-wise style familiar from dependent type theory [Str05]. Here, it is very important to ensure that the universe imposes no spurious structures on the maps it classifies; for instance, if U is the universe of κ-small maps, we may develop the theory of κ-small groups in terms of U. Then, a predicate defined over κ-small groups should have the same meaning as a predicate defined over U-groups.

The universes of OTT and XTT are *not* of this kind: the existence of a code $\widehat{A}$ for a type A in these universes expresses not only the smallness of A, but also the fact that A is either a dependent product, a dependent sum, an equation, or it is the booleans (etc.), and the same for all of A's subterms. Therefore, a mathematical statement quantifying over elements $\widehat{A}$ does *not* have the right meaning, considering these additional assumptions on the form of the decoding of A; worse, there exist perverse functions *out of* the universe which reflect on the encoding of types.

Some have suggested that "parametric" or uniform quantifiers may be used to avoid these pathologies, but it is quite easy to see that parametricity cannot be used to solve the fundamental problems. For instance, the perverse deduction "If $A \to B$ is equal to $C \to D$, then A is equal to C" is *not* ruled out by parametric quantification; of course, this deduction (a mathematical taboo) is the *raison d'être* for the non-standard universes of OTT/XTT as detailed in Section 2.3.4.

8.2. **What is a proposition?** A considerably more subtle obstacle for using either OTT or XTT in the formalization of mathematics is to be found when choosing a suitable notion of *proposition* or *relation*. In type theory, there are *a priori* two ways to formalize propositions:

(1) A *strict* proposition is a type whose elements are all judgmentally equal.
(2) A *weak* proposition is a type X together with a function $(x, y : X) \to \mathsf{path}_X(x, y)$.

In OTT/XTT, the strict and weak notions of proposition do not agree, though they could be forced to agree by adding equality reflection. Unfortunately, when investigating the interplay between the indispensible principles of function comprehension and effectivity of equivalence relations, we will find that this mismatch cannot be resolved by favoring either the strict or the weak notion.

As soon as one has chosen a notion of proposition, one may consider the corresponding "squash type", the reflection of types into propositions:

(1) Given a type A, the strict squash type $|A|_s$ is a strict proposition; a function $|A|_s \to B$ is a function $f : A \to B$ such that every $f(x)$ is judgmentally equal to $f(y)$. The strict squash type was investigated by Awodey and Bauer [AB04], and appears in recent versions of Coq and Agda [GCST19].
(2) Given a type A, the weak squash type $|A|_w$ is a weak proposition; a function $|A|_w \to B$ is a function $f : A \to B$ together with a function assigning to each $x, y : A$ an element of $\mathsf{path}_B(f(x), f(y))$. The weak squash type appears in homotopy type theory as *propositional truncation* [Uni13].

Dependent product and binary product preserve the property of being a (strict, weak) proposition, and may therefore be used as universal quantification and conjunction in a

logic of propositions. Dependent sum and binary sum do not preserve this property, but they can be squashed in order to define existential quantification and disjunction. The logic of (strict, weak) propositions is summarized below:

$$\begin{aligned}
\forall x : A.P(x) &\stackrel{def}{=\!=} (x : A) \to P(x) \\
P \supset Q &\stackrel{def}{=\!=} P \to Q \\
P \wedge Q &\stackrel{def}{=\!=} P \times Q \\
\exists^{s/w} x : A.P(x) &\stackrel{def}{=\!=} |(x : A) \times P(x)|_{s/w} \\
P \vee^{s/w} Q &\stackrel{def}{=\!=} |(x : \mathsf{bool}) \times \mathsf{if}(x; P, Q)|_{s/w} \\
(M =_A N) &\stackrel{def}{=\!=} \mathsf{path}_A(M, N)
\end{aligned}$$

8.2.1. *Function comprehension.* What is the meaning of "function"? There is only one possible answer: it is an element of an exponential object. In some categories, however, these exponentials can be reconstructed as representing objects for collections of *functional relations*. This isomorphism between the collection of functions $A \to B$ and subobjects of $A \times B$ satisfying a unique existence property is traditionally referred to as the "axiom of unique choice", though it is perhaps better to refer to it as *function comprehension*. Function comprehension is a crucial feature of both classical *and* constructive mathematics, and life becomes very difficult in categories where function comprehension fails.

A (strict, weak) functional relation from A to B is a family of (strict, weak) propositions $x : A, y : B \vdash R(x, y)$ together with a proof of the following (strict, weak) proposition:

$$\forall x : A.\ \exists^{s/w} y : B.\ R(x, y) \wedge \forall y' : B.\ R(x, y') \supset y =_B y' \qquad \text{(functionality)}$$

The proposition above may be rendered into the language of types as follows:

$$(x : A) \to |(y : B) \times R(x, y) \times ((y' : B) \to R(x, y') \to \mathsf{path}_B(y, y'))|_{s/w}$$

The function comprehension principle is immediate for weak propositions in OTT and XTT, but fails for strict propositions.

(1) To exhibit a function $A \to B$ from a weak functional relation, we may extract the $y : B$ using the universal property of the weak squash type, fulfilling the auxiliary obligation using the weak uniqueness of y with $R(x, y)$.
(2) In doing the same with a strict functional relation, we run into a problem: to make a function out of an element of the strict squash type, we end up needing that y is unique with $R(x, y)$ *up to judgmental equality*, but we have only an element of $\mathsf{path}_B(y, y')$ for each y' such that $R(x, y')$.

Therefore, short of adding equality reflection, we must conclude that the weak notion of proposition is the "correct" one, and the strict one is not particularly useful for mathematics in an environment without equality reflection. Unfortunately, we will see that another indispensible reasoning principle in constructive and classical mathematics, the effectivity of equivalence relations, appears to be compatible only with the strict notion in a boundary separated environment lacking equality reflection. (In contrast, full cubical type theory satisfies a vastly stronger exactness condition called *descent*, generalizing both the disjointness of coproducts and the effectivity of equivalence relations.)

8.2.2. *Effectivity of equivalence relations.* It is possible to add quotient types to both XTT and OTT (an extension implemented, for instance, as part of the experimental Epigram 2 proof assistant [McB10]); likewise, Nuprl has supported a version of quotient types for decades. Unfortunately, these quotients can be made to have good properties only for *certain* equivalence relations:

(1) In Nuprl, only equivalence relations valued in "strong propositions" (types having at most one element up to the intensional *untyped* equivalence of Howe [How89]) have good quotients. Equivalence relations valued in general propositions (types having at most one element up to extensional equality) do *not* necessarily have good quotients, a serious problem alluded to in the work of Nogin [Nog02].
(2) In OTT and XTT, only equivalence relations valued in strict propositions can have good quotients.

Writing $[-] : A \to A/R$ for the quotient map, the quotient A/R is "good" when each type $\mathsf{path}_{A/R}([x],[y])$ is equivalent to $R(x,y)$; this property, called the effectivity of R, does not follow from the rules of quotient types alone and in fact fails in many categories (such as categories of partial equivalence relations). The effectivity of *all* equivalence relations is, however, indispensible for practical use of quotients in mathematics.

In type theory, the effectivity of equivalence relations follows from propositional extensionality, a restricted version of the univalence principle that places bi-implications $(f,g) : P \leftrightarrow Q$ into correspondence with proofs of equality $\mathsf{pua}(f,g) : \mathsf{path}(P,Q)$; in topos theory, this corresponds to the existence of a subobject classifier. We will see, however, that it is not possible to extend either XTT or OTT (or any type theory satisfying boundary separation or definitional UIP) with a univalence principle for weak propositions without some fundamentally new ideas.

In cubical type theories, univalence is supported by means of a special connective taking an equivalence of types and returning a path between the corresponding types [Ang19]; we might attempt to extend XTT by a version of this connective restricted to weak propositions:

$$\frac{\begin{array}{c}\Gamma \vdash r : \mathbb{I} \qquad \Gamma, r = 0 \vdash P\ \mathit{prop} \qquad \Gamma \vdash Q\ \mathit{prop} \\ \Gamma, r = 0 \vdash f : P \to Q \qquad \Gamma, r = 0 \vdash g : Q \to P\end{array}}{\Gamma \vdash \bigvee_r(P,Q,f,g)\ \mathit{prop}\ [\partial(r) \to [r = 0 \to P \mid r = 1 \to Q]]}$$

Unfortunately, we can show that a univalent universe of weak propositions Prop cannot be boundary separated.

Lemma 8.1. *Suppose that we have a boundary separated universe of propositions closed under $\bigvee$-types (thence univalent); then, all equivalences between two propositions are judgmentally equal.*

Proof. Let P, Q be two propositions classified by the univalent universe of propositions, and let (f,g) and (f',g') be two equivalences between them. Abstracting a dimension $i : \mathbb{I}$, we therefore have two $\bigvee$-types $\bigvee_i(P,Q,f,g)$ and $\bigvee_i(P,Q,f',g')$; by boundary separation, we in fact have $\bigvee_i(P,Q,f,g) = \bigvee_i(P,Q,f',g')$.

We will show that $f = f'$ judgmentally by using coercion in the $\bigvee$-type, following the computation rules described by Angiuli [Ang19].

$$\begin{aligned} \lambda x.f(x) &= \lambda x.\mathsf{coe}^{0 \rightsquigarrow 1}_{\_.Q}\, f(x) && \text{regularity} \\ &= \lambda x.\mathsf{coe}^{0 \rightsquigarrow 1}_{i.\bigvee_i(P,Q,f,g)}\, x && \text{[Ang19, p.163]} \end{aligned}$$

$$
\begin{aligned}
&= \lambda x.\mathsf{coe}^{0 \rightsquigarrow 1}_{i.\mathsf{V}_i(P,Q,f',g')}\, x && \text{boundary separation} \\
&= \lambda x.\mathsf{coe}^{0 \rightsquigarrow 1}_{\_.Q}\, f'(x) && \text{[Ang19, p.163]} \\
&= \lambda x.f'(x) && \text{regularity}
\end{aligned}
$$

To see that $g = g'$, simply repeat the procedure with the inverse equivalences. □

Remark 8.2. Lemma 8.1 can likewise be replayed when glue-types à la [ABC+19] are used instead of V-types à la [AHH17]. In either case, coercion can be used (modulo regularity) to recover the equivalence from the line of types. ❦

The assumptions of Lemma 8.1 imply some intensional type theoretic taboos.

Corollary 8.3. *Under the assumptions of Lemma 8.1, all propositions are strict propositions.*

Proof. Let P be a proposition, and let x, y be proofs of P. The constant functions $\lambda\_.x$ and $\lambda\_.y$ are both equivalences $P \leftrightarrow P$; by Lemma 8.1, they are judgmentally equal. Therefore, x and y are judgmentally equal. □

Corollary 8.4. *Under the assumptions of Lemma 8.1, equality reflection holds.*

Proof. Let $A : \mathsf{set}$ and let $a \in A$; then $S_A(a) \stackrel{def}{=\!=} (x \in A) \times \mathsf{path}_{\mathsf{el}(A)}(a, x)$ is a weak proposition. By Corollary 8.3, $S_A(a)$ is moreover a strict proposition. Let $a' \in A$ and $p : \mathsf{path}_{\mathsf{el}(A)}(a, a')$; therefore we have $\langle a, \lambda\_.a\rangle = \langle a', p\rangle : S_A(a)$, whence $a = a'$ judgmentally. □

Acknowledgment

We thank Thorsten Altenkirch, Mathieu Anel, Steve Awodey, Lars Birkedal, Evan Cavallo, David Thrane Christiansen, Thierry Coquand, Kuen-Bang Hou (Favonia), Marcelo Fiore, Jonas Frey, Ambrus Kaposi, Krzysztof Kapulkin, Alex Kavvos, András Kovács, Dan Licata, Conor McBride, Darin Morrison, Anders Mörtberg, Michael Shulman, Bas Spitters, and Thomas Streicher for helpful conversations about extensional equality, algebraic type theory, and categorical gluing. We thank our anonymous reviewers for their insightful comments, and especially thank Robert Harper for valuable conversations throughout the development of this work.

The authors gratefully acknowledge the support of the Air Force Office of Scientific Research through MURI grant FA9550-15-1-0053. 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

[AB04] Steven Awodey and Andrej Bauer. Propositions As [Types]. *J. Log. and Comput.*, 14(4):447–471, August 2004.

[ABC+19] Carlo Angiuli, Guillaume Brunerie, Thierry Coquand, Kuen-Bang Hou (Favonia), Robert Harper, and Daniel R. Licata. Syntax and models of cartesian cubical type theory. Preprint, February 2019.

[Abe13] Andreas Abel. *Normalization by Evaluation: Dependent Types and Impredicativity.* Habilitation, Ludwig-Maximilians-Universität München, 2013.

[ABFJ17] Mathieu Anel, Georg Biedermann, Eric Finster, and André Joyal. A generalized blakers-massey theorem. 2017.

[ABKT19] Thorsten Altenkirch, Simon Boulier, Ambrus Kaposi, and Nicolas Tabareau. Setoid type theory—a syntactic translation. In Graham Hutton, editor, *Mathematics of Program Construction*, pages 155–196, Cham, 2019. Springer International Publishing.

[ABSS14] Steve Awodey, Carsten Butz, Alex Simpson, and Thomas Streicher. Relating first-order set theories, toposes and categories of classes. *Annals of Pure and Applied Logic*, 165(2):428–502, 2014.

[ACD08] Andreas Abel, Thierry Coquand, and Peter Dybjer. Verifying a semantic $\beta\eta$-conversion test for Martin-Löf Type Theory. In Philippe Audebaud and Christine Paulin-Mohring, editors, *Mathematics of Program Construction*, pages 29–56, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg.

[ACH+18a] Carlo Angiuli, Evan Cavallo, Kuen-Bang Hou (Favonia), Robert Harper, Anders Mörtberg, and Jonathan Sterling. redtt: implementing Cartesian cubical type theory, 2018. Dagstuhl Seminar 18341: Formalization of Mathematics in Type Theory.

[ACH+18b] Carlo Angiuli, Evan Cavallo, Kuen-Bang Hou (Favonia), Robert Harper, and Jonathan Sterling. The RedPRL Proof Assistant (Invited Paper). In *Proceedings of the 13th International Workshop on Logical Frameworks and Meta-Languages: Theory and Practice, LFMTP@FSCD 2018, Oxford, UK, 7th July 2018.*, pages 1–10, 2018.

[ACP09] Andreas Abel, Thierry Coquand, and Miguel Pagano. A modular type-checking algorithm for type theory with singleton types and proof irrelevance. In Pierre-Louis Curien, editor, *Typed Lambda Calculi and Applications*, pages 5–19, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

[AGV72] Michael Artin, Alexander Grothendieck, and Jean-Louis Verdier. *Théorie des topos et cohomologie étale des schémas*. Springer-Verlag, Berlin, 1972. 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, Lecture Notes in Mathematics, Vol. 269, 270, 305.

[AHH17] Carlo Angiuli, Kuen-Bang Hou (Favonia), and Robert Harper. Computational higher type theory III: Univalent universes and exact equality. 2017.

[AHH18] Carlo Angiuli, Kuen-Bang Hou (Favonia), and Robert Harper. Cartesian Cubical Computational Type Theory: Constructive Reasoning with Paths and Equalities. In Dan Ghica and Achim Jung, editors, *27th EACSL Annual Conference on Computer Science Logic (CSL 2018)*, volume 119 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 6:1–6:17, Dagstuhl, Germany, 2018. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

[AJ19] Mathieu Anel and André Joyal. Topo-logie. Preprint, March 2019.

[AK16] Thorsten Altenkirch and Ambrus Kaposi. Normalisation by Evaluation for Dependent Types. In Delia Kesner and Brigitte Pientka, editors, *1st International Conference on Formal Structures for Computation and Deduction (FSCD 2016)*, volume 52 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 6:1–6:16, Dagstuhl, Germany, 2016. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

[All87] Stuart Frazier Allen. *A non-type-theoretic semantics for type-theoretic language*. PhD thesis, Cornell University, Ithaca, NY, USA, 1987.

[Alt99] Thorsten Altenkirch. Extensional equality in intensional type theory. In *Proceedings. 14th Symposium on Logic in Computer Science (Cat. No. PR00158)*, pages 412–420, July 1999.

[AM06] Thorsten Altenkirch and Conor McBride. Towards Observational Type Theory, 2006.

[AMS07] Thorsten Altenkirch, Conor McBride, and Wouter Swierstra. Observational equality, now! In *Proceedings of the 2007 Workshop on Programming Languages Meets Program Verification*, PLPV '07, pages 57–68, Freiburg, Germany, 2007. ACM.

[Ang19] Carlo Angiuli. *Computational Semantics of Cartesian Cubical Type Theory*. PhD thesis, Carnegie Mellon University, 2019.

[AOV17] Andreas Abel, Joakim Öhman, and Andrea Vezzosi. Decidability of conversion for type theory in type theory. *Proc. ACM Program. Lang.*, 2:23:1–23:29, December 2017.

[AS12] Andreas Abel and Gabriel Scherer. On Irrelevance and Algorithmic Equality in Predicative Type Theory. *Logical Methods in Computer Science*, Volume 8, Issue 1, March 2012.

[AW09] Steve Awodey and Michael A. Warren. Homotopy theoretic models of identity types. *Mathematical Proceedings of the Cambridge Philosophical Society*, 146(1):45–55, January 2009.

[Awo08] Steve Awodey. A brief introduction to algebraic set theory. *Bulletin of Symbolic Logic*, 14(3):281–298, September 2008.

[Awo10] Steve Awodey. *Category Theory*. Oxford University Press, Inc., New York, NY, USA, 2nd edition, 2010.

[Awo15] Steve Awodey. Notes on cubical models of type theory. 2015.

[Awo18a] Steve Awodey. A cubical model of homotopy type theory. *Annals of Pure and Applied Logic*, 169(12):1270–1294, 2018. Logic Colloquium 2015.

[Awo18b] Steve Awodey. Natural models of homotopy type theory. *Mathematical Structures in Computer Science*, 28(2):241–286, 2018.

[BC15] Marc Bezem and Thierry Coquand. A kripke model for simplicial sets. *Theor. Comput. Sci.*, 574(C):86–91, April 2015.

[BCH14] Marc Bezem, Thierry Coquand, and Simon Huber. A Model of Type Theory in Cubical Sets. In Ralph Matthes and Aleksy Schubert, editors, *19th International Conference on Types for Proofs and Programs (TYPES 2013)*, volume 26 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 107–128, Dagstuhl, Germany, 2014. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

[BGH+16] Andrej Bauer, Gaëtan Gilbert, Philipp Haselwarter, Matija Pretnar, and Christopher A. Stone. Design and implementation of the Andromeda proof assistant. TYPES, 2016.

[Bis67] Errett Bishop. *Foundations of Constructive Analysis*. McGraw-Hill, New York, 1967.

[Bor94a] Francis Borceux. *Handbook of Categorical Algebra*, volume 1 of *Encyclopedia of Mathematics and its Applications*. Cambridge University Press, 1994.

[Bor94b] Francis Borceux. *Handbook of Categorical Algebra 2 – Categories and Structures*. Cambridge University Press, 1994.

[Bor10] Francis Borceux. *Handbook of Categorical Algebra 3 – Categories of Sheaves*. Cambridge University Press, 2010.

[BP15] Andrej Bauer and Matija Pretnar. Programming with algebraic effects and handlers. *Journal of Logical and Algebraic Methods in Programming*, 84(1):108–123, 2015. Special Issue: The 23rd Nordic Workshop on Programming Theory (NWPT 2011).

[CAB+86] R. L. Constable, S. F. Allen, H. M. Bromley, W. R. Cleaveland, J. F. Cremer, R. W. Harper, D. J. Howe, T. B. Knoblock, N. P. Mendler, P. Panangaden, J. T. Sasaki, and S. F. Smith. *Implementing Mathematics with the Nuprl Proof Development System*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1986.

[Car86] John Cartmell. Generalised algebraic theories and contextual categories. *Annals of Pure and Applied Logic*, 32:209–243, 1986.

[CCHM17] Cyril Cohen, Thierry Coquand, Simon Huber, and Anders Mörtberg. Cubical Type Theory: a constructive interpretation of the univalence axiom. *IfCoLog Journal of Logics and their Applications*, 4(10):3127–3169, November 2017.

[CD14] Pierre Clairambault and Peter Dybjer. The biequivalence of locally cartesian closed categories and martin-löf type theories. *Mathematical Structures in Computer Science*, 24(6), 2014.

[CFM18] James Chapman, Fredrik Nordvall Forsberg, and Conor McBride. The Box of Delights (Cubical Observational Type Theory). Unpublished note, January 2018.

[CHS19] Thierry Coquand, Simon Huber, and Christian Sattler. Homotopy canonicity for cubical type theory. In Herman Geuvers, editor, *4th International Conference on Formal Structures for Computation and Deduction (FSCD 2019)*, volume 131 of *Leibniz International Proceedings in Informatics (LIPIcs)*, Dagstuhl, Germany, 2019. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

[CMS20] Evan Cavallo, Anders Mörtberg, and Andrew W Swan. Unifying Cubical Models of Univalent Type Theory. In Maribel Fernández and Anca Muscholl, editors, *28th EACSL Annual Conference on Computer Science Logic (CSL 2020)*, volume 152 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 14:1–14:17, Dagstuhl, Germany, 2020. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

[Con82] Robert L. Constable. Intensional analysis of functions and types. techreport CSR-118-82, 1982.

[Coq91] Thierry Coquand. *An algorithm for testing conversion in type theory*, pages 255–279. Cambridge University Press, 1991.

[Coq96] Thierry Coquand. An algorithm for type-checking dependent types. *Science of Computer Programming*, 26(1):167–177, 1996.

[Coq13] Thierry Coquand. Presheaf model of type theory. Unpublished note, 2013.

[Coq17] Thierry Coquand. Universe of Bishop sets, February 2017.

[Coq19] Thierry Coquand. Canonicity and normalization for dependent type theory. *Theoretical Computer Science*, 777:184–191, 2019. In memory of Maurice Nivat, a founding father of Theoretical Computer Science - Part I.

[Cra98] Karl Crary. *Type–Theoretic Methodology for Practical Programming Languages*. PhD thesis, Cornell University, Ithaca, NY, August 1998.

[CZ84] Robert L. Constable and Daniel R. Zlatin. The type theory of PL/CV3. *ACM Trans. Program. Lang. Syst.*, 6(1):94–117, January 1984.

[Dag13] Pierre-Évariste Dagand. *A Cosmology of Datatypes: Reusability and Dependent Types*. PhD thesis, University of Strathclyde, Glasgow, Scotland, August 2013.

[FS99] Marcelo Fiore and Alex Simpson. Lambda definability with sums via Grothendieck logical relations. In Jean-Yves Girard, editor, *Typed Lambda Calculi and Applications*, pages 147–161, Berlin, Heidelberg, 1999. Springer Berlin Heidelberg.

[GCST19] Gaëtan Gilbert, Jesper Cockx, Matthieu Sozeau, and Nicolas Tabareau. Definitional Proof-Irrelevance without K. *Proceedings of the ACM on Programming Languages*, pages 1–28, January 2019.

[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.

[Gir72] Jean-Yves Girard. *Interprétation fonctionelle et élimination des coupures de l'arithmétique d'ordre supérieur*. PhD thesis, Université Paris VII, 1972.

[GKNB20] Daniel Gratzer, G.A. Kavvos, Andreas Nuyts, and Lars Birkedal. Multimodal dependent type theory. Unpublished draft, 2020.

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

[GSB19] Daniel Gratzer, Jonathan Sterling, and Lars Birkedal. Implementing a modal dependent type theory. *Proceedings of the ACM on Programming Languages*, 3(ICFP):107:1–107:29, July 2019.

[Har92] Robert Harper. Constructing type systems over an operational semantics. *Journal of Symbolic Computation*, 14(1):71–84, July 1992.

[HHP93] Robert Harper, Furio Honsell, and Gordon Plotkin. A framework for defining logics. *J. ACM*, 40(1):143–184, January 1993.

[Hic01] Jason J. Hickey. *The MetaPRL Logical Programming Environment*. PhD thesis, Cornell University, Ithaca, NY, January 2001.

[HM95] Robert Harper and Greg Morrisett. Compiling polymorphism using intensional type analysis. In *Proceedings of the 22nd ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, pages 130–141, San Francisco, California, USA, 1995. Association for Computing Machinery.

[Hof95] Martin Hofmann. *Extensional concepts in intensional type theory*. PhD thesis, University of Edinburgh, Edinburgh, January 1995.

[Hof97] Martin Hofmann. Syntax and semantics of dependent types. In *Semantics and Logics of Computation*, pages 79–130. Cambridge University Press, 1997.

[How89] Douglas J. Howe. Equality in lazy computation systems. In *Proceedings of Fourth IEEE Symposium on Logic in Computer Science*, pages 198–203, New York, 1989. IEEE Computer Society.

[HP05] Robert Harper and Frank Pfenning. On equivalence and canonical forms in the LF type theory. *ACM Trans. Comput. Logic*, 6(1):61–101, January 2005.

[HS97] Martin Hofmann and Thomas Streicher. Lifting Grothendieck universes. Unpublished note, 1997.

[HS98] Martin Hofmann and Thomas Streicher. The groupoid interpretation of type theory. In *Twenty-five years of constructive type theory (Venice, 1995)*, volume 36 of *Oxford Logic Guides*, pages 83–111. Oxford Univ. Press, New York, 1998.

[Hub18] Simon Huber. Canonicity for cubical type theory. *Journal of Automated Reasoning*, June 2018.

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

[Joy17] Andre Joyal. Notes on clans and tribes. 2017.

[JT93] Achim Jung and Jerzy Tiuryn. A new characterization of lambda definability. In Marc Bezem and Jan Friso Groote, editors, *Typed Lambda Calculi and Applications*, pages 245–257, Berlin, Heidelberg, 1993. Springer Berlin Heidelberg.

[KHS19] Ambrus Kaposi, Simon Huber, and Christian Sattler. Gluing for type theory. In Herman Geuvers, editor, *4th International Conference on Formal Structures for Computation and Deduction (FSCD 2019)*, volume 131 of *Leibniz International Proceedings in Informatics (LIPIcs)*, Dagstuhl, Germany, 2019. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

[KKA19] Ambrus Kaposi, András Kovács, and Thorsten Altenkirch. Constructing quotient inductive-inductive types. *Proc. ACM Program. Lang.*, 3(POPL):2:1–2:24, January 2019.

[KL16] Chris Kapulkin and Peter LeFanu Lumsdaine. The simplicial model of Univalent Foundations (after Voevodsky). Preprint, June 2016.

[KS19] Chris Kapulkin and Christian Sattler. Homotopy canonicity of homotopy type theory, August 2019. Slides from a talk given at the International Conference on Homotopy Type Theory (HoTT 2019).

[Law63] F. William Lawvere. *Functorial Semantics of Algebraic Theories*. PhD thesis, Columbia University, 1963.

[Lum10] Peter LeFanu Lumsdaine. Weak omega-categories from intensional type theory. *Logical Methods in Computer Science*, Volume 6, Issue 3, September 2010.

[Lur09] Jacob Lurie. *Higher Topos Theory*. Princeton University Press, 2009.

[McB99] Conor McBride. *Dependently typed functional programs and their proofs*. PhD thesis, University of Edinburgh, 1999.

[McB10] Conor McBride. Quotients, 2010.

[ML75a] Per Martin-Löf. About models for intuitionistic type theories and the notion of definitional equality. In Stig Kanger, editor, *Proceedings of the Third Scandinavian Logic Symposium*, volume 82 of *Studies in Logic and the Foundations of Mathematics*, pages 81–109. Elsevier, 1975.

[ML75b] Per Martin-Löf. An intuitionistic theory of types: Predicative part. In H. E. Rose and J. C. Shepherdson, editors, *Logic Colloquium '73*, volume 80 of *Studies in Logic and the Foundations of Mathematics*, pages 73–118. Elsevier, 1975.

[ML79] Per Martin-Löf. Constructive mathematics and computer programming. In *6th International Congress for Logic, Methodology and Philosophy of Science*, pages 153–175, Hanover, August 1979. Published by North Holland, Amsterdam. 1982.

[ML84] Per Martin-Löf. *Intuitionistic type theory*, volume 1 of *Studies in Proof Theory*. Bibliopolis, 1984.

[ML87] Per Martin-Löf. Truth of a proposition, evidence of a judgement, validity of a proof. *Synthese*, 73(3):407–420, 1987.

[ML98] Saunders Mac Lane. *Categories for the Working Mathematician*. Springer-Verlag New York, 2nd edition, 1998.

[MS93] John C. Mitchell and Andre Scedrov. Notes on sconing and relators. In E. Börger, G. Jäger, H. Kleine Büning, S. Martini, and M. M. Richter, editors, *Computer Science Logic*, pages 352–378, Berlin, Heidelberg, 1993. Springer Berlin Heidelberg.

[New18] Clive Newstead. *Algebraic Models of Dependent Type Theory*. PhD thesis, Carnegie Mellon University, 2018.

[Nog02] Aleksey Nogin. Quotient types: A modular approach. In Victor A. Carreño, César A. Muñoz, and Sofiène Tahar, editors, *Theorem Proving in Higher Order Logics*, pages 263–280, Berlin, Heidelberg, 2002. Springer Berlin Heidelberg.

[NPS90] Bengt Nordström, Kent Peterson, and Jan M. Smith. *Programming in Martin-Löf's Type Theory*, volume 7 of *International Series of Monographs on Computer Science*. Oxford University Press, NY, 1990.

[RB16] Vincent Rahli and Mark Bickford. A nominal exploration of Intuitionism. In *Proceedings of the 5th ACM SIGPLAN Conference on Certified Programs and Proofs*, CPP 2016, pages 130–141, St. Petersburg, FL, USA, 2016. ACM.

[Rie19] Emily Riehl. The equivariant uniform kan fibration model of cubical homotopy type theory, 2019. A talk presented at the First International Conference on Homotopy Type Theory (HoTT 2019).

[RS17] Emily Riehl and Michael Shulman. A type theory for synthetic ∞-categories. *Higher Structures*, 1, 2017.

[SA20] Jonathan Sterling and Carlo Angiuli. Gluing models of type theory along flat functors. Unpublished draft, 2020.

[SAG19] Jonathan Sterling, Carlo Angiuli, and Daniel Gratzer. Cubical syntax for reflection-free extensional equality. In Herman Geuvers, editor, *4th International Conference on Formal Structures for Computation and Deduction (FSCD 2019)*, volume 131 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 31:1–31:25, Dagstuhl, Germany, 2019. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

[SH00] Christopher A. Stone and Robert Harper. Deciding type equivalence in a language with singleton kinds. In *Proceedings of the 27th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, pages 214–227, Boston, MA, USA, 2000. Association for Computing Machinery.

[SH06] Christopher A. Stone and Robert Harper. Extensional equivalence and singleton types. *ACM Transactions on Computational Logic*, 2006.

[Shu15] Michael Shulman. Univalence for inverse diagrams and homotopy canonicity. *Mathematical Structures in Computer Science*, 25(5):1203–1277, 2015.

[Str94] Thomas Streicher. Investigations into intensional type theory. Habilitationsschrift, Universität München, 1994.

[Str05] Thomas Streicher. Universes in toposes. In Laura Crosilla and Peter Schuster, editors, *From Sets and Types to Topology and Analysis: Towards practical foundations for constructive mathematics*, volume 48 of *Oxford Logical Guides*, pages 78–90. Oxford University Press, Oxford, 2005.

[Str14] Thomas Streicher. Semantics of type theory formulated in terms of representability, February 2014.

[SW15] Vilhelm Sjöberg and Stephanie Weirich. Programming up to congruence. In *POPL '15: Proceedings of the 42nd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, pages 369–382, Mumbai, India, 2015. ACM.

[Tai67] W. W. Tait. Intensional Interpretations of Functionals of Finite Type I. *The Journal of Symbolic Logic*, 32(2):198–212, 1967.

[Tay99] Paul Taylor. *Practical Foundations of Mathematics*. Cambridge studies in advanced mathematics. Cambridge University Press, Cambridge, New York (N. Y.), Melbourne, 1999.

[Tv88] Anne Troelstra and van Dalen, Dirk. *Constructivism in Mathematics: An Introduction*, volume 1 of *Studies in logic and the foundations of mathematics*. North-Holland, Amsterdam, New-York, Oxford, 1988.

[Uem17] Taichi Uemura. Fibred fibration categories. In *Proceedings of the 32nd Annual ACM/IEEE Symposium on Logic in Computer Science*, pages 24:1–24:12, Reykjavik, Iceland, 2017. IEEE Press.

[Uem19] Taichi Uemura. A general framework for the semantics of type theory. 2019.

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

[vdBG11] Benno van den Berg and Richard Garner. Types are weak ω-groupoids. *Proceedings of the London Mathematical Society*, 102(2):370–394, 2011.

[VMA19] Andrea Vezzosi, Anders Mörtberg, and Andreas Abel. Cubical Agda: A Dependently Typed Programming Language with Univalence and Higher Inductive Types. In *Proceedings of the 24th ACM SIGPLAN International Conference on Functional Programming*, ICFP '19, Boston, Massachusetts, USA, 2019. ACM.

[Voe06] Vladimir Voevodsky. A very short note on homotopy λ-calculus. *Unpublished*, September 2006.

[Voe10] Vladimir Voevodsky. The equivalence axiom and univalence models of type theory, February 2010. Talk at CMU.

[Voe16] Vladimir Voevodsky. Mathematical theory of type theories and the initiality conjecture, April 2016. Research proposal to the Templeton Foundation for 2016-2019, project description.