Normalization for Cubical Type Theory

Jonathan Sterling
Computer Science Department
Carnegie Mellon University

Carlo Angiuli
Computer Science Department
Carnegie Mellon University

***Abstract*—We prove normalization for (univalent, Cartesian) cubical type theory, closing the last major open problem in the syntactic metatheory of cubical type theory. Our normalization result is *reduction-free*, in the sense of yielding an injective function from equivalence classes of terms in context to a tractable language of β/η-normal forms. As corollaries we obtain both decidability of judgmental equality as well as the injectivity of type constructors in judgmentally consistent contexts.**

I. Introduction

De Morgan [19] and Cartesian [8, 9] cubical type theory are recent extensions of Martin-Löf type theory which provide constructive formulations of higher inductive types and Voevodsky's univalence axiom; unlike homotopy type theory [64], both enjoy *canonicity*, the property that closed terms of base type are judgmentally equal to constructors [8, 34].

Several proof assistants already implement cubical type theory, most notably Cubical Agda [65] (for the De Morgan variant) and `redtt` [49] (for Cartesian). Like most type-theoretic proof assistants, both typecheck terms using algorithms inspired by *normalization by evaluation* [1], which interleave evaluation and decomposition of types. The correctness of these algorithms hinges not on canonicity but on *normalization* theorems characterizing judgmental equivalence classes of *open* terms—and consequences thereof, such as decidability of equality and injectivity of type constructors. But unlike canonicity, normalization and its corollaries have until now remained conjectures for cubical type theory.

We contribute the first normalization proof for Cartesian cubical type theory [9]. By relying on recent advances in the metatheory of type theory, our proof is significantly more abstract and concise than existing canonicity proofs for cubical type theory; moreover, it can be adapted to De Morgan cubical type theory without conceptual changes.

A. *Cubical type theory and synthetic semantics*

Cubical type theory extends type theory with a number of features centered around a primitive interval $\mathbb{I}$ with elements $0, 1 : \mathbb{I}$. Propositional equality is captured by a path type $\mathsf{path}(A, a_0, a_1)$ whose elements are functions $f : \prod_{i:\mathbb{I}} A(i)$ satisfying $f(0) = a_0$ and $f(1) = a_1$ judgmentally. Congruence of paths follows from substitution; the remaining properties of equality are defined at each type by the *Kan operations* of coercion and box filling (or composition).

In Cartesian cubical type theory, coercion is a function

$$\mathsf{coe} : \textstyle\prod_{A:\mathbb{I}\to\mathcal{U}} \prod_{r,s:\mathbb{I}} A(r) \to A(s)$$

satisfying $\mathsf{coe}(A, r, r, a) = a$ for all $A : \mathbb{I} \to \boldsymbol{\mathcal{U}}$ and $a : A(r)$,[1] and additional equations for each particular connective, e.g.:

$$\begin{aligned} &\mathsf{coe}\big(\lambda i.\big(\textstyle\prod_{x:A(i)} B(i,x)\big), r, s, f\big) \\ &= \lambda x.\mathsf{coe}(\lambda i.B(i, \mathsf{coe}(A, s, i, x)), r, s, f(\mathsf{coe}(A, s, r, x))) \end{aligned}$$

The equations governing coercion and composition are complex, especially for the glue type which justifies univalence; calculating (and verifying the well-definedness of) these equations was a major obstacle in early work on cubical type theory. Orton and Pitts [44, 47] streamlined this process by observing that the model construction for De Morgan cubical type theory—the most technical part of which is these equations—can be carried out *synthetically* in the internal extensional type theory of any topos satisfying nine axioms (e.g., whose objects are De Morgan cubical sets); Angiuli et al. [9] establish an analogous result for the Cartesian variant.

A subtle aspect of these models is that coercion is defined not on types $A : \boldsymbol{\mathcal{U}}$ but on type families $A : \mathbb{I} \to \boldsymbol{\mathcal{U}}$; consequently, a semantic universe $\mathscr{U}$ of types-with-coercion must strangely admit a coercion structure for every figure $\mathbb{I} \to \mathscr{U}$. Licata et al. [44] obtain $\mathscr{U}$ by transposing the coercion map across the right adjoint to exponentiation by $\mathbb{I}$ given by the tininess of $\mathbb{I}$.

The synthetic approach of Orton and Pitts simplifies and clarifies the model construction of cubical type theory by factoring out naturality obligations, in much the same way that homotopy type theory provides a "synthetic homotopy theory" that factors out e.g., continuity obligations.

In this paper, we combine ideas of Orton and Pitts with the *synthetic Tait computability* (STC) theory of Sterling and Harper [56], which factors out bureaucratic aspects of syntactic metatheory. In STC, one considers an extensional type theory whose types are (proof-relevant) logical relations; the underlying syntax is exposed via a proof-irrelevant proposition syn under which the syntactic part of a logical relation is projected.

B. *Canonicity for cubical type theory*

Traditional canonicity proofs fix an evaluation strategy for closed terms, and associate to each closed type a proof-irrelevant *computability predicate* or *logical relation* ranging over closed terms of that type. Then, one ensures that evaluation is contained within judgmental equality, that computability is closed under evaluation, and that computability at base

[1] In De Morgan cubical type theory, coercion is limited to $A(0) \to A(1)$, a restriction counterbalanced by the additional (De Morgan) structure on $\mathbb{I}$.

type implies evaluation to a constructor; canonicity follows by proving that every well-typed closed term is computable.

Whereas evaluation in ordinary type theory need not descend under binders, evaluating a closed coercion $\mathsf{coe}(\lambda i.A, r, s, a)$ in cubical type theory requires determining the head constructor of (i.e., evaluating) the type A in context $i : \mathbb{I}$. Accordingly, the cubical canonicity proofs of Angiuli et al. [7, 8], Huber [34] define evaluation and computability for terms in context $(i_1 : \mathbb{I}, \ldots, i_n : \mathbb{I})$. In both proofs, the difficulty arises that typing and thus computability are closed under substitutions of the form $\mathbb{I}^n \longrightarrow \mathbb{I}^m$, but evaluation is not; both resolve the issue by showing evaluation is closed under computability up to judgmental equality.

C. *Semantic and proof-relevant computability*

The past several years have witnessed an explosion in *semantic* computability techniques for establishing syntactic metatheorems [4, 20, 22, 26, 39, 52, 56, 57, 62]. What makes semantic computability different from "free-hand" computability is that it is expressed as a *gluing model*, parameterized in the generic model of the type theory; hence one is always working with typed terms up to judgmental equality.

A new feature of semantic computability, forced in many cases by the absence of raw terms, is that a term may be computable in more than one way. This *proof-relevance* plays an important role in the normalization arguments of Altenkirch et al. [4], Coquand [20], Fiore [26] as well as the canonicity arguments of Coquand [20], Sterling et al. [57]. The proof-relevant approach is significantly simpler to set up than the alternative and it provides a compositional account of computability for universe hierarchies, which had been the main difficulty in conventional free-hand arguments.

Example 1. A semantic canonicity argument for ordinary type theory associates to each closed type a *computability structure* which assigns to each equivalence class of closed terms of that type a set of "computability proofs." We choose the computability structure at base type to be a collection of "codes" for each constant; then, by exhibiting a choice of computability proof for every well-typed term, we conclude that every equivalence class at base type is a constant.

Crucially, the ability to store data within the computability proofs circumvents the need to define a subequational evaluation function, allowing us to carry out the entire argument over equivalence classes of terms; rather than choosing a representative of each equivalence class, we encode canonical forms as a *structure* indexed over equivalence classes of terms.

1) In what contexts do we compute?: Semantic computability arguments have already been used to establish ordinary canonicity, cubical canonicity, and ordinary normalization. The key difference between these arguments lies in what is considered an "element" of a type, or more precisely, what are the contexts (and substitutions) of interest. In ordinary canonicity proofs, the only context of interest is the closed context, in which each type has just a *set* of elements; the computability structures are thus families of sets.

Following Huber and Angiuli, Favonia, and Harper [8, 34], cubical canonicity proofs must consider terms in all contexts $\mathbb{I}^n$, with all substitutions $\mathbb{I}^n \longrightarrow \mathbb{I}^m$ between them. These contexts and substitutions induce a *cubical set* (i.e., a presheaf) of elements of each type; the computability structures in question are thus families of cubical sets indexed in the application of a "cubical nerve" applied to a syntactical object, an arrangement suggested by Awodey in 2015. Notably, because semantic computability arguments are not evaluation-based, cubical canonicity proofs in this style (e.g., that of Sterling, Angiuli, and Gratzer [57]) entirely sidestep the evaluation coherence difficulties of prior work [8, 34].

The passage to presheaves of elements is not a novel feature of cubical canonicity; it appears already in normalization proofs for ordinary type theory, in the guise of Kripke logical relations of varying arities [37]. Because normalization is by definition a characterization of open terms, one must necessarily consider the presheaf of elements of a type relative to *all* contexts; but in light of the fact that normal forms are not closed under substitutions of terms for variables, one considers only a restricted class of substitutions (e.g., weakenings, injective renamings, or all variable renamings).

Following Tait [60], the normal forms of type theory are defined mutually as the *neutral forms* $\mathsf{ne}(A)$ (variables and eliminations thereof) and the *normal forms* $\mathsf{nf}(A)$ (constants and constructors applied to normals) of each open type A. Proof-irrelevant normalization arguments then establish that every neutral term is computable (via *reflection* $\uparrow_A$), and that every computable term has a normal form (via *reification* $\downarrow_A$). Proof-relevant normalization arguments follow the same yoga of reflection and reification, except that we speak not of the subset of neutral terms but rather the collection of neutrals and normals encoding each equivalence class of terms [20].

2) Related work on gluing for type theory: The past forty years have brought a steady stream of research developing the *gluing* perspective on logical relations [4, 23, 26, 27, 59]; however, only in the past several years has our understanding of the syntax and semantics of dependent types [3, 12, 28, 45, 63] caught up with the mathematical tools required to advance a truly objective metatheory of dependent type theory.

In particular, Coquand's analysis [20] of proof-relevant canonicity and normalization arguments for dependent type theory in terms of categorical gluing was the catalyst for a number of recent works that obtain non-trivial metatheorems for dependent type theory by semantic means, although some years earlier Shulman [52] had already used gluing to prove a homotopy canonicity result for a univalent type theory.

Uemura [62] proved a general gluing theorem for certain dependent type theories in the language of Shulman's type theoretic fibration categories; Kaposi et al. [39] proved a similar result in the language of categories with families. Coquand et al. [22] employed gluing to prove a homotopy canonicity result for a version of cubical type theory that omits certain computation rules, and Kapulkin and Sattler [40] used gluing to prove homotopy canonicity for homotopy type theory (as famously conjectured by Voevodsky). Sterling et al. [57] adapted Coquand's gluing argument to prove the first non-operational strict canonicity result for a cubical type theory. Gratzer et al. [29] used gluing to prove canonicity for a general *multi-modal* dependent type theory. Sterling and Harper [56] employ a different gluing argument to establish a proof-relevant generalization of the Reynolds Abstraction Theorem for a calculus of ML modules. Gratzer and Sterling [28] used gluing to establish the conservativity of higher-order judgments for dependent type theories.

3) What are the neutrals of cubical type theory?: Today's obstacles to proving cubical normalization are entirely different from the obstacles faced in the first proofs of cubical canonicity [8, 34]. As we have already discussed, coherence of evaluation is a non-issue for semantic computability; moreover, as normalization already descends under binders, this feature of coercion poses no additional difficulty.

However, cubical type theory includes a number of open judgmental equalities that challenge the yoga of reflection and reification. Consider the rule that applying any element, even a variable, of type $\mathsf{path}(A, a_0, a_1)$ to $0 : \mathbb{I}$ (resp., $1 : \mathbb{I}$) equals a_0 (resp., a_1). Whereas application of neutrals (e.g., variables) to normal forms (e.g., constants) is typically irreducible, here path application of a variable to a constant (but *not* to a variable) is a redex which may uncover further redexes:

$$x : \mathsf{path}(\lambda\_.\mathbb{N} \to \mathbb{N}, \mathsf{fib}, \mathsf{fib}) \vdash x(0)(7) = 13 : \mathbb{N}$$

One might imagine defining the normal form of path application by a case split on the elements of $\mathbb{I}$ (sending $0, 1 : \mathbb{I}$ to the normal form of $\mathsf{fib}(7)$, and $i : \mathbb{I}$ to a neutral application), but such a case split requires us to model $\mathbb{I}$ as a coproduct, which will not be tiny, preventing us from obtaining a universe of Kan types following Licata et al. [44].

Similar issues arise with a number of equations in both Cartesian and De Morgan cubical type theory; in fact, the Cartesian variant is *a priori* more challenging in this regard because contraction of interval variables $i, j : \mathbb{I}$ may also induce computation (e.g., in $\mathsf{coe}(A, i, j, a)$).

In this paper, we index neutrals $\mathsf{ne}_\phi(A)$ by a proposition ϕ representing their *locus of instability*, or where they cease to be neutral. For example, path application sends a ϕ-unstable neutral of type $\mathsf{path}(A, a_0, a_1)$ and an element $r : \mathbb{I}$ to a $(\phi \vee (r = 0) \vee (r = 1))$-unstable neutral of type $A(r)$. Reflection $\Uparrow_A^\phi$ operates on *stabilized neutrals*, pairs of a neutral $\mathsf{ne}_\phi(A)$ with a proof that the term is computable under ϕ. In the case of path application $x(r)$, one must provide computability proofs for a_0, a_1 under the assumption $r = 0 \vee r = 1$.

Terms in the ordinary fragment are never unstable (hence $\phi = \bot$), in which case a stabilized neutral is a neutral in the ordinary sense; "neutrals" with cubical redexes (such as $x(0)$) have $\phi = \top$, in which case their stabilized neutral is just a computability proof (and $\Uparrow_A^\top$ is the identity). To our knowledge, this is the first time that computability data appears in the domain of the reflection operation.

D. Contributions

We establish the normalization theorem (Theorem 42) for Cartesian cubical type theory closed under Π, Σ, path, glue, and a higher inductive circle type, using a cubical extension of synthetic Tait computability [56]; the new idea on which our argument hinges is the concept of *stabilized neutrals* described above. As corollaries to our main result, we obtain the admissible injectivity of type constructors (Theorem 43) as well as an algorithm to decide judgmental equality (Theorem 51).

The present paper does not describe universes or the modifications necessary to prove normalization for De Morgan cubical type theory; but note that univalence can be stated without universes, as we have done here. The novel aspects of cumulative, univalent universes are already supported because of the tininess of the interval and the account of glue types; the difference is that the operator projecting a normal type from a normalization structure of size α must be generalized over $\beta \geq \alpha$. Our argument carries over *mutatis mutandis* to a normalization proof for De Morgan cubical type theory.

In Section II we discuss the syntax of Cartesian cubical type theory and its situation within a dependently sorted logical framework. In Section III, we axiomatize a cubical version of synthetic Tait computability (STC) [56], a modal type theory of synthetic logical relations suitable for proving syntactic metatheorems; we construct in cubical STC a "normalization model" of cubical type theory displayed over the generic model. In Section IV we construct a topos model of cubical STC, which takes us the remaining distance to the main results of this paper, which are described in Section V.

II. Cartesian cubical type theory

We define the subject of our normalization theorem, intensional Cartesian cubical type theory, as a locally Cartesian closed category of judgments $\mathscr{T}_\square$ generated by the signature in Figs. 1 and 2. Readers may consult [9] and [6, Appendix B] for further exposition, including rule-based presentations.

A. LCCCs as a logical framework

The primary aspects of a type theory are its judgments A type and $a : A$ and their derivations, many of which require hypothetical judgments (e.g., $\lambda x.b : A \to B$ when $x : A \vdash b : B$). One typically restricts which judgments may be hypothesized, allowing $(x : A)$ but not (X type), judgmental equalities $(a = b : A)$, or hypothetical judgments. These restrictions, realized by the notion of *context*, are crucial to the syntactic metatheorems on which implementations rely; for example, decidability of equality requires that intensional type theory lacks a context former (isomorphic to) $\langle \Gamma, a = b : A \rangle$.

Both the rule-based presentations and the common categorical semantics of type theory—including categories with families [25], natural models [12], and Uemura's recent generalization thereof known as representable map categories [63]—include a notion of context as part of the definition of a type theory, and require these contexts to be preserved by models and their homomorphisms.

$$
\begin{aligned}
\mathbb{I}, \mathbb{F}, \mathsf{tp} &: \mathbf{jdg} \\
[-] &: \mathbb{F} \to \mathbf{jdg} \\
\mathsf{tm} &: \mathsf{tp} \to \mathbf{jdg} \\
0, 1 &: \mathbb{I} \\
(=) &: (\mathbb{I} \times \mathbb{I}) \to \mathbb{F} \\
(\wedge_{\mathbb{F}}), (\vee_{\mathbb{F}}) &: (\mathbb{F} \times \mathbb{F}) \to \mathbb{F} \\
(\forall_{\mathbb{I}}) &: (\mathbb{I} \to \mathbb{F}) \to \mathbb{F}
\end{aligned}
$$

$$
\begin{aligned}
\_ &: \{\phi\} \textstyle\prod_{p,q:[\phi]} p =_{[\phi]} q \\
\_ &: \{\phi, \psi\} \, ([\phi] \cong [\psi]) \cong (\phi =_{\mathbb{F}} \psi) \\
\_ &: \{\phi\} \, \left(\textstyle\prod_{i:\mathbb{I}} [\phi(i)]\right) \cong [\forall_{\mathbb{I}} \phi] \\
\_ &: \{r, s\} \, (r =_{\mathbb{I}} s) \cong [r = s] \\
\_ &: \{\phi, \psi\} \, ([\phi] \times [\psi]) \cong [\phi \wedge_{\mathbb{F}} \psi] \\
\_ &: \{\phi_0, \phi_1\} \, [\phi_i] \to [\phi_0 \vee_{\mathbb{F}} \phi_1]
\end{aligned}
$$

For each judgment $\mathbf{J} \in \{\mathbb{I}, \mathbb{F}, \mathsf{tp}, [\phi], \mathsf{tm}(A)\}$:

$$
\begin{aligned}
\mathsf{abort}_{\mathbf{J}} &: [0 = 1] \to \mathbf{1} \cong \mathbf{J} \\
\mathsf{split}_{\mathbf{J}} &: \{\phi, \psi\} \textstyle\prod_{x_\phi:[\phi]\to\mathbf{J}} \prod_{x_\psi:[\psi]\to\{\mathbf{J} \mid \phi \hookrightarrow x_\phi\}} [\phi \vee_{\mathbb{F}} \psi] \to \mathbf{J} \\
\_ &: \{\phi, \psi\} \textstyle\prod_{\_:[\phi]} \mathsf{split}_{\mathbf{J}}(x_\phi, x_\psi) =_{\mathbf{J}} x_\phi \\
\_ &: \{\phi, \psi\} \textstyle\prod_{\_:[\psi]} \mathsf{split}_{\mathbf{J}}(x_\phi, x_\psi) =_{\mathbf{J}} x_\psi \\
\_ &: \{\phi, \psi, x\} \, x =_{\mathbf{J}} \mathsf{split}_{\mathbf{J}}(x, x)
\end{aligned}
$$

$$
\begin{aligned}
\{\mathbf{J} \mid \phi \hookrightarrow x_\phi\} &:= \textstyle\sum_{x:\mathbf{J}} \prod_{p:[\phi]} x =_{\mathbf{J}} x_\phi(p) \\
\bot_{\mathbb{F}} &:= (0 = 1) \\
\partial i &:= (i = 0) \vee_{\mathbb{F}} (i = 1) \\
[]_{\mathbf{J}} &:= \mathsf{abort}_{\mathbf{J}} \\
[\phi \hookrightarrow x_\phi, \psi \hookrightarrow x_\psi]_{\mathbf{J}} &:= \mathsf{split}_{\mathbf{J}}(x_\phi, x_\psi)
\end{aligned}
$$

Fig. 1. Basic judgmental structure of Cartesian cubical type theory.

$$
\begin{aligned}
\mathsf{path} &: \left(\textstyle\sum_{A:\mathbb{I}\to\mathsf{tp}} \mathsf{tm}(A(0)) \times \mathsf{tm}(A(1))\right) \to \mathsf{tp} \\
\Pi, \Sigma &: \left(\textstyle\sum_{A:\mathsf{tp}} (\mathsf{tm}(A) \to \mathsf{tp})\right) \to \mathsf{tp} \\
\mathsf{glue} &: \textstyle\prod_{\phi:\mathbb{F}} \left\{ \left(\sum_{B:\mathsf{tp}} \sum_{A:[\phi]\to\mathsf{tp}} \prod_{\_:[\phi]} \mathsf{tm}(\mathsf{Equiv}(A, B))\right) \to \mathsf{tp} \,\middle|\, \phi \hookrightarrow \lambda(B, A, f).A \right\} \\
\mathsf{S1} &: \mathsf{tp} \\
\mathsf{path/tm} &: \{A, a_0, a_1\} \left(\textstyle\prod_{i:\mathbb{I}} \{\mathsf{tm}(A(i)) \mid \partial i \hookrightarrow [\overline{i = \epsilon \hookrightarrow a_\epsilon}]_{\mathsf{tm}(A(i))}\}\right) \cong \mathsf{tm}(\mathsf{path}(A, a_0, a_1)) \\
\Pi/\mathsf{tm} &: \{A, B\} \left(\textstyle\prod_{x:\mathsf{tm}(A)} \mathsf{tm}(B(x))\right) \cong \mathsf{tm}(\Pi(A, B)) \\
\Sigma/\mathsf{tm} &: \{A, B\} \left(\textstyle\sum_{x:\mathsf{tm}(A)} \mathsf{tm}(B(x))\right) \cong \mathsf{tm}(\Sigma(A, B)) \\
\mathsf{glue/tm} &: \{\phi, B, A, f\} \left\{ \left(\textstyle\sum_{a:\prod_{\_:[\phi]} \mathsf{tm}(A)} \{\mathsf{tm}(B) \mid \phi \hookrightarrow f(a)\}\right) \cong \mathsf{tm}(\mathsf{glue}(\phi, B, A, f)) \,\middle|\, \phi \hookrightarrow \lambda(a, b).a \right\} \\
\mathsf{base} &: \mathsf{tm}(\mathsf{S1}) \\
\mathsf{loop} &: \textstyle\prod_{i:\mathbb{I}} \{\mathsf{tm}(\mathsf{S1}) \mid \partial i \hookrightarrow \mathsf{base}\} \\
\mathsf{ind}_{\mathsf{S1}} &: \textstyle\prod_{C:\mathsf{tm}(\mathsf{S1})\to\mathsf{tp}} \prod_{b:\mathsf{tm}(C(\mathsf{base}))} \prod_{l:\prod_{i:\mathbb{I}} \{\mathsf{tm}(C(\mathsf{loop}(i))) \mid \partial i \hookrightarrow b\}} \prod_{x:\mathsf{tm}(\mathsf{S1})} \mathsf{tm}(C(x)) \\
\_ &: \{C, b, l\} \, \mathsf{ind}_{\mathsf{S1}}(C, b, l, \mathsf{base}) =_{\mathsf{tm}(C(\mathsf{base}))} b \\
\_ &: \{C, b, l, i\} \, \mathsf{ind}_{\mathsf{S1}}(C, b, l, \mathsf{loop}(i)) =_{\mathsf{tm}(C(\mathsf{loop}(i)))} l(i) \\
\mathsf{hcom} &: \textstyle\prod_{A:\mathsf{tp}} \prod_{r,s:\mathbb{I}} \prod_{\phi:\mathbb{F}} \prod_{a:\prod_{i:\mathbb{I}} \prod_{\_:[i=r\vee_{\mathbb{F}}\phi]} \mathsf{tm}(A)} \{\mathsf{tm}(A) \mid r = s \vee_{\mathbb{F}} \phi \hookrightarrow a(s)\} \\
\mathsf{coe} &: \textstyle\prod_{A:\mathbb{I}\to\mathsf{tp}} \prod_{r,s:\mathbb{I}} \prod_{a:\mathsf{tm}(A(r))} \{\mathsf{tm}(A(s)) \mid r = s \hookrightarrow a\}
\end{aligned}
$$

$$
\begin{aligned}
&\mathsf{isContr} : \mathsf{tp} \to \mathsf{tp} \\
&\mathsf{isContr} := \lambda A.\Sigma(A, \lambda x.\Pi(A, \lambda y.\mathsf{path}(\lambda\_.A, x, y)))
\end{aligned}
$$

$$
\begin{aligned}
&\mathsf{Equiv} : \mathsf{tp} \to \mathsf{tp} \to \mathsf{tp} \\
&\mathsf{Equiv} := \lambda A.\lambda B.\Sigma(\Pi(A, \lambda\_.B), \lambda f.\Pi(B, \lambda b.\mathsf{isContr}(\Sigma(A, \lambda a.\mathsf{path}(\lambda\_.B, f(a), b)))))
\end{aligned}
$$

$$
\begin{aligned}
&\mathsf{unglue} : \{\phi, B, A, f\} \, \mathsf{tm}(\mathsf{glue}(\phi, B, A, f)) \to \mathsf{tm}(B) \\
&\mathsf{unglue} := \lambda g.\pi_2(\mathsf{glue/tm}^{-1}(g))
\end{aligned}
$$

Fig. 2. Generating clauses in the signature for Cartesian cubical type theory pertaining to connectives. For space reasons, we omit the computation rules of the Kan operations for each connective, which can be found in [9].

In contrast, higher-order logical frameworks for defining type theories, such as Martin-Löf's logical framework [46] and the Edinburgh Logical Framework [31], elevate the *judgmental* structure of a type theory; then, as explicated by Harper et al. [31], one may impose after the fact a collection of LF contexts (or *worlds*) relative to which adequacy and other metatheorems hold [30]. These worlds, which can be seen to correspond roughly to the *arities* of Jung and Tiuryn [37], were subsequently implemented in the Twelf proof assistant [48] as "`%worlds` declarations."

In light of that perspective, we regard a notion of context as a structure placed on a locally Cartesian closed *category of judgments* $\mathscr{T}$ of a type theory, whose objects and morphisms are (equivalence classes of) judgments and deductions. The dependent products of $\mathscr{T}$ encode hypothetical judgments, and the finite limits both encode substitution and judgmental equality; a notion of context is often a full subcategory $\mathscr{C} \subseteq \mathscr{T}$ spanned by objects distinguished as contexts.

Example 2. The category of judgments $\mathscr{T}$ of Martin-Löf type theory without any types is the free LCCC generated by a single map $\mathsf{tm} \longrightarrow \mathsf{tp}$. The category of contexts $\mathscr{C} \subseteq \mathscr{T}$ is inductively defined as the full subcategory spanned by the terminal object and any fiber of $\mathsf{tm} \longrightarrow \mathsf{tp}$ over a context.

Equality is undecidable in $\mathscr{T}$ (as it has all finite limits), but *is* decidable for *terms and types in context*, objectified by the restricted Yoneda embedding $\mathscr{T} \longrightarrow \mathrm{Pr}(\mathscr{C})$ taking the judgment $\mathsf{tp} : \mathscr{T}$ to its "functor of context-valued points" $\mathrm{Hom}(\mathscr{C}, \mathsf{tp}) : \mathscr{C}^{\mathrm{op}} \longrightarrow \mathbf{Set}$. Proofs of decidability proceed by further restricting to a category of (contexts and) renamings $\mathscr{R}$ along the forgetful map $\mathscr{R} \longrightarrow \mathscr{C}$ [4, 20, 26]. □

A second distinction between LCCCs and (for instance) Uemura's framework is that the latter is stratified to ensure that the use of hypothetical judgment in a theory is strictly positive, whereas both LCCCs and syntactic logical frameworks place no restriction on hypothetical judgments and even allow binders of higher-level (e.g., in Martin-Löf's $\mathsf{funsplit}$ operator [46]). However, using an Artin gluing argument due to Paul Taylor [61], Gratzer and Sterling [28] have observed that the extension of a representable map category à la Uemura to a LCCC is conservative (in fact, fully faithful), ensuring the adequacy of LCCC encodings of type theories.

We therefore define the category of judgments of Cartesian cubical type theory as the LCCC $\mathscr{T}_\square$ generated by the signature in Figs. 1 and 2; then, locally Cartesian closed functors $\mathsf{M} : \mathscr{T}_\square \longrightarrow \mathscr{E}$ determine algebras for that signature valued in $\mathscr{E}$. Unlike homomorphisms of models of type theory, such functors preserve higher-order judgments; note however that we are proving a single theorem about the syntactic category $\mathscr{T}_\square$, not studying the model theory of cubical type theory.

B. The signature of Cartesian cubical type theory

In Fig. 1 we present the judgmental structure of cubical type theory; we inherit from Martin-Löf type theory the basic forms of judgment $\mathsf{tp} : \mathscr{T}_\square$ and $\mathsf{tm} : (\mathscr{T}_\square)_{/\mathsf{tp}}$ classifying types and terms respectively, and add three additional forms of judgment for cubical phenomena: $\mathbb{I} : \mathscr{T}_\square$ for elements of the interval, $\mathbb{F} : \mathscr{T}_\square$ for *cofibrations* (or *cofibrant propositions*), and $[-] : (\mathscr{T}_\square)_{/\mathbb{F}}$ for proofs of cofibrations. The standard notion of context is generated by $\mathbf{1}$ and context extension by $(x : A)$, $(i : \mathbb{I})$, and $(\_ : [\phi])$.[2] In this paper, we will only consider a more restricted notion of *atomic context* (Section IV-C) that plays a role analogous to the renamings of Example 2.

Cofibrations are strict propositions; the cofibration classifier $\mathbb{F}$ is strictly univalent and closed under $\wedge_\mathbb{F}$, $\vee_\mathbb{F}$, $\forall_\mathbb{I}$, and $=_\mathbb{I}$. We define $\wedge_\mathbb{F}$, $\forall_\mathbb{I}$, and $=_\mathbb{I}$ in terms of the same notions in $\mathscr{T}_\square$, but $\mathscr{T}_\square$ has no disjunction or empty type by which to define $\vee_\mathbb{F}$ or stipulate $(0 = 1) \to \bot$. Instead, we axiomatize these by eliminators $\mathsf{abort}_\mathbf{J}, \mathsf{split}_\mathbf{J}$ for each generating judgment $\mathbf{J}$.

The following notations in Fig. 1 are used pervasively throughout this paper. (At present, "propositions" are elements of $\mathbb{F}$; later they will be elements of a subobject classifier Ω.)

Notation 3 (Extent types [50])**.** Given a proposition ϕ and a *partial element* $a_\phi : [\phi] \to A$, we write $\{A \mid \phi \hookrightarrow a_\phi\}$ for the collection of elements $a : A$ that restrict to a_ϕ under the assumption of $z : [\phi]$. In other words:

$$\{A \mid \phi \hookrightarrow a_\phi\} := \{a : A \mid \forall z : [\phi]. a = a_\phi(z)\}$$

We will implicitly coerce elements of $\{A \mid \phi \hookrightarrow a_\phi\}$ to A.

Notation 4 (Systems [19])**.** Let ϕ, ψ be propositions. Under $\_ : [\phi \vee \psi]$, given a pair of partial elements $a_\phi : [\phi] \to A$ and $a_\psi : [\psi] \to A$ that agree when $\_ : [\phi \wedge \psi]$, we write

$$[\phi \hookrightarrow a_\phi, \psi \hookrightarrow a_\psi] : A$$

for the "case split" that extends a_ϕ, a_ψ. Likewise, under the assumption $\_ : [\bot]$, we write $[] : A$ for the unique element of A. We will leave abstraction and application over $z : \phi$ implicit; where it improves clarity, we may write the unary system $[\phi \hookrightarrow a]$ for $\lambda z : \phi.a$.

In Fig. 2 we define the connectives of cubical type theory. We specify the elements of Π, Σ, path, and glue by isomorphisms whose underlying functions encode introduction and elimination rules, and whose equations encode β and η rules; we will leave the first three of these isomorphisms (and the projection from equivalences to functions) implicit. The higher inductive circle $\mathsf{S1}$ has constructors base and loop and an eliminator $\mathsf{ind}_{\mathsf{S1}}$ with computation rules. Finally, we specify the Kan operations hcom and coe; for space reasons we do not reproduce the computation rules for hcom and coe in each type, which can be found in Angiuli et al. [9].

The signature for De Morgan cubical type theory [19, 21] differs only in the structure imposed on $\mathbb{I}$ and the types and computation rules of hcom and coe.

III. Synthetic Tait computability

In this section, we axiomatize a category $\mathscr{E}$ whose internal language provides a "type theory of proof-relevant logical relations" à la Sterling and Harper's synthetic Tait computability [56]. Inside that type theory, we then define

[2] Syntactic presentations typically write $\langle \Gamma, \phi \rangle$ for $\langle \Gamma, \_ : [\phi] \rangle$.

Axiom	Substantiation
Axiom STC-1	Lemma 26
Axiom STC-2	Construction 30 and Corollary 32
Axiom STC-3	Computation 27
Axiom STC-4	Remark 28 and Construction 30
Axiom STC-5	Lemmas 33, 62 and 63
Axiom STC-6	Construction 24 and Remark 29

Fig. 3. A dictionary between the axioms of Section III and their substantiations in the category of computability structures.

a reflection–reification computability model of cubical type theory from which we will derive normalization. We defer to Section IV an explicit construction of $\mathscr{E}$ as a category of computability structures; Fig. 3 provides forward references to our justifications of the axioms.

We begin by assuming that $\mathscr{E}$ satisfies Giraud's axioms [10]; all such categories interpret extensional Martin-Löf type theory extended by a strictly univalent universe Ω of all proof-irrelevant propositions. Next, we assume that $\mathscr{E}$ contains a cumulative hierarchy of universes whose elements satisfy a strictification axiom introduced by Orton and Pitts [47].

Definition 5. An *Orton–Pitts universe* is a type theoretic universe $\mathcal{U}$ strictly closed under dependent products, dependent sums, inductive types, quotients, and the subobject classifier, such that the following additional *strictification* axiom holds:

> Given $A : \mathcal{U}$, write $\mathsf{Iso}(A) := \sum_{B:\mathcal{U}}(A \cong B)$ for the type of $\mathcal{U}$-isomorphs of A. For any proposition $\phi : \Omega$, there is a section to the weakening map $\mathsf{Iso}(A) \longrightarrow (\phi \to \mathsf{Iso}(A))$.

Axiom STC-1. *There exists a cumulative hierarchy of Orton–Pitts universes $\mathcal{U}_0 \subseteq \mathcal{U}_1 \ldots$ in $\mathscr{E}$ such that every map in $\mathscr{E}$ is classified by some $\mathcal{U}_i$.*

We schematically write $\mathcal{U}, \mathcal{V}$ for arbitrary universes in the hierarchy specified by Axiom STC-1.

Axiom STC-2. *There exists a tiny [43, 67] interval object $\mathbb{I} : \mathcal{U}$ with two endpoints $0, 1 : \mathbb{I}$.*

What it means for the interval to be *tiny* is that the exponential functor $(\mathbb{I} \to -) : \mathscr{E} \longrightarrow \mathscr{E}$ has a right adjoint $(-)_{\mathbb{I}}$. Equivalently, the exponential functor preserves colimits.

A. Modalities for syntax and semantics

The central assumption of synthetic Tait computability is the existence of an uninterpreted proposition syn from which we will generate the modal syntax–semantics duality.

Axiom STC-3. *There exists a proposition $\mathsf{syn} : \Omega$.*

The proposition syn generates complementary *open* and *closed* lex idempotent modalities $\bigcirc, \bullet$ that we interpret as respectively projecting the syntactic and semantic aspects of a given computability structure. Because lex modalities descend to the slices in a fibered way, we can use $\bigcirc, \bullet$ naïvely in the internal language of $\mathscr{E}$ [14, 51]; however, we find it most convenient to begin by considering the universes $\mathcal{U}_\bigcirc, \mathcal{U}_\bullet$ of "syntactic" and "semantic" types.

1) Universe of syntactic types: Given a universe $\mathcal{U} : \mathcal{V}$, we define the universe $\mathcal{U}_\bigcirc : \mathcal{V}$ of syntactic types together with its (dependent) modality; the following definitions are justified by Orton–Pitts strictification (Axiom STC-1), setting $\phi := \mathsf{syn}$.

$$\mathcal{U}_\bigcirc : \{\mathcal{V} \mid \mathsf{syn} \hookrightarrow \mathcal{U}\} \qquad \bigcirc : \{\mathcal{U} \to \mathcal{U}_\bigcirc \mid \mathsf{syn} \hookrightarrow \lambda A.A\}$$
$$\mathcal{U}_\bigcirc \cong \mathsf{syn} \to \mathcal{U} \qquad \bigcirc \cong \lambda A.\lambda\_ : \mathsf{syn}.A$$
$$\mathsf{el}_\bigcirc : \{\mathcal{U}_\bigcirc \to \mathcal{U} \mid \mathsf{syn} \hookrightarrow \lambda A.A\}$$
$$\mathsf{el}_\bigcirc \cong \lambda A.\textstyle\prod_{z:\mathsf{syn}} A(z)$$

To see how strictification applies, observe that $\mathsf{syn} \to \mathcal{U}$ is an isomorph of $\mathcal{U}$ under the assumption $\_ : \mathsf{syn}$; we may therefore choose $\mathcal{U}_\bigcirc$ to be (totally) isomorphic to $(\mathsf{syn} \to \mathcal{U})$ and under $\_ : \mathsf{syn}$ strictly equal to $\mathcal{U}$. The remaining definitions go through directly given that $\mathcal{U}_\bigcirc$.

Because it causes no ambiguity, we will leave the decoding $\mathsf{el}_\bigcirc$ implicit in our notations; furthermore, we will leave both abstraction and application over syn implicit.

Our use of Orton–Pitts strictification above can be summed up in the following syntactic realignment lemma, the workhorse of synthetic Tait computability.

Corollary 6 (Syntactic realignment [55, 56, 58])**.** *Given $A : \mathcal{U}$, $A_\circ : \mathcal{U}_\bigcirc$, and an isomorphism $f : \bigcirc(A_\circ \cong A)$, we may define a strictly aligned type $f^*A : \{\mathcal{U} \mid \mathsf{syn} \hookrightarrow A_\circ\}$ and a strictly aligned isomorphism $f^\dagger : \{f^*A \cong A \mid \mathsf{syn} \hookrightarrow f\}$.*

Remark 7. The syntactic modality commutes with dependent products, dependent sums, equality, etc.

Then we axiomatize the existence of an algebra for the signature of Cartesian cubical type theory in $\mathscr{E}$. One can internalize as a dependent record the collection $\mathcal{T}_\square\text{-}\mathbf{Mod}(\mathcal{V})$ of $\mathcal{T}_\square$-algebras/models valued in types classified by any universe $\mathcal{V}$, writing $\mathsf{M.tp}$, $\mathsf{M.tm}$, etc. for each component. In Axiom STC-4 below, we require a $\mathcal{T}_\square$-model M valued in $\mathcal{U}_\bigcirc$, such that $\mathsf{M}.\mathbb{I}$ is the syntactic part of the interval of Axiom STC-2.

Axiom STC-4. *There exists a $\mathcal{T}_\square$-model $\mathsf{M} : \mathcal{T}_\square\text{-}\mathbf{Mod}(\mathcal{U}_\bigcirc)$ such that $\bigcirc(\mathsf{M}.\mathbb{I} = \mathbb{I})$.*

2) Universe of semantic types: A type $A : \mathcal{U}$ is called *semantic* (or *$\bigcirc$-connected*) when it has no syntactic component, i.e. we have an isomorphism $\mathbf{1} \cong \bigcirc A$. Using this idea as a prototype, we define a dual universe of semantic types:

$$\mathcal{U}_\bullet : \{\mathcal{V} \mid \mathsf{syn} \hookrightarrow \mathbf{1}\} \qquad \bullet : \mathcal{U} \to \mathcal{U}_\bullet \qquad \mathsf{el}_\bullet : \mathcal{U}_\bullet \to \mathcal{U}$$
$$\mathcal{U}_\bullet \cong \{\mathcal{U} \mid \mathsf{syn} \hookrightarrow \mathbf{1}\} \qquad \bullet \cong \lambda A.A \sqcup_{A \times \mathsf{syn}} \mathsf{syn} \qquad \mathsf{el}_\bullet \cong \lambda A.A$$

Above we are writing $A \sqcup_{A \times \mathsf{syn}} \mathsf{syn}$ for the pushout of the two product projections from $A \times \mathsf{syn}$.

The definitions of $\mathcal{U}_\bullet, \bullet, \mathsf{el}_\bullet$ are likewise justified by syntactic realignment (Corollary 6): fixing $\_ : \mathsf{syn}$ we note that each of the types above becomes a singleton, so it can be aligned to restrict to $\mathbf{1}$ strictly. As with the syntactic modality, we leave the decoding $\mathsf{el}_\bullet$ implicit.

Warning 8. The semantic modality commutes with dependent sums and equality, but not much else.

B. Cofibrations and locality

We construct the universe of cofibrations in two steps: first we define a universe of propositions $\mathbb{E} : \{\mathcal{U} \mid \mathsf{syn} \hookrightarrow \mathsf{M}.\mathbb{F}\}$, and then we constrain it to a subclass $\mathbb{F} \subseteq \mathbb{E}$ generated by equality of dimensions, disjunction, conjunction, and universal quantification over the interval. The purpose of this constraint is to support an external *algorithm* that decides equality underneath a cofibration (Theorem 51).

$$\mathbb{E} : \{\mathcal{U} \mid \mathsf{syn} \hookrightarrow \mathsf{M}.\mathbb{F}\}$$
$$\mathbb{E} \cong \textstyle\sum_{\phi:\mathsf{M}.\mathbb{F}}\{\Omega \mid \mathsf{syn} \hookrightarrow \mathsf{M}.[\phi]\}$$

It is trivial to close $\mathbb{E}$ under conjunction, equality of the interval, and universal quantification over the interval.

Notation 9. The canonical map $(- = \top_{\mathbb{E}}) : \mathbb{E} \to \Omega$ is a suitable decoding function; we treat it as an implicit coercion.

As in Fig. 1, the difficult part is to close $\mathbb{E}$ under disjunction and to enforce $0 \neq 1$; because $\mathsf{M}.\mathsf{split}_{\mathbf{J}}$ only eliminates into components of M, the "disjunction" $\mathsf{M}.(\vee_{\mathbb{E}})$ is not even a disjunction relative to types in $\mathcal{U}_{\bigcirc}$, much less in all of $\mathcal{U}$ (and similarly for $\mathsf{M}.\mathsf{abort}_{\mathbf{J}}$ and $\mathsf{M}.\bot_{\mathbb{F}}$).

Construction 10 (Disjunction)**.** We explicitly glue together the syntactic disjunction with the semantic disjunction; to ensure that the resulting proposition is aligned over the (weaker) syntactic disjunction, we place the semantic disjunction underneath the modality $\bullet$ to force it to become $\bigcirc$-connected.

$$(\vee_{\mathbb{E}}) : \{\mathbb{E} \times \mathbb{E} \to \mathbb{E} \mid \mathsf{syn} \hookrightarrow \mathsf{M}.(\vee_{\mathbb{F}})\}$$
$$\phi \vee_{\mathbb{E}} \psi = (\phi\, \mathsf{M}.\vee_{\mathbb{F}}\, \psi, \mathsf{M}.[\phi\, \mathsf{M}.\vee_{\mathbb{F}}\, \psi] \wedge \bullet([\phi] \vee [\psi]))$$

No realignment is required, because Ω is strictly univalent.

We may then define the universe of cofibrations $\mathbb{F} \subseteq \mathbb{E}$ to be the smallest subobject of $\mathbb{E}$ closed under the following rules:

$$\frac{z : \mathsf{syn} \qquad \phi : \mathbb{E}}{\phi \in \mathbb{F}} \qquad \frac{\phi \in \mathbb{F} \qquad \psi \in \mathbb{F}}{\phi \wedge_{\mathbb{E}} \psi \in \mathbb{F} \qquad \phi \vee_{\mathbb{E}} \psi \in \mathbb{F}}$$

$$\frac{\forall i : \mathbb{I}.(\phi(i) \in \mathbb{F})}{(\forall i.\phi(i)) \in \mathbb{F}} \qquad \frac{r, s : \mathbb{I}}{(r = s) \in \mathbb{F}}$$

To avoid confusion, we will write $\wedge_{\mathbb{F}}, \vee_{\mathbb{F}} : \mathbb{F} \times \mathbb{F} \to \mathbb{F}$ for the maps induced by the closure of $\mathbb{F}$ under $\vee_{\mathbb{E}}$. Likewise we define $\top_{\mathbb{F}} = (1 = 1)$ and $\bot_{\mathbb{F}} = (0 = 1)$ in $\mathbb{F}$. We observe that the universe of cofibrations can be aligned over $\mathsf{M}.\mathbb{F}$, i.e. we have $\mathbb{F} : \{\mathcal{U} \mid \mathsf{syn} \hookrightarrow \mathsf{M}.\mathbb{F}\}$; this follows from the fact that $(\phi \in \mathbb{F}) =_{\Omega} \top$ for any $\phi : \mathbb{E}$ assuming $z : \mathsf{syn}$.

We must define semantic conditions for types that are *local* with respect to $\vee_{\mathbb{F}}$ and $\bot_{\mathbb{F}}$, in the sense that they behave as though the positive cofibrations satisfy universal properties.

Definition 11. A type $A : \mathcal{U}$ is called $\bot_{\mathbb{F}}$*-connected* when it behaves as if $0 \neq 1$, i.e. we have $([\bot_{\mathbb{F}}] \to A) \cong \mathbf{1}$.

Definition 12. A type $A : \mathcal{U}$ is called $\mathbb{F}$*-local* when it is $\bot_{\mathbb{F}}$-connected and, for any two cofibrations $\phi, \psi : \mathbb{F}$, the type A is right-orthogonal to the canonical map $[\phi] \vee [\psi] \longrightarrow [\phi \vee_{\mathbb{F}} \psi]$ in the sense that the dotted map below always exists uniquely:

$$\begin{array}{ccc}
[\phi] \vee [\psi] & \xrightarrow{[\phi \hookrightarrow a_\phi,\, \psi \hookrightarrow a_\psi]} & A \\
\downarrow & \overset{[\phi \hookrightarrow a_\phi,\, \psi \hookrightarrow a_\psi]_A}{\nearrow} & \downarrow \\
[\phi \vee_{\mathbb{F}} \psi] & \longrightarrow & \mathbf{1}
\end{array}$$

A necessary condition for a type $A : \mathcal{U}$ being $\mathbb{F}$-local is that its syntactic part $\bigcirc A$ is $\mathbb{F}$-local. In Axiom STC-5 below, we assert that this condition is sufficient.

Axiom STC-5. *A type $A : \mathcal{U}$ is $\mathbb{F}$-local if and only if $\bigcirc A$ is $\mathbb{F}$-local.*

The interpretation of every syntactic sort of Cartesian cubical type theory in M can be seen to be $\mathbb{F}$-local. Therefore, by Axiom STC-5, any type A whose syntactic part $\bigcirc A$ is isomorphic to one of those sorts is automatically $\mathbb{F}$-local.

C. Kan computability structures

Definition 13. We define $\mathcal{U}_{\mathsf{tp}}$ to be the object of *computability structures*, which pair a syntactic type $A : \mathsf{M}.\mathsf{tp}$ with a total type aligned over its elements:

$$\mathcal{U}_{\mathsf{tp}} : \{\mathcal{V} \mid \mathsf{syn} \hookrightarrow \mathsf{M}.\mathsf{tp}\}$$
$$\mathcal{U}_{\mathsf{tp}} \cong \textstyle\sum_{A:\mathsf{M}.\mathsf{tp}}\{\mathcal{U} \mid \mathsf{syn} \hookrightarrow \mathsf{M}.\mathsf{tm}(A)\}$$

We leave the projection $\pi : \mathcal{U}_{\mathsf{tp}} \longrightarrow \mathcal{U}$ implicit in our notation.

Definition 14. A *homogeneous composition structure* on $A : \mathcal{U}_{\mathsf{tp}}$ is an element of the type $\mathsf{HCom}(A)$ defined in Fig. 4; such a structure asserts the existence of an operation hcom_A that is aligned over the existing syntactic homogeneous composition operation. We define by realignment a weak classifying object $\mathcal{U}^{\mathsf{hcom}}_{\mathsf{tp}}$ for computability structures equipped with a homogeneous composition structure:

$$\mathcal{U}^{\mathsf{hcom}}_{\mathsf{tp}} : \{\mathcal{V} \mid \mathsf{syn} \hookrightarrow \mathsf{M}.\mathsf{tp}\}$$
$$\mathcal{U}^{\mathsf{hcom}}_{\mathsf{tp}} \cong \textstyle\sum_{A:\mathcal{U}_{\mathsf{tp}}}\mathsf{HCom}(A)$$

We leave the projection $\mathcal{U}^{\mathsf{hcom}}_{\mathsf{tp}} \longrightarrow \mathcal{U}_{\mathsf{tp}}$ implicit.

Likewise, a *coercion structure* on a line of computability structures $A : \mathbb{I} \to \mathcal{U}_{\mathsf{tp}}$ is an element of the type $\mathsf{Coe}(A)$ also defined in Fig. 4; such a structure provides a coe_A operation aligned over the existing syntactic coercion operation.

Constructing a weak classifying object for coercion structures is more challenging; we use the method of Licata et al. [44], which relies crucially on the tininess of the interval.

Construction 15. Using the right adjoint $(-)_{\mathbb{I}}$ to $(\mathbb{I} \to -)$ given by Axiom STC-2, we transpose the map $\mathsf{Coe} : (\mathbb{I} \to \mathcal{U}^{\mathsf{hcom}}_{\mathsf{tp}}) \longrightarrow \mathcal{U}_{\bullet}$ to obtain $\mathsf{Coe}^{\sharp} : \mathcal{U}^{\mathsf{hcom}}_{\mathsf{tp}} \longrightarrow (\mathcal{U}_{\bullet})_{\mathbb{I}}$. Pulling back the "root" of the generic family $\dot{\mathcal{U}}_{\bullet} \longrightarrow \mathcal{U}_{\bullet}$ along this map, we obtain a weak classifying object $\mathcal{U}^{\mathsf{kan}}_{\mathsf{tp}}$ for

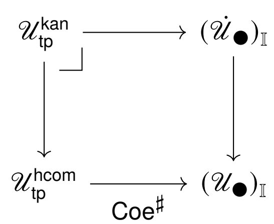
$$\begin{aligned}\mathsf{HCom} &: \mathscr{U}_{\mathsf{tp}} \longrightarrow \mathscr{U}_{\bullet}\\ \mathsf{Coe} &: (\mathbb{I} \to \mathscr{U}_{\mathsf{tp}}) \longrightarrow \mathscr{U}_{\bullet}\\ \mathsf{HCom}(A) &\cong \left\{\prod_{r,s:\mathbb{I}}\prod_{\phi:\mathbb{F}}\prod_{a:\prod_{i:\mathbb{I}}\prod_{\_:[i=r\vee_{\mathbb{F}}\phi]}A}\{A \mid r=s \vee_{\mathbb{F}} \phi \hookrightarrow a(s)\} \;\middle|\; \mathsf{syn} \hookrightarrow \mathsf{M.hcom}_A\right\}\\ \mathsf{Coe}(A) &\cong \left\{\prod_{r,s:\mathbb{I}}\prod_{a:A(r)}\{A(s) \mid r=s \hookrightarrow a\} \;\middle|\; \mathsf{syn} \hookrightarrow \mathsf{M.coe}_A\right\}\end{aligned}$$

Fig. 4. Definitions of homogeneous composition structures for a computability structure and coercion structures for a line of computability structures; the use of $\mathscr{U}_{\bullet}$ indicates that these structures are $\bigcirc$-*connected*, i.e. equivalently classified by the subuniverse $\{\mathscr{U} \mid \mathsf{syn} \hookrightarrow \mathbf{1}\}$.

$$\begin{aligned}\mathsf{nftp} &: \{\mathscr{U} \mid \mathsf{syn} \hookrightarrow \mathsf{M.tp}\}\\ \mathsf{ne}_{(-)} &: \prod_{\phi:\mathbb{F}}\prod_{A:\mathsf{M.tp}}\{\mathscr{U} \mid \phi \vee \mathsf{syn} \hookrightarrow \mathsf{M.tm}(A)\}\\ \mathsf{nf} &: \prod_{A:\mathsf{M.tp}}\{\mathscr{U} \mid \mathsf{syn} \hookrightarrow \mathsf{M.tm}(A)\}\\ \mathfrak{var} &: \{A\}\ \{\mathsf{var}(A) \to \mathsf{ne}_{\perp_{\mathbb{F}}}(A) \mid \mathsf{syn} \hookrightarrow \lambda x.x\}\end{aligned}$$

Fig. 5. Axiomatization of the structure of normal and neutral forms. The remainder of the generators are distributed across Figs. 6 to 10.

computability structures with both homogeneous composition and coercion, which we call *Kan*.



Because all the structures we are adding to $\mathscr{U}_{\mathsf{tp}}$ remain $\bigcirc$-connected, we may align this pullback as a large type $\mathscr{U}^{\mathsf{kan}}_{\mathsf{tp}} : \{\mathscr{V} \mid \mathsf{syn} \hookrightarrow \mathsf{M.tp}\}$. The left map $\mathscr{U}^{\mathsf{kan}}_{\mathsf{tp}} \longrightarrow \mathscr{U}^{\mathsf{hcom}}_{\mathsf{tp}}$ projects a homogeneous composition operation hcom_A for every Kan computability structure $A : \mathscr{U}^{\mathsf{kan}}_{\mathsf{tp}}$; transposing the upstairs map, we see that each line of Kan computability structures $A : \mathbb{I} \to \mathscr{U}^{\mathsf{kan}}_{\mathsf{tp}}$ also carries a coercion structure coe_A. We leave the composite projection $\mathscr{U}^{\mathsf{kan}}_{\mathsf{tp}} \longrightarrow \mathscr{U}_{\mathsf{tp}}$ implicit.

D. Neutral and normal forms

In order to axiomatize the neutral and normal forms, we will need a computability structure of *term variables*.

Axiom STC-6. *We assume a family of types* $\mathsf{var} : \prod_{A:\mathsf{M.tp}}\{\mathscr{U} \mid \mathsf{syn} \hookrightarrow \mathsf{M.tm}(A)\}$.

In Fig. 5, we axiomatize the judgmental structure of the normal and neutral forms of cubical type theory in the language of synthetic Tait computability; Figs. 6 to 8 contain the normal and neutral forms of path, Π, and glue types (within dashed boxes); Σ-types and the circle are located in the appendix (Figs. 9 and 10). The main part of the normalization argument only needs constants of the kind listed to exist, but we will substantiate these constants with an external inductive definition in Appendix C.

As discussed in Section I-C3, the main difficulty in adapting Coquand's semantic normalization argument [20] to cubical type theory is that neutral terms do *not* evince a cubically-stable aspect of the syntax of cubical type theory. The simplest example of this behavior is the neutral form of a path application $\mathfrak{papp}(p, i)$, which is "neutral" in the traditional sense only so long as i is neither 0 nor 1.

Warning 16. It is reasonable but ultimately futile to try and restrict the second argument of $\mathfrak{papp}$ to be a "variable" of some kind—in doing so, one refutes either the tininess of the interval or the existence of a Tait *reflection* operation for paths.

The failure of all previous attempts to isolate the neutral forms of cubical type theory stems ultimately from an insistence on characterizing *positively* the conditions under which a term is neutral. We have taken the opposite perspective, by indexing the neutrals in a "locus of instability" $\phi : \mathbb{F}$ under which they *cease* to be neutral; as soon as ϕ becomes true, the semantic information carried by $a : \mathsf{ne}_{\phi}(A)$ collapses to a point. Our negative perspective suggests a way to "stabilize" a neutral form by gluing computability data onto it along its locus of instability.

Definition 17 (Stabilized neutrals)**.** Let $A : \mathscr{U}_{\mathsf{tp}}$ be a computability structure; a *stabilized neutral* is a pair of a neutral $a_0 : \mathsf{ne}_{\phi}(A)$ together with a computability datum $a : \prod_{\_:[\phi]}\{A \mid \mathsf{syn} \hookrightarrow a_0\}$ defined on its locus of instability. We will write $[a_0 \mid \phi \hookrightarrow a]$ for such pairs, and obtain by realignment a type family of stabilized neutrals:

$$\begin{aligned}\mathsf{sne}_{\phi} &: \{\mathscr{U}_{\mathsf{tp}} \to \mathscr{U} \mid \mathsf{syn} \vee \phi \hookrightarrow [\mathsf{syn} \hookrightarrow \mathsf{M.tm}, \phi \hookrightarrow \lambda A.A]\}\\ \mathsf{sne}_{\phi}(A) &\cong \sum_{a_0:\mathsf{ne}_{\phi}(A)}\prod_{\_:[\phi]}\{A \mid \mathsf{syn} \hookrightarrow a_0\}\end{aligned}$$

E. Cubical normalization structures

We now reach the central definition of this paper, that of a *cubical normalization structure*, a notion inspired by the Tait closure condition [60] under which neutrals can be reflected to computable elements and computable elements can be reified to normals, as presented for instance by Coquand [20]. Our version of the Tait reflection operation takes *stabilized* neutrals to computable elements.

Definition 18. A *cubical normalization structure* $A : \mathsf{tp}$ consists of the following data:

$$\begin{aligned}&[A] : \mathscr{U}^{\mathsf{kan}}_{\mathsf{tp}}\\ &\Downarrow A : \{\mathsf{nftp} \mid \mathsf{syn} \hookrightarrow [A]\} && \text{(normal form)}\\ &\uparrow^{(-)}_{A} : \prod_{\phi:\mathbb{F}}\{\mathsf{sne}_{\phi}([A]) \to [A] \mid \mathsf{syn} \vee \phi \hookrightarrow \lambda a.a\} && \text{(reflect)}\\ &\downarrow_{A} : \{[A] \to \mathsf{nf}([A]) \mid \mathsf{syn} \hookrightarrow \lambda a.a\} && \text{(reify)}\end{aligned}$$

$$\begin{aligned}
\mathfrak{path} &: \left\{\left(\textstyle\sum_{A:\mathbb{I}\to\mathsf{nftp}}\mathsf{nf}(A(0))\times\mathsf{nf}(A(1))\right)\to\mathsf{nftp}\;\middle|\;\mathsf{syn}\hookrightarrow\mathsf{M.path}\right\}\\
\mathfrak{plam} &: \{A\}\left\{\textstyle\prod_{p:\prod_{i:\mathbb{I}}\mathsf{nf}(A(i))}\mathsf{nf}(\mathsf{M.path}(A,p(0),p(1)))\;\middle|\;\mathsf{syn}\hookrightarrow\lambda p.\lambda i.p(i)\right\}\\
\mathfrak{papp} &: \{\phi,A,a_0,a_1\}\left\{\mathsf{ne}_\phi(\mathsf{M.path}(A,a_0,a_1))\to\textstyle\prod_{i:\mathbb{I}}\mathsf{ne}_{\phi\vee_{\mathbb{F}}\partial i}(A(i))\;\middle|\;\mathsf{syn}\hookrightarrow\lambda p.\lambda i.p(i)\right\}
\end{aligned}$$

$$\begin{aligned}
\mathsf{path} &: \left\{\left(\textstyle\sum_{A:\mathbb{I}\to\mathsf{tp}}A(0)\times A(1)\right)\to\mathsf{tp}\;\middle|\;\mathsf{syn}\hookrightarrow\mathsf{M.path}\right\}\\
[\mathsf{path}(A,a_0,a_1)] &\cong \textstyle\prod_{i:\mathbb{I}}\{A(i)\mid\partial i\hookrightarrow[\overline{i=\epsilon\hookrightarrow a_\epsilon}]_{\mathsf{M.tm}(A(i))}\}\\
\mathsf{hcom}^{r\to s;\phi}_{\mathsf{path}(A,a_0,a_1)}p &= \lambda i.\mathsf{hcom}^{r\to s;\phi\vee_{\mathbb{F}}\partial i}_{A(i)}\lambda k.[k=r\vee_{\mathbb{F}}\phi\hookrightarrow p(k),\overline{i=\epsilon\hookrightarrow a_\epsilon}]_{\mathsf{M.tm}(A(i))}\\
\mathsf{coe}^{r\to s}_{\lambda j.\mathsf{path}(A(i,j),a_0(j),a_1(j))}p &= \lambda i.\mathsf{com}^{r\to s;\partial i}_{\lambda j.A(i,j)}\lambda j.[j=r\to p(i),\overline{i=\epsilon\hookrightarrow a_\epsilon(j)}]_{\mathsf{M.tm}(A(i,j))}\\
\Downarrow\mathsf{path}(A,a_0,a_1) &= \mathfrak{path}(\lambda i.\Downarrow A(i),\downarrow_{A(0)}a_0,\downarrow_{A(1)}a_1)\\
\uparrow^{\phi}_{\mathsf{path}(A,a_0,a_1)}[p_0\mid\phi\hookrightarrow p] &= \lambda i.\uparrow^{\phi\vee_{\mathbb{F}}\partial i}_{A(i)}[\mathfrak{papp}(p_0,i)\mid\phi\vee_{\mathbb{F}}\partial i\hookrightarrow[\phi\hookrightarrow p(i),\overline{i=\epsilon\hookrightarrow a_\epsilon}]_{\mathsf{M.tm}(A(i))}]\\
\downarrow_{\mathsf{path}(A,a_0,a_1)}p &= \mathfrak{plam}(\lambda i.\downarrow_{A(i)}p(i))
\end{aligned}$$

Fig. 6. The cubical normalization structure for dependent path types.

$$\begin{aligned}
\mathfrak{pi} &: \left\{\left(\textstyle\sum_{A:\mathsf{nftp}}\prod_{x:\mathsf{var}(A)}\mathsf{nftp}\right)\to\mathsf{nftp}\;\middle|\;\mathsf{syn}\hookrightarrow\mathsf{M}.\Pi\right\}\\
\mathfrak{lam} &: \{A,B\}\left\{\left(\textstyle\prod_{x:\mathsf{var}(A)}\mathsf{nf}(B(x))\right)\to\mathsf{nf}(\mathsf{M}.\Pi(A,B))\;\middle|\;\mathsf{syn}\hookrightarrow\lambda f.\lambda x.f(x)\right\}\\
\mathfrak{app} &: \{\phi,A,B\}\left\{\mathsf{ne}_\phi(\mathsf{M}.\Pi(A,B))\to\textstyle\prod_{x:\mathsf{nf}(A)}\mathsf{ne}_\phi(B(x))\;\middle|\;\mathsf{syn}\hookrightarrow\lambda f.\lambda x.f(x)\right\}
\end{aligned}$$

$$\begin{aligned}
\Pi &: \left\{\left(\textstyle\sum_{A:\mathsf{tp}}(A\to\mathsf{tp})\right)\to\mathsf{tp}\;\middle|\;\mathsf{syn}\hookrightarrow\mathsf{M}.\Pi\right\}\\
[\Pi(A,B)] &\cong \textstyle\prod_{x:A}B(x)\\
\mathsf{hcom}^{r\to s;\phi}_{\Pi(A,B)}f &= \lambda x.\mathsf{hcom}^{r\to s;\phi}_{B}\lambda i.[i=r\vee_{\mathbb{F}}\phi\hookrightarrow f(x,i)]\\
\mathsf{coe}^{r\to s}_{\lambda i.\Pi(A(i),B(i))}f &= \lambda x.\mathsf{coe}^{r\to s}_{\lambda i.B(i,\mathsf{coe}^{s\to i}_{A}x)}f(\mathsf{coe}^{s\to r}_{A}x)\\
\Downarrow\Pi(A,B) &= \mathfrak{pi}(\Downarrow A,\lambda x.\Downarrow B(\uparrow^{\bot_{\mathbb{F}}}_{A}[\mathfrak{var}(x)\mid\bot_{\mathbb{F}}\hookrightarrow[]_{\mathsf{M.tm}(A)}]))\\
\uparrow^{\phi}_{\Pi(A,B)}[f_0\mid\phi\hookrightarrow f] &= \lambda x.\uparrow^{\phi}_{B(x)}[\mathfrak{app}(f_0,\downarrow_A x)\mid\phi\hookrightarrow f(x)]\\
\downarrow_{\Pi(A,B)}f &= \mathfrak{lam}(\lambda x.\mathbf{let}\ \tilde{x}=\uparrow^{\bot_{\mathbb{F}}}_{A}[\mathfrak{var}(x)\mid\bot_{\mathbb{F}}\hookrightarrow[]_{\mathsf{M.tm}(A)}]\ \mathbf{in}\ \downarrow_{B(\tilde{x})}f(\tilde{x}))
\end{aligned}$$

Fig. 7. The cubical normalization structure for dependent product types.

$$\begin{aligned}
\mathfrak{glue} &: \{\phi\}\left\{\left(\textstyle\sum_{B:\mathsf{nftp}}\sum_{A:[\phi]\to\mathsf{nftp}}\prod_{_:[\phi]}\mathsf{nf}(\mathsf{Equiv}(A,B))\right)\to\mathsf{nftp}\;\middle|\;\mathsf{syn}\vee\phi\hookrightarrow[\mathsf{syn}\hookrightarrow\mathsf{M.glue}(\phi,-),\phi\hookrightarrow\lambda(B,A,f).A]\right\}\\
\mathfrak{englue} &: \{\phi,B,A,f\}\left\{\left(\textstyle\sum_{a:\prod_{_:[\phi]}\mathsf{nf}(A)}\{\mathsf{nf}(B)\mid\mathsf{syn}\wedge\phi\hookrightarrow f(a)\}\right)\to\mathsf{nf}(\mathsf{M.glue}(\phi,B,A,f))\;\middle|\;\mathsf{syn}\vee\phi\hookrightarrow\lambda(a,b).[\mathsf{syn}\hookrightarrow\mathsf{M.glue/tm}(a,b),\phi\hookrightarrow a]\right\}\\
\mathfrak{unglue} &: \{\phi,B,A,f,\psi\}\{\mathsf{ne}_\psi(\mathsf{M.glue}(\phi,B,A,f))\to\mathsf{ne}_{\psi\vee_{\mathbb{F}}\phi}(B)\mid\mathsf{syn}\hookrightarrow\mathsf{M.unglue}(\phi,-)\}
\end{aligned}$$

$$\begin{aligned}
\mathsf{glue} &: \left\{\textstyle\prod_{\phi:\mathbb{F}}\left\{\left(\sum_{B:\mathsf{tp}}\sum_{A:[\phi]\to\mathsf{tp}}\prod_{_:[\phi]}\mathsf{Equiv}(A,B)\right)\to\mathsf{tp}\;\middle|\;\phi\hookrightarrow\lambda(B,A,f).A\right\}\;\middle|\;\mathsf{syn}\hookrightarrow\mathsf{M.glue}\right\}\\
[\mathsf{glue}(\phi,B,A,f)] &\cong \textstyle\sum_{a:\prod_{_:[\phi]}A}\{B\mid\phi\hookrightarrow f(a)\}\\
\mathsf{hcom}^{r\to s;\psi}_{\mathsf{glue}(\phi,B,A,f)}p &= \langle\text{omitted for brevity, see [9]}\rangle\\
\mathsf{coe}^{r\to s}_{\lambda i.\mathsf{glue}(\phi(i),B(i),A(i),f(i))}f &= \langle\text{omitted for brevity, see [9]}\rangle\\
\Downarrow\mathsf{glue}(\phi,B,A,f) &= \mathfrak{glue}(\phi,\Downarrow B,\Downarrow A,\downarrow_{\mathsf{Equiv}(A,B)}f)\\
\uparrow^{\psi}_{\mathsf{glue}(\phi,B,A,f)}[g_0\mid\psi\hookrightarrow g] &= \left([\phi\hookrightarrow\uparrow^{\psi}_{A}[g_0\mid\psi\hookrightarrow g]],\uparrow^{\psi}_{B}[\mathfrak{unglue}(g_0)\mid\psi\vee_{\mathbb{F}}\phi\hookrightarrow[\psi\hookrightarrow\pi_2(g),\phi\hookrightarrow f(\uparrow^{\psi}_{A}[g_0\mid\psi\hookrightarrow g])]_B]\right)\\
\downarrow_{\mathsf{glue}(\phi,B,A,f)}(a,b) &= \mathfrak{englue}([\phi\hookrightarrow\downarrow_A a],\downarrow_B b)
\end{aligned}$$

Fig. 8. The cubical normalization structure for glue types.

We define by realignment the large type tp of cubical normalization structures, noting that the latter three components of a cubical normalization structure are $\bigcirc$-connected:

$$\mathsf{tp} : \{\mathscr{V} \mid \mathsf{syn} \hookrightarrow \mathsf{M.tp}\}$$

Remark 19 (Vertical maps)**.** We refer to the reflection and reification maps as *vertical*, in the sense that they are constrained to lie over the syntactic identity function.

The role of verticality is to ensure that reification takes computability data for a given term to a normal form *of the same term*, etc. Likewise, our presentation of the neutral and normal forms use extent types to express their relationship to the syntactic $\mathscr{T}_{\square}$-model without escaping the internal language of $\mathscr{E}$. In this way, extent types and vertical maps play a very important role in synthetic Tait computability.

The main result of this section is the construction of a computability algebra for Cartesian cubical type theory; this equips each syntactic type with a Kan computability structure, normal form, and reflection and reification maps.

Theorem 20. *We have a* computability $\mathscr{T}_{\square}$*-model* $\mathsf{M}' : \{\mathscr{T}_{\square}\textbf{-Mod}(\mathscr{U}) \mid \mathsf{syn} \hookrightarrow \mathsf{M}\}$ *aligned over the syntactic algebra.*

Proof. We define $\mathsf{M}'.\mathsf{tp} = \mathsf{tp}$, $\mathsf{M}'.\mathsf{tm}(A) = [A]$, $\mathsf{M}'.\mathbb{I} = \mathbb{I}$, and $\mathsf{M}'.\mathbb{F} = \mathbb{F}$. In Figs. 6 to 11, we show how to close the universe of cubical normalization structures tp under the connectives path, Π, Σ, glue, and $\mathsf{S1}$. The fact that the resulting model is aligned over M follows from each of these components being aligned over their syntactic counterparts; in particular, each connective is aligned over the syntactic connective in M. □

IV. The computability topos

We now define the category $\mathscr{E}$ in which Section III takes place, as a category of sheaves on a generalized space $\mathbf{G}_{\square}$ which combines syntax and semantics. Mirroring the modal syntax–semantics duality introduced in Section III-A, sheaves on $\mathbf{G}_{\square}$ function as computability structures because they have syntactic and semantic aspects obtained by restriction to the corresponding regions of the space.

We construct $\mathbf{G}_{\square}$ in turn by starting with a syntactic topos $\mathbf{T}_{\square}$ that contains the generic model of Cartesian cubical type theory, and gluing it onto a semantic topos $\mathbf{A}_{\square}$ over which the notions of variable, neutral, and normal form are definable.

A. The language of topoi

Topoi are sometimes thought of as generalized topological spaces, and sometimes as special kinds of categories. These perspectives are complementary, but one avoids many notational and conceptual quagmires by distinguishing them formally [5, 16, 66]: in the tradition of Grothendieck, the 2-category of topoi is *opposite* to the 2-category of cocomplete and finitely complete categories satisfying Giraud's exactness axioms [10].[3] This explains why the product of two topoi is given by a *tensor product* of categories; the situation is analogous to the other dualities between geometry and algebra in mathematics, such as locales/frames, affine schemes/commutative rings, Stone spaces/Boolean algebras, etc.

[3]The 1-cells are reversed, but the 2-cells remain the same.

Notation 21. Given a topos $\mathbf{X}$, we will write $\mathrm{Sh}(\mathbf{X})$ for its *category of sheaves*, which is the formal avatar of $\mathbf{X}$ in the opposite category. In traditional parlance, morphisms of topoi go in the "direct image" direction, and morphisms of categories of sheaves go in the "inverse image" direction.

Example 22 (Presheaves)**.** Given a small category $\mathscr{C}$, the category of presheaves $\mathrm{Pr}(\mathscr{C})$ is the category of functors $\mathscr{C}^{\mathrm{op}} \longrightarrow \mathbf{Set}$. Because $\mathrm{Pr}(\mathscr{C})$ satisfies Giraud's axioms, there is a topos $\widehat{\mathscr{C}}$ satisfying $\mathrm{Sh}(\widehat{\mathscr{C}}) = \mathrm{Pr}(\mathscr{C})$.

If $\mathscr{C}$ has all finite limits, then $\widehat{\mathscr{C}}$ has a universal property by Diaconescu's theorem [24]: morphisms of topoi $\mathbf{X} \longrightarrow \widehat{\mathscr{C}}$ are the same as left exact functors $\mathscr{C} \longrightarrow \mathrm{Sh}(\mathbf{X})$, i.e. functors that preserve finite limits. Viewing $\mathscr{C}$ as a finite limit theory, this universal property is summed up by referring to $\widehat{\mathscr{C}}$ as the classifying topos of $\mathscr{C}$-models.

B. The syntactic topos

Recall from Section II that a model of $\mathscr{T}_{\square}$ in $\mathrm{Sh}(\mathbf{X})$ is a locally Cartesian closed functor $\mathscr{T}_{\square} \longrightarrow \mathrm{Sh}(\mathbf{X})$; when this functor only preserves finite limits but not dependent products, we refer to it as a *pre-model*. Pre-models play an important role in the metatheory of higher-order logic (as in the "general models" of Henkin [32], later studied in the language of topoi by Awodey and Butz [11]), as well as the metatheory of dependent type theory (as in pseudo-morphisms of cwfs [39, 45]).

We define the syntactic topos $\mathbf{T}_{\square}$ as the presheaf topos $\widehat{\mathscr{T}_{\square}}$. In light of Example 22, $\mathbf{T}_{\square}$ is the classifying topos of pre-models of $\mathscr{T}_{\square}$, in the sense that morphisms of topoi $\mathbf{X} \longrightarrow \widehat{\mathscr{T}_{\square}}$ correspond to pre-models of $\mathscr{T}_{\square}$ in $\mathrm{Sh}(\mathbf{X})$.

C. The topos of cubical atomic terms

Next, we define the semantic topos $\mathbf{A}_{\square}$ and an essential morphism of topoi $\boldsymbol{\alpha} : \mathbf{A}_{\square} \longrightarrow \mathbf{T}_{\square}$ along which we will glue in Section IV-D; that $\boldsymbol{\alpha}$ is essential means there is an additional left adjoint $\boldsymbol{\alpha}_! \dashv \boldsymbol{\alpha}^* \dashv \boldsymbol{\alpha}_*$, a technical condition that will play an important role in Section IV-E. Intuitively, $\mathbf{A}_{\square}$ is the topos of *cubically atomic terms*, i.e., term variables and elements of the interval; concretely, we equip $\mathbf{A}_{\square}$ with a tiny interval object $\mathbb{I} \cong \boldsymbol{\alpha}^*\mathbb{I} : \mathrm{Sh}(\mathbf{A}_{\square})$ and a fiberwise-tiny family of term variables $\mathsf{var} : \mathrm{Sh}(\mathbf{A}_{\square})_{/\boldsymbol{\alpha}^*\mathsf{tm}}$ indexed in syntactic types.

We define $\mathbf{A}_{\square} := \widehat{\mathscr{A}_{\square}}$, where $\mathscr{A}_{\square}$ is a category of *cubical atomic contexts and substitutions* whose objects $\Gamma : \mathscr{A}_{\square}$ we define simultaneously with their decodings $\boldsymbol{\alpha}(\Gamma) : \mathscr{T}_{\square}$:

$$\frac{}{\cdot : \mathscr{A}_{\square} \qquad \boldsymbol{\alpha}(\cdot) = \mathbf{1}_{\mathscr{T}_{\square}}} \qquad \frac{\Gamma : \mathscr{A}_{\square} \qquad A : \boldsymbol{\alpha}(\Gamma) \longrightarrow \mathsf{tp}}{\Gamma.A : \mathscr{A}_{\square} \qquad \boldsymbol{\alpha}(\Gamma.A) = \boldsymbol{\alpha}(\Gamma).A}$$

$$\frac{\Gamma : \mathscr{A}_{\square}}{\Gamma.\mathbb{I} : \mathscr{A}_{\square} \qquad \boldsymbol{\alpha}(\Gamma.\mathbb{I}) = \boldsymbol{\alpha}(\Gamma) \times \mathbb{I}}$$

Before defining the morphisms of $\mathscr{A}_\square$, we first characterize the term variables:

$$\textsc{top variable}\quad \frac{A : \boldsymbol{\alpha}(\Gamma) \longrightarrow \mathsf{tp}}{\Gamma.A \Vdash \mathfrak{z}_A : A \qquad \boldsymbol{\alpha}(\mathfrak{z}_A) = q_A}$$

$$\textsc{pop variable}\quad \frac{\Gamma \Vdash \alpha : A \qquad Q \in \{\mathbb{I}\} \cup \{B : \boldsymbol{\alpha}(\Gamma) \longrightarrow \mathsf{tp}\}}{\Gamma.Q \Vdash \mathfrak{s}_Q(\alpha) : A \circ p_Q \qquad \boldsymbol{\alpha}(\mathfrak{s}_Q(\alpha)) = \boldsymbol{\alpha}(\alpha) \circ p_Q}$$

Then, we define the morphisms $\gamma : \Delta \longrightarrow \Gamma$ simultaneously with their decodings $\boldsymbol{\alpha}(\gamma)$ in terms of the cubical atomic terms:

$$\textsc{empty}\quad \frac{}{\cdot : \Delta \longrightarrow \cdot \qquad \boldsymbol{\alpha}(\cdot) = !_{\boldsymbol{\alpha}(\Delta)}}$$

$$\textsc{variable}\quad \frac{\gamma : \Delta \longrightarrow \Gamma \qquad \Delta \Vdash \alpha : A \circ \boldsymbol{\alpha}(\gamma)}{\gamma.\alpha : \Delta \longrightarrow \Gamma.A \qquad \boldsymbol{\alpha}(\gamma.\alpha) = \langle \boldsymbol{\alpha}(\gamma), \boldsymbol{\alpha}(\alpha) \rangle}$$

$$\textsc{dimension}\quad \frac{\gamma : \Delta \longrightarrow \Gamma \qquad r : \boldsymbol{\alpha}(\Delta) \longrightarrow \mathbb{I}}{\gamma.r : \Delta \longrightarrow \Gamma.\mathbb{I} \qquad \boldsymbol{\alpha}(\gamma.r) = \langle \boldsymbol{\alpha}(\gamma), r \rangle}$$

The above decodings assemble into a functor $\boldsymbol{\alpha} : \mathscr{A}_\square \longrightarrow \mathscr{T}_\square$, which automatically induces an essential morphism of topoi that we will also write $\boldsymbol{\alpha} : \mathbf{A}_\square \longrightarrow \mathbf{T}_\square$.

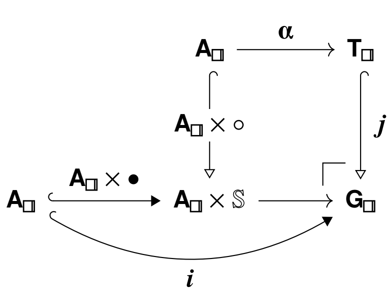
Lemma 23. *The chosen interval structure is preserved by restriction along $\boldsymbol{\alpha} : \mathbf{A}_\square \longrightarrow \mathbf{T}_\square$; that is, we have an isomorphism $\mathsf{y}_{\mathscr{A}_\square}(\cdot.\mathbb{I}) \cong \boldsymbol{\alpha}^*\mathsf{y}_{\mathscr{T}_\square}(\mathbb{I})$.*

Construction 24 (The presheaf of variables)**.** We have a family $\mathsf{var} : \mathrm{Sh}(\mathbf{A}_\square)_{/\boldsymbol{\alpha}^*\mathsf{tm}}$, whose fiber at each $a : \boldsymbol{\alpha}(\Gamma) \longrightarrow \mathsf{tm}(A)$ is the set of variables $\Gamma \Vdash \mathfrak{a} : A$ with $\boldsymbol{\alpha}(\mathfrak{a}) = a$.

D. The glued topos

Let $\mathbb{S}$ be the *Sierpiński* topos satisfying $\mathrm{Sh}(\mathbb{S}) = \mathbf{Set}^{\rightarrow}$. Writing (pt) for the punctual topos satisfying $\mathrm{Sh}(\mathsf{pt}) = \mathbf{Set}$, we have open and closed points $\circ : (\mathsf{pt}) \hookrightarrow \mathbb{S}$ and $\bullet : (\mathsf{pt}) \hookrightarrow \mathbb{S}$ corresponding under inverse image to the codomain and domain functors on $\mathbf{Set}$ respectively. In geometrical terms, these endpoints render $\mathbb{S}$ a *directed interval* $\{\bullet \to \circ\}$ that can be used to form cylinders by cartesian product.

We define the glued topos $\mathbf{G}_\square$ by gluing the open end of the cylinder $\mathbf{A}_\square \times \mathbb{S}$ onto $\mathbf{T}_\square$ along $\boldsymbol{\alpha} : \mathbf{A}_\square \longrightarrow \mathbf{T}_\square$, obtaining a diagram of open and closed immersions $\mathbf{T}_\square \overset{\boldsymbol{j}}{\hookrightarrow} \mathbf{G}_\square \overset{\boldsymbol{i}}{\hookleftarrow} \mathbf{A}_\square$ as follows:



Finally, we define the category $\mathscr{E}$ axiomatized in Section III to be $\mathscr{E} := \mathrm{Sh}(\mathbf{G}_\square)$.

Remark 25. The gluing construction above has the advantage of being expressed totally in the category of Grothendieck topoi; for familiarity, we note that the category $\mathscr{E} = \mathrm{Sh}(\mathbf{G}_\square)$ is the traditional Artin gluing of the inverse image functor $\boldsymbol{\alpha}^* : \mathrm{Pr}(\mathscr{T}_\square) \longrightarrow \mathrm{Pr}(\mathscr{A}_\square)$ and can hence be explicitly computed as the comma category $\mathrm{Pr}(\mathscr{A}_\square) \downarrow \boldsymbol{\alpha}^*$. Therefore, an object of $\mathscr{E}$ is a pair $(E, A \longrightarrow \boldsymbol{\alpha}^*E)$ of a presheaf $E : \mathrm{Pr}(\mathscr{T}_\square)$ and a presheaf $A : \mathrm{Pr}(\mathscr{A}_\square)$ equipped with a structure map to $\boldsymbol{\alpha}^*E$.

E. Verifying the axioms

Because $\mathscr{E} = \mathrm{Sh}(\mathbf{G}_\square)$ is a category of sheaves, it satisfies Giraud's axioms and thus interprets extensional type theory extended with a universe Ω of propositions. To see that $\mathscr{E}$ moreover admits a hierarchy of type-theoretic universes satisfying the strictification axiom (Axiom STC-1), we observe that $\mathscr{E}$ also admits a presentation as a presheaf category.

Lemma 26 (Orton–Pitts universes)**.** *In $\mathscr{E}$ we have a cumulative and transfinite hierarchy of Orton–Pitts universes (Definition 5) $\mathscr{U}_0 \subseteq \mathscr{U}_1 \subseteq \ldots$ corresponding to the sequence of strongly inaccessible cardinals in the background set theory.*

Proof. First, we observe that $\mathbf{G}_\square$ can be presented as $\widehat{\mathscr{G}}$ for some small category $\mathscr{G}$, following a standard result of topos theory that the Artin gluing of a *continuous* and accessible functor between categories of presheaves is again a category of presheaves [10, 18]. Note that $\boldsymbol{\alpha}^*$ is continuous, as the inverse image part of an essential morphism of topoi.

We therefore obtain a hierarchy of Hofmann–Streicher universes [33] in $\mathscr{E} \simeq \mathrm{Pr}(\mathscr{G})$, which Coquand has additionally shown to be cumulative. Finally, using the argument of Orton and Pitts [47, Theorem 6.3] and our assumption that the background set theory is boolean, we conclude that the Hofmann–Streicher universes satisfy the strictification axiom. □

Using our computation of $\mathscr{E}$ as a comma category (Remark 25), we can explicitly compute the proposition syn stipulated by Axiom STC-3. Then, we construct the syntactic $\mathscr{T}_\square$-model, family of term variables, and interval object stipulated by Axioms STC-4, STC-6, and STC-2 respectively.

Computation 27 (The syntactic open)**.** We define $\mathsf{syn} : \mathscr{E}$ to be the subterminal object $\boldsymbol{j}_!\mathbf{1} = (\mathbf{1}, \mathbf{0} \longrightarrow \boldsymbol{\alpha}^*\mathbf{1})$, for which we have an equivalence of categories $\mathrm{Pr}(\mathscr{T}_\square) \simeq \mathscr{E}_{/\mathsf{syn}}$. The inverse image part of the open immersion can be viewed as the pullback map $\mathsf{syn}^* : \mathscr{E} \longrightarrow \mathscr{E}_{/\mathsf{syn}}$, and the direct image is the dependent product map $\mathsf{syn}_* : \mathscr{E}_{/\mathsf{syn}} \longrightarrow \mathscr{E}$.

Remark 28 (The syntactic $\mathscr{T}_\square$-model)**.** As the classifier of pre-models of $\mathscr{T}_\square$, the topos $\mathbf{T}_\square$ contains the *generic pre-model* of $\mathscr{T}_\square$, namely the Yoneda embedding $\mathscr{T}_\square \hookrightarrow \mathrm{Sh}(\mathbf{T}_\square) = \mathrm{Pr}(\mathscr{T}_\square)$. Because the Yoneda embedding is locally Cartesian closed, the generic pre-model is in fact a genuine $\mathscr{T}_\square$-model; because $\mathscr{T}_\square$ is small, the size of this model is bounded by the smallest $\bigcirc$-modal Hofmann–Streicher universe, allowing us to internalize it as an element of $\mathscr{T}_\square$-$\mathbf{Mod}(\mathscr{U}_\bigcirc)$.

Remark 29 (The family of term variables)**.** In light of Remark 25, the presheaf $\mathsf{var} : \mathrm{Pr}(\mathscr{A}_{\square})_{/\boldsymbol{\alpha}^*\mathsf{tm}}$ from Construction 24 can be internalized as a $\mathscr{U}$-valued family of types in $\mathscr{E}$ that restricts to $\mathsf{tm} : \mathrm{Pr}(\mathscr{T}_{\square})$ under the open immersion.

Construction 30 (The glued interval)**.** We take the interval $\mathbb{I} : \mathscr{E}$ to be the direct image $\mathbb{I} := \boldsymbol{j}_*\mathsf{y}_{\mathscr{T}_{\square}}(\mathbb{I})$ of the syntactic interval under the open immersion, so $\boldsymbol{j}^*\mathbb{I} = \mathsf{y}_{\mathscr{T}_{\square}}(\mathbb{I})$. Because of the isomorphism from Lemma 23, it is easy enough to see that $\boldsymbol{i}^*\mathbb{I} \cong \mathsf{y}_{\mathscr{A}_{\square}}(\cdot.\mathbb{I})$, completing our justification of Axiom STC-4.

To see that the glued interval is tiny, we use a general fact about Artin gluings along *inverse image* functors.

Lemma 31. *Let $f : \mathbf{Y} \longrightarrow \mathbf{U}$ be a morphism of topoi. Write $\mathbf{X}$ for the Artin gluing of the inverse image functor f^*, and write $j : \mathbf{U} \hookrightarrow \mathbf{X}$ and $i : \mathbf{Y} \hookrightarrow \mathbf{X}$ for the respective open and closed immersions of topoi. Suppose that $X : \mathrm{Sh}(\mathbf{X})$ is a sheaf such that j^*X is a tiny object in $\mathrm{Sh}(\mathbf{U})$ and i^*X is a tiny object in $\mathrm{Sh}(\mathbf{Y})$; then X is tiny.*

Proof. It suffices to check that the exponential functor $(X \to -)$ preserves colimits; see the appendix for details. □

Corollary 32. *The glued interval is tiny in $\mathscr{E}$.*

We substantiate Axiom STC-5 by showing that $\bot_{\mathbb{F}} \le \mathsf{syn}$ in the lattice of opens of $\mathbf{G}_{\square}$, and unfolding Construction 10; details are in the appendix (Lemmas 62 and 63).

Lemma 33. *A type $A : \mathscr{U}$ in $\mathscr{E}$ is $\mathbb{F}$-local if and only if $\bigcirc A$ is $\mathbb{F}$-local.*

Finally, we construct the normal and neutral forms as an indexed quotient inductive type [3] valued in $\mathscr{U}_{\bullet}$; the full definition appears in Fig. 12 in the supplementary appendix.

V. Normalization for cubical type theory

Finally, we show that our computability model (Theorem 20) lets us compute the normal form of every syntactic type, implying the (external to $\mathscr{E}$) decidability of type equality in cubical type theory, and the injectivity of type constructors.

Remark 34. Because $\boldsymbol{\alpha} : \mathbf{A}_{\square} \longrightarrow \mathbf{T}_{\square}$ is an essential morphism of topoi with additional left adjoint $\boldsymbol{\alpha}_! \dashv \boldsymbol{\alpha}^*$, so is the closed immersion $\boldsymbol{i} : \mathbf{A}_{\square} \hookrightarrow \mathbf{G}_{\square}$; the additional left adjoint $\boldsymbol{i}_!$ takes $E : \mathrm{Pr}(\mathscr{A}_{\square})$ to the computability structure $(\boldsymbol{\alpha}_!E, E \longrightarrow \boldsymbol{\alpha}^*\boldsymbol{\alpha}_!E)$ determined by the unit of the monad $\boldsymbol{\alpha}^*\boldsymbol{\alpha}_!$.

Construction 35. Let $\Gamma : \mathscr{A}_{\square}$ be an atomic context; we write $(\!|\Gamma|\!) : \mathscr{E}$ for $\boldsymbol{i}_!\mathsf{y}_{\mathscr{A}_{\square}}(\Gamma)$, the computability structure of vectors of "atoms of type Γ" tracked by honest substitutions/terms.

Lemma 36. *For any $X : \mathscr{E}$, we have a canonical isomorphism $[(\!|-|\!), X] \cong \boldsymbol{i}^*X : \mathrm{Pr}(\mathscr{A}_{\square})$ determined by adjoint transpose and the Yoneda lemma.*

Construction 37. The computability model evinces a locally Cartesian closed functor $\mathsf{M}' : \mathscr{T}_{\square} \longrightarrow \mathscr{E}$; restricting along $\boldsymbol{\alpha} : \mathscr{A}_{\square} \longrightarrow \mathscr{T}_{\square}$, we have an interpretation functor $[\![-]\!] : \mathscr{A}_{\square} \longrightarrow \mathscr{E}$ taking each atomic context to its computability structure $\mathsf{M}'(\boldsymbol{\alpha}(\Gamma))$. We observe that $\boldsymbol{j}^*[\![\Gamma]\!] = \mathsf{y}_{\mathscr{T}_{\square}}(\boldsymbol{\alpha}(\Gamma)) = \boldsymbol{\alpha}_!\mathsf{y}_{\mathscr{A}_{\square}}(\Gamma)$.

Construction 38. For any $X : \mathscr{E}$ there is a canonical natural transformation $[[\![-]\!], X] \longrightarrow \boldsymbol{\alpha}^*\boldsymbol{j}^*X : \mathrm{Pr}(\mathscr{A}_{\square})$ which restricts $\mathsf{M}'(\boldsymbol{\alpha}(\Gamma)) \longrightarrow X : \mathscr{E}$ to its syntactic part, noting that $\mathrm{Hom}(\boldsymbol{\alpha}_!\mathsf{y}_{\mathscr{A}_{\square}}(\Gamma), \boldsymbol{j}^*X) \cong \boldsymbol{\alpha}^*\boldsymbol{j}^*X(\Gamma)$. Viewed as a sheaf on $\mathbf{G}_{\square}$, we write $X_{\mathsf{M}'} : \mathscr{E}$ for the pair $\left(\boldsymbol{j}^*X, [[\![-]\!], X] \longrightarrow \boldsymbol{\alpha}^*\boldsymbol{j}^*X\right)$.

Construction 39. We define a pointwise vertical (Remark 19) natural transformation $\mathsf{atom} : (\!|-|\!) \longrightarrow [\![-]\!] : \mathrm{Hom}(\mathscr{A}_{\square}, \mathscr{E})$ that reflects each atomic substitution as a computable substitution. The definition follows by recursion on the index $\Gamma : \mathscr{A}_{\square}$, and uses the fact that the locus of instability of a variable is empty:

$$\begin{aligned} \mathsf{atom}_{(\cdot)}(\cdot) &= \cdot \\ \mathsf{atom}_{\Gamma.\mathbb{I}}(\gamma.r) &= (\mathsf{atom}_\Gamma(\gamma), r) \\ \mathsf{atom}_{\Gamma.A}(\gamma.x) &= \left(\mathsf{atom}_\Gamma(\gamma), \uparrow^{\bot_{\mathbb{F}}}_{\mathsf{M}'(A)}[\mathfrak{var}(x) \mid \bot_{\mathbb{F}} \hookrightarrow []_{\mathsf{M}'(A)}]\right) \end{aligned}$$

Lemma 40. *The pointwise vertical natural transformation $\mathsf{atom} : (\!|-|\!) \longrightarrow [\![-]\!]$ induces by precomposition a vertical map $\mathsf{atom}^*_X : X_{\mathsf{M}'} \longrightarrow X$ for any sheaf $X : \mathscr{E}$.*

Theorem 41 (The normalization function)**.** *The functor $\mathsf{M}' : \mathscr{T}_{\square} \longrightarrow \mathscr{E}$ induces a vertical map $\mathsf{M.tp} \longrightarrow (\mathsf{M'.tp})_{\mathsf{M}'}$. Composing this with the vertical maps $\mathsf{atom}^*_{\mathsf{M'.tp}}$ and $\Downarrow$, we obtain a vertical* normalization *map* nbe *sending a syntactic type to the normal form chosen by its normalization structure in M':*

$$\mathsf{M.tp} \dashrightarrow (\mathsf{M'.tp})_{\mathsf{M}'} \xrightarrow{\mathsf{atom}^*_{\mathsf{M'.tp}}} \mathsf{M'.tp} \xrightarrow{\Downarrow} \mathsf{nftp}$$

We can similarly exhibit a pointwise vertical normalization function for syntactic *terms*:

$$\textstyle\prod_{A:\mathsf{M.tp}}\{\mathsf{M.tm}(A) \to \mathsf{nf}(A) \mid \mathsf{syn} \hookrightarrow \mathbf{id}\}$$

The standard correctness conditions (soundness and completeness) for normalization follow immediately.

Theorem 42 (Correctness of normalization)**.** *The normalization function is sound and complete for cubical type theory.*

1) Completeness — *if two (types, terms) are equal, then they are taken to equal normal forms.*
2) Soundness — *if two (types, terms) are taken to the same normal form, then they are equal.*

Proof. Completeness is automatic because our entire development was carried out relative to judgmental equivalence classes of terms. Soundness follows from the fact that the normalization function is vertical, hence a section to the unit of the open modality, hence a monomorphism. □

We would not expect the Π constructor to be injective in the syntactic category (a derivability): because monomorphisms are preserved by left exact functors, this would imply that any model of cubical type theory has injective type constructors. However, there is a *modal* version of injectivity corresponding to the traditional admissibility statement that *does* hold.

Theorem 43 (Injectivity of type constructors)**.** *The following formula holds in the internal logic of* $\mathscr{E} = \mathrm{Sh}(\mathbf{G}_{\square})$*:*

$$\forall A, A', B, B'.\quad \mathsf{M}.\Pi(A,B) = \mathsf{M}.\Pi(A',B') \implies \bullet((A,B) = (A',B'))$$

Proof. By completeness, $\mathsf{M}.\Pi(A,B) = \mathsf{M}.\Pi(A',B')$ implies:

$$\mathfrak{pi}(\mathsf{nbe}(A), \lambda x.\mathsf{nbe}(B(x))) = \mathfrak{pi}(\mathsf{nbe}(A'), \lambda x.\mathsf{nbe}(B'(x)))$$

By soundness, it suffices to show:

$$\bullet((\mathsf{nbe}(A), \lambda x.\mathsf{nbe}(B(x))) = (\mathsf{nbe}(A'), \lambda x.\mathsf{nbe}(B'(x))))$$

The result follows from injectivity of $\mathfrak{pi}$ in nftp, shown by a standard inductive argument (Lemma 60 in the appendix). □

Lemma 44 (Decidability of equality of normal forms)**.** *Given two normal forms* $A_0, A_1 : (\!|\Gamma|\!) \longrightarrow \mathsf{nftp}$ *externally, there is an effective algorithm to determine whether* $A_0 = A_1$ *or* $A_0 \neq A_1$*.*

Proof. As in the normal form presentation of strict coproducts [2], elements of nftp are not pure data: they include binders of type $\mathbb{I}$, $\mathsf{var}(A)$, and $[\phi]$. Nevertheless, equality is algorithmically decidable as follows, by recursion on Γ, A_0, A_1.

At a binder of type $\mathbb{I}$ or $\mathsf{var}(A)$, we continue at $\Gamma.\mathbb{I}$ or $\Gamma.A$ respectively. At a binder of type $[\phi]$, we note that our definition of $\mathbb{F}$ (Section III-B) licenses a case split on the form of $\phi : \mathbb{F}$. We eliminate universal quantifications in the style of Cohen et al. [19], proceeding by "left inversion" until reaching a conjunction of equations $\overline{r =_{\mathbb{I}} s}$. If the conjunction implies $0 = 1$, we halt; otherwise, we proceed under the equalizing atomic substitution $\Delta \to \Gamma$. □

Both the syntax and the normal forms of cubical type theory can be presented by finitely many rules in a conventional deductive system.

Theorem 45 (Normalization algorithm)**.** *There is a recursive function that assigns a normal form to a given well-typed raw term.*

Proof. The set of normal forms for a given term can be enumerated. By Theorems 41 and 42 this set is non-empty, hence there is a terminating recursive function that chooses a minimal normal form for any well-typed term. □

It is *a priori* not necessarily the case that the recursive function from Theorem 45 extends to a function on equivalence classes of terms. This follows, however, from the fact that the normalization function defined in Theorem 41 is *stable* in the sense of Lemma 48 below.

Notation 46. To begin with, we will use the following notations for the vertical interpretation maps induced by the normalization algebra M' defined as in Theorem 41:

$$[\![-]\!] : \mathsf{M}.\mathsf{tp} \to \mathsf{tp}$$
$$[\![-]\!] : \mathsf{M}.\mathsf{tm}(A) \to [\![A]\!]$$

Lemma 47 (Stability for variables)**.** *For all* $x : \mathsf{var}(A)$*, we have* $[\![x]\!] = \uparrow^{\bot}_{[\![A]\!]}[\mathfrak{var}(x) \mid \bot \hookrightarrow []]$.

Proof. Because this equation already holds when restricted over the open subtopos, it suffices to reason "upstairs" in $\mathrm{Sh}(\mathbf{A}_{\square})$, hence pointwise with respect to an arbitrary atomic context $\Gamma : \mathscr{A}_{\square}$. Here x is an atomic term $\Gamma \Vdash x : A$ and $[\![x]\!]$ is a function that projects the corresponding cell from any atomic substitution $\gamma : \mathsf{y}_{\mathscr{A}_{\square}}(\Gamma)$ and *reflects it* in the chosen normalization structure $[\![A]\!]\gamma$ as $\uparrow^{\bot}_{[\![A]\!]\gamma}[\mathfrak{var}(x\gamma) \mid \bot \hookrightarrow []]$, recalling Construction 39. □

Lemma 48 (Stability neutrals and normals)**.** *The normalization function is stable in the sense that the following hold:*

1) *For all* $x : \mathsf{ne}_{\phi}(A)$*, we must have* $[\![x]\!] = \uparrow^{\phi}_{[\![A]\!]}[x \mid \phi \hookrightarrow [\![x]\!]]$.
2) *For all* $a : \mathsf{nf}(A)$*, we must have* $\downarrow_{[\![A]\!]}[\![a]\!] = a$.
3) *For all* $A : \mathsf{nftp}$*, we must have* $\Downarrow[\![A]\!] = A$.

Hence unfolding the definition of the normalization function, we have $\mathsf{nbe}(A) = A$ *for any normal form* $A : \mathsf{nftp}$*, and likewise for normal forms of terms.*

Proof. By simultaneous induction on the normal and neutral forms, using essentially the argument of Kaposi [38]. Because the syntactic part of the normalization function is the identity, it suffices to reason in the language of $\mathrm{Sh}(\mathbf{A}_{\square})$.

1) The case for a variable is Lemma 48.
2) The case for a neutral function application $\mathfrak{app}(f, a)$ is as follows. By induction we have $[\![f]\!] = \uparrow^{\phi}_{[\![\mathsf{M}.\Pi(A,B)]\!]}[f \mid \phi \hookrightarrow [\![f]\!]]$ and $\downarrow_{[\![A]\!]}[\![a]\!] = a$, and we need to check that $[\![\mathfrak{app}(f,a)]\!]$ is equal to $\uparrow^{\phi}_{[\![B(a)]\!]}[\mathfrak{app}(f,a) \mid \phi \hookrightarrow [\![\mathfrak{app}(f,a)]\!]]$.

$$\begin{aligned}
&[\![\mathfrak{app}(f,a)]\!] \\
&\quad = [\![f]\!][\![a]\!] \\
&\quad = (\uparrow^{\phi}_{[\![\mathsf{M}.\Pi(A,B)]\!]}[f \mid \phi \hookrightarrow [\![f]\!]])[\![a]\!] \\
&\quad = (\uparrow^{\phi}_{\Pi([\![A]\!],\lambda x.[\![B(x)]\!])}[f \mid \phi \hookrightarrow [\![f]\!]])[\![a]\!] \\
&\quad = \uparrow^{\phi}_{[\![B(a)]\!]}\left[\mathfrak{app}(f, \downarrow_{[\![A]\!]}[\![a]\!]) \mid \phi \hookrightarrow [\![f]\!][\![a]\!]\right] \\
&\quad = \uparrow^{\phi}_{[\![B(a)]\!]}[\mathfrak{app}(f,a) \mid \phi \hookrightarrow [\![f]\!][\![a]\!]] \\
&\quad = \uparrow^{\phi}_{[\![B(a)]\!]}[\mathfrak{app}(f,a) \mid \phi \hookrightarrow [\![\mathfrak{app}(f,a)]\!]]
\end{aligned}$$

3) The case for neutral path application $\mathfrak{papp}(p, r)$ is as follows.

$$\begin{aligned}
&[\![\mathfrak{papp}(p,r)]\!] \\
&\quad = [\![p]\!](r) \\
&\quad = (\uparrow^{[\![\mathsf{M}.\mathsf{path}(A,a_0,a_1)]\!]}_{\phi}[p \mid \phi \hookrightarrow [\![p]\!]])(r) \\
&\quad = \uparrow^{\phi \vee_{\mathbb{F}} \partial r}_{[\![A(i)]\!]}[\mathfrak{papp}(p,i) \mid \phi \vee_{\mathbb{F}} \partial r \hookrightarrow [\phi \hookrightarrow [\![p]\!](r), \overline{r = \epsilon \hookrightarrow a_{\epsilon}}]] \\
&\quad = \uparrow^{\phi \vee_{\mathbb{F}} \partial r}_{[\![A(i)]\!]}[\mathfrak{papp}(p,i) \mid \phi \vee_{\mathbb{F}} \partial r \hookrightarrow [\![p]\!](r)] \\
&\quad = \uparrow^{\phi \vee_{\mathbb{F}} \partial r}_{[\![A(i)]\!]}[\mathfrak{papp}(p,i) \mid \phi \vee_{\mathbb{F}} \partial r \hookrightarrow [\![\mathfrak{papp}(p,r)]\!]]
\end{aligned}$$

4) The case for stabilizing a neutral element of the circle is as follows. Starting with a neutral s_0 such that $[\![s_0]\!] =$

$\uparrow^{\phi}_{\mathsf{S1}}[s_0 \mid \phi \hookrightarrow [\![s_0]\!]]$ and a partial normal form s_ϕ such that $\downarrow_{\mathsf{S1}}[\![s_\phi]\!] = s_\phi$, we compute:

$$\begin{aligned}&\downarrow_{\mathsf{S1}}[\![\mathfrak{lift}_\phi(s_0, s_\phi)]\!]\\ &\quad = \downarrow_{\mathsf{S1}}[\![s_0]\!]\\ &\quad = \downarrow_{\mathsf{S1}}\uparrow^{\phi}_{\mathsf{S1}}[s_0 \mid \phi \hookrightarrow [\![s_0]\!]]\\ &\quad = \downarrow_{\mathsf{S1}}\uparrow^{\phi}_{\mathsf{S1}}[s_0 \mid \phi \hookrightarrow [\![s_\phi]\!]]\\ &\quad = \downarrow_{\mathsf{S1}}\mathsf{lift}(\phi, s_0, [\![s_\phi]\!])\\ &\quad = \mathfrak{lift}_\phi(s_0, \downarrow_{\mathsf{S1}}[\![s_\phi]\!])\\ &\quad = \mathfrak{lift}_\phi(s_0, s_\phi)\end{aligned}$$

5) The case for the dependent product type constant is as follows.

$$\begin{aligned}&\Downarrow[\![\mathfrak{pi}(A, B)]\!]\\ &\quad = \Downarrow\Pi([\![A]\!], \lambda x.[\![B(x)]\!])\\ &\quad = \mathfrak{pi}(\Downarrow[\![A]\!], \lambda x.\Downarrow[\![B(x)]\!])\\ &\quad = \mathfrak{pi}(A, B)\end{aligned}$$ □

Corollary 49. *The normal form presentation is* tight*: a given term has only a single normal form.*

Proof. Suppose we have two different normal forms $\mathfrak{a} \neq \mathfrak{a}'$ for a given term a; the normalization function is vertical, so at least one of $\mathfrak{a}, \mathfrak{a}'$ lies outside its image; but this contradicts Lemma 48. □

Corollary 50. *The normalization function defined in Theorem 41 is computed by the algorithm defined in Theorem 45.*

Theorem 51 (Decidability of equality)**.** *Judgmental equality of types and terms in atomic contexts is decidable.*

Proof. First we obtain their unique normal forms using the algorithm described in Theorem 45; using Lemma 44 we may decide whether those normal forms are equal. □

Remark 52. The above results unfold to statements about judgments in atomic contexts (i.e., in the image of $\boldsymbol{\alpha} : \mathscr{A}_\square \longrightarrow \mathscr{T}_\square$), but standard presentations of cubical type theory also allow context extension by $\_ : [\phi]$, written Γ, ϕ. Note however that one can algorithmically eliminate such assumptions, as in the proof of Lemma 44 and the implementations of Cubical Agda and redtt. Simply left invert $\phi \vee \psi$ and perform quantifier elimination on $\forall i.\phi(i)$; under $0 = 1$, all judgments hold; and under a consistent cofibration $r = s$, one can test equality by substitution or by using a union-find algorithm.

Acknowledgments

We thank Mathieu Anel, Steve Awodey, Rafaël Bocquet, Thierry Coquand, Favonia, Marcelo Fiore, Daniel Gratzer, Robert Harper, Kenji Maillard, Anders Mörtberg, Michael Shulman, Thomas Streicher, and Andrea Vezzosi for useful conversations on cubical computability and Artin gluing, and for pointing out mistakes in previous drafts of this paper. We thank Tristan Nguyen at AFOSR for his support.

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

References

[1] A. Abel, "Normalization by evaluation: Dependent types and impredicativity," Habilitation, Ludwig-Maximilians-Universität München, 2013.

[2] T. Altenkirch, P. Dybjer, M. Hofmann, and P. Scott, "Normalization by evaluation for typed lambda calculus with coproducts," in *Proceedings of the 16th Annual IEEE Symposium on Logic in Computer Science*. Washington, DC, USA: IEEE Computer Society, 2001, pp. 303–.

[3] T. Altenkirch and A. Kaposi, "Type theory in type theory using quotient inductive types," in *Proceedings of the 43rd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, ser. POPL '16. St. Petersburg, FL, USA: ACM, 2016, pp. 18–29.

[4] T. Altenkirch, M. Hofmann, and T. Streicher, "Categorical reconstruction of a reduction free normalization proof," in *Category Theory and Computer Science*, D. Pitt, D. E. Rydeheard, and P. Johnstone, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 1995, pp. 182–199.

[5] M. Anel and A. Joyal, "Topo-logie," Mar. 2019, preprint. [Online]. Available: http://mathieu.anel.free.fr/mat/doc/Anel-Joyal-Topo-logie.pdf

[6] C. Angiuli, "Computational semantics of cartesian cubical type theory," Ph.D. dissertation, Carnegie Mellon University, 2019.

[7] C. Angiuli, R. Harper, and T. Wilson, "Computational higher-dimensional type theory," in *POPL 2017: Proceedings of the 44th ACM SIGPLAN Symposium on Principles of Programming Languages*. Paris, France: ACM, 2017, pp. 680–693.

[8] C. Angiuli, K.-B. Hou (Favonia), and R. Harper, "Cartesian Cubical Computational Type Theory: Constructive Reasoning with Paths and Equalities," in *27th EACSL Annual Conference on Computer Science Logic (CSL 2018)*, ser. Leibniz International Proceedings in Informatics (LIPIcs), D. Ghica and A. Jung, Eds., vol. 119. Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2018, pp. 6:1–6:17. [Online]. Available: http://drops.dagstuhl.de/opus/volltexte/2018/9673

[9] C. Angiuli, G. Brunerie, T. Coquand, K.-B. Hou (Favonia), R. Harper, and D. R. Licata, "Syntax and models of cartesian cubical type theory," Feb. 2019, preprint. [Online]. Available: https://github.com/dlicata335/cart-cube

[10] M. Artin, A. Grothendieck, and J.-L. Verdier, *Théorie des topos et cohomologie étale des schémas*. Berlin: Springer-Verlag, 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.

[11] S. Awodey and C. Butz, "Topological completeness for higher-order logic," *The Journal of Symbolic Logic*, vol. 65, no. 3, pp. 1168–1182, 2000. [Online]. Available: http://www.jstor.org/stable/2586693

[12] S. Awodey, "Natural models of homotopy type theory," *Mathematical Structures in Computer Science*, vol. 28, no. 2, pp. 241–286, 2018.

[13] A. Bauer, "First steps in synthetic computability theory," *Electronic Notes in Theoretical Computer Science*, vol. 155, pp. 5–31, 2006, proceedings of the 21st Annual Conference on Mathematical Foundations of Programming Semantics (MFPS XXI).

[14] L. Birkedal, R. E. Møgelberg, J. Schwinghammer, and K. Stovring, "First steps in synthetic guarded domain theory: Step-indexing in the topos of trees," in *Proceedings of the 2011 IEEE 26th Annual Symposium on Logic in Computer Science*. Washington, DC, USA: IEEE Computer Society, 2011, pp. 55–64.

[15] I. Blechschmidt, "Using the internal language of toposes in algebraic geometry," Ph.D. dissertation, Universität Augsberg, 2017.

[16] M. Bunge and J. Funk, *Singular coverings of toposes*, ser. Lecture notes in mathematics, 1890. Berlin: Springer, 2006.

[17] M. Bunge, F. Gago, and A. M. San Luis, *Synthetic Differential Topology*, ser. London Mathematical Society Lecture Note Series. Cambridge University Press, 2018.

[18] A. Carboni and P. Johnstone, "Connected limits, familial representability and artin glueing," *Mathematical Structures in Computer Science*, vol. 5, no. 4, pp. 441–459, 1995.

[19] C. Cohen, T. Coquand, S. Huber, and A. Mörtberg, "Cubical Type Theory: a constructive interpretation of the univalence axiom," *IfCoLog Journal of Logics and their Applications*, vol. 4, no. 10, pp. 3127–3169, Nov. 2017. [Online]. Available: http://www.collegepublications.co.uk/journals/ifcolog/?00019

[20] T. Coquand, "Canonicity and normalization for dependent type theory," *Theoretical Computer Science*, vol. 777, pp. 184–191, 2019, in memory of Maurice Nivat, a founding father of Theoretical Computer Science - Part I.

[21] T. Coquand, S. Huber, and A. Mörtberg, "On higher inductive types in cubical type theory," in *Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science*. Oxford, United Kingdom: ACM, 2018, pp. 255–264.

[22] T. Coquand, S. Huber, and C. Sattler, "Homotopy canonicity for cubical type theory," in *4th International Conference on Formal Structures for Computation and Deduction (FSCD 2019)*, ser. Leibniz International Proceedings in Informatics (LIPIcs), H. Geuvers, Ed., vol. 131. Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2019.

[23] R. L. Crole, *Categories for Types*, ser. Cambridge Mathematical Textbooks. New York: Cambridge University Press, 1993.

[24] R. Diaconescu, "Change of base for toposes with generators," *Journal of Pure and Applied Algebra*, vol. 6, no. 3, pp. 191–218, 1975.

[25] P. Dybjer, "Internal type theory," in *Types for Proofs and Programs: International Workshop, TYPES '95 Torino, Italy, June 5–8, 1995 Selected Papers*, S. Berardi and M. Coppo, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 1996, pp. 120–134.

[26] M. Fiore, "Semantic analysis of normalisation by evaluation for typed lambda calculus," in *Proceedings of the 4th ACM SIGPLAN International Conference on Principles and Practice of Declarative Programming*, ser. PPDP '02. Pittsburgh, PA, USA: ACM, 2002, pp. 26–37.

[27] P. Freyd, "On proving that **1** is an indecomposable projective in various free categories," 1978, unpublished manuscript.

[28] D. Gratzer and J. Sterling, "Syntactic categories for dependent type theory: sketching and adequacy," 2020.

[29] D. Gratzer, G. A. Kavvos, A. Nuyts, and L. Birkedal, "Multimodal dependent type theory," in *Proceedings of the 35th Annual ACM/IEEE Symposium on Logic in Computer Science*. Saarbrücken, Germany: Association for Computing Machinery, 2020, pp. 492–506.

[30] R. Harper and D. R. Licata, "Mechanizing metatheory in a logical framework," *Journal of Functional Programming*, vol. 17, no. 4-5, pp. 613–673, Jul. 2007.

[31] R. Harper, F. Honsell, and G. Plotkin, "A framework for defining logics," *J. ACM*, vol. 40, no. 1, pp. 143–184, Jan. 1993.

[32] L. Henkin, "Completeness in the theory of types," *The Journal of Symbolic Logic*, vol. 15, no. 2, pp. 81–91, 1950. [Online]. Available: http://www.jstor.org/stable/2266967

[33] M. Hofmann and T. Streicher, "Lifting Grothendieck universes," 1997, unpublished note. [Online]. Available: https://www2.mathematik.tu-darmstadt.de/~streicher/NOTES/lift.pdf

[34] S. Huber, "Canonicity for cubical type theory," *Journal of Automated Reasoning*, Jun. 2018.

[35] J. M. E. Hyland, "First steps in synthetic domain theory," in *Category Theory*, A. Carboni, M. C. Pedicchio, and G. Rosolini, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 1991, pp. 131–156.

[36] P. T. Johnstone, *Sketches of an Elephant: A Topos Theory Compendium: Volumes 1 and 2*, ser. Oxford Logical Guides. Oxford Science Publications, 2002, no. 43.

[37] A. Jung and J. Tiuryn, "A new characterization of lambda definability," in *Typed Lambda Calculi and Applications*, M. Bezem and J. F. Groote, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 1993, pp. 245–257.

[38] A. Kaposi, "Type theory in a type theory with quotient inductive types," Ph.D. dissertation, University of Not-

tingham, 2017.

[39] A. Kaposi, S. Huber, and C. Sattler, "Gluing for type theory," in *4th International Conference on Formal Structures for Computation and Deduction (FSCD 2019)*, ser. Leibniz International Proceedings in Informatics (LIPIcs), H. Geuvers, Ed., vol. 131. Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2019.

[40] C. Kapulkin and C. Sattler, "Homotopy canonicity of homotopy type theory," Aug. 2019, slides from a talk given at the International Conference on Homotopy Type Theory (HoTT 2019). [Online]. Available: https://hott.github.io/HoTT-2019/conf-slides/Sattler.pdf

[41] A. Kock, *Synthetic Differential Geometry*, 2nd ed., ser. London Mathematical Society Lecture Note Series. Cambridge University Press, 2006.

[42] ——, *Synthetic Geometry of Manifolds*, ser. Cambridge Tracts in Mathematics. Cambridge University Press, 2009.

[43] F. W. Lawvere, "Toward the description in a smooth topos of the dynamically possible motions and deformations of a continuous body," *Cahiers de Topologie et Géométrie Différentielle Catégoriques*, vol. 21, no. 4, pp. 377–392, 1980.

[44] D. R. Licata, I. Orton, A. M. Pitts, and B. Spitters, "Internal universes in models of homotopy type theory," in *3rd International Conference on Formal Structures for Computation and Deduction, FSCD 2018, July 9-12, 2018, Oxford, UK*, 2018, pp. 22:1–22:17.

[45] C. Newstead, "Algebraic models of dependent type theory," Ph.D. dissertation, Carnegie Mellon University, 2018.

[46] B. Nordström, K. Peterson, and J. M. Smith, *Programming in Martin-Löf's Type Theory*, ser. International Series of Monographs on Computer Science. NY: Oxford University Press, 1990, vol. 7.

[47] I. Orton and A. M. Pitts, "Axioms for modelling cubical type theory in a topos," in *25th EACSL Annual Conference on Computer Science Logic, CSL 2016, August 29 - September 1, 2016, Marseille, France*, 2016, pp. 24:1–24:19.

[48] F. Pfenning and C. Schürmann, "System description: Twelf — a meta-logical framework for deductive systems," in *Automated Deduction — CADE-16*. Berlin, Heidelberg: Springer Berlin Heidelberg, 1999, pp. 202–206.

[49] T. RedPRL Development Team, "redtt," 2018. [Online]. Available: https://www.github.com/RedPRL/redtt

[50] E. Riehl and M. Shulman, "A type theory for synthetic ∞-categories," *Higher Structures*, vol. 1, 2017.

[51] E. Rijke, M. Shulman, and B. Spitters, "Modalities in homotopy type theory," *Logical Methods in Computer Science*, vol. Volume 16, Issue 1, Jan. 2020. [Online]. Available: https://lmcs.episciences.org/6015

[52] M. Shulman, "Univalence for inverse diagrams and homotopy canonicity," *Mathematical Structures in Computer Science*, vol. 25, no. 5, pp. 1203–1277, 2015.

[53] ——, "Internalizing the external, or the joys of codiscreteness," 2011. [Online]. Available: https://golem.ph.utexas.edu/category/2011/11/internalizing_the_external_or.html

[54] J. Sterling, "Objective Metatheory of (Cubical) Type Theories," 2020, Thesis Proposal. [Online]. Available: http://www.jonmsterling.com/pdfs/proposal-slides.pdf

[55] J. Sterling and D. Gratzer, "Lectures on Synthetic Tait Computability," 2020, notes on a lecture given by Sterling in Summer 2020.

[56] J. Sterling and R. Harper, "Logical Relations As Types: Proof-Relevant Parametricity for Program Modules," 2020, under review.

[57] J. Sterling, C. Angiuli, and D. Gratzer, "Cubical syntax for reflection-free extensional equality," in *4th International Conference on Formal Structures for Computation and Deduction (FSCD 2019)*, ser. Leibniz International Proceedings in Informatics (LIPIcs), H. Geuvers, Ed., vol. 131. Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2019, pp. 31:1–31:25. [Online]. Available: http://drops.dagstuhl.de/opus/volltexte/2019/10538

[58] ——, "A cubical language for Bishop sets," 2020, under review.

[59] T. Streicher, "Categorical intuitions underlying semantic normalisation proofs," in *Preliminary Proceedings of the APPSEM Workshop on Normalisation by Evaluation*, O. Danvy and P. Dybjer, Eds. Department of Computer Science, Aarhus University, 1998.

[60] W. W. Tait, "Intensional Interpretations of Functionals of Finite Type I," *The Journal of Symbolic Logic*, vol. 32, no. 2, pp. 198–212, 1967. [Online]. Available: http://www.jstor.org/stable/2271658

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

[62] T. Uemura, "Fibred fibration categories," in *Proceedings of the 32nd Annual ACM/IEEE Symposium on Logic in Computer Science*. Reykjavik, Iceland: IEEE Press, 2017, pp. 24:1–24:12. [Online]. Available: http://dl.acm.org/citation.cfm?id=3329995.3330019

[63] ——, "A general framework for the semantics of type theory," 2019.

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

[65] A. Vezzosi, A. Mörtberg, and A. 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*, ser. ICFP '19. Boston, Massachusetts, USA: ACM, 2019. [Online]. Available: http://www.cs.cmu.edu/~amoertbe/

papers/cubicalagda.pdf

[66] S. Vickers, *Locales and Toposes as Spaces*. Dordrecht: Springer Netherlands, 2007, pp. 429–496.

[67] D. Yetter, "On right adjoints to exponential functors," *Journal of Pure and Applied Algebra*, vol. 45, no. 3, pp. 287–304, 1987. [Online]. Available: http://www.sciencedirect.com/science/article/pii/0022404987900776

Appendix

A. Normalization structures

In Figs. 9 to 11, we present the remaining normalization structures that could not fit in the main body of Section III.

B. Explicit computations

In Section IV-E we gave a presentation of $\mathscr{E}$ as a category of presheaves, an apparently necessary step to substantiate the strict universes of synthetic Tait computability; it is still useful, however, to gain intuitions for the more traditional presentation of $\mathscr{E}$ as the comma category $\mathrm{Sh}(\mathsf{A}_\square) \downarrow \boldsymbol{\alpha}^*$, and to understand the explicit computations of the inverse image and direct image parts of the open and closed immersions respectively.

Computation 53 (Comma category)**.** An object of the comma category $\mathscr{E} \simeq \mathrm{Sh}(\mathsf{A}_\square) \downarrow \boldsymbol{\alpha}^*$ is an object $E : \mathrm{Pr}(\mathscr{T}_\square)$ together with a family of presheaves $E' \longrightarrow \boldsymbol{\alpha}^* E : \mathrm{Pr}(\mathscr{A}_\square)$; a morphism from $(F, F' \longrightarrow \boldsymbol{\alpha}^* F)$ to $(E, E' \longrightarrow \boldsymbol{\alpha}^* E)$ is a morphism $e : F \longrightarrow E : \mathscr{T}_\square$ together with a commuting square of the following kind:

$$\begin{array}{ccc} F' & \xrightarrow{e'} & E \\ \downarrow & & \downarrow \\ \boldsymbol{\alpha}^* F & \xrightarrow[\boldsymbol{\alpha}^* e]{} & \boldsymbol{\alpha}^* E \end{array}$$

Computation 54 (Open immersion)**.** The open immersion $\boldsymbol{j} : \mathsf{T}_\square \hookrightarrow \mathsf{G}_\square$ corresponds under inverse image to the *codomain* fibration $\boldsymbol{j}^* : \mathrm{Pr}(\mathscr{A}_\square) \downarrow \boldsymbol{\alpha}^* \longrightarrow \mathrm{Pr}(\mathscr{T}_\square)$. Hence we may compute the adjunction $\boldsymbol{j}^* \dashv \boldsymbol{j}_*$ as follows:

$$\begin{aligned} \boldsymbol{j}^* &: (E, E' \longrightarrow \boldsymbol{\alpha}^* E) \longmapsto E \\ \boldsymbol{j}_* &: E \longmapsto (E, \boldsymbol{\alpha}^* E \longrightarrow \boldsymbol{\alpha}^* E) \end{aligned}$$

The direct image functor $\boldsymbol{j}_*$ is fully faithful. In fact, we have two additional adjoints $\boldsymbol{j}_! \dashv \boldsymbol{j}^* \dashv \boldsymbol{j}_* \dashv \boldsymbol{j}^!$, the exceptional right adjoint by virtue of the adjunction $\boldsymbol{\alpha}^* \dashv \boldsymbol{\alpha}_*$.

$$\begin{aligned} \boldsymbol{j}_! &: E \longmapsto (E, \mathbf{0} \longrightarrow \boldsymbol{\alpha}^* E) \\ \boldsymbol{j}^! &: (E, E' \longrightarrow \boldsymbol{\alpha}^* E) \longmapsto \boldsymbol{\alpha}_* E' \times_{\boldsymbol{\alpha}_* \boldsymbol{\alpha}^* E} E \end{aligned}$$

Computation 55 (Closed immersion)**.** The closed immersion $\boldsymbol{i} : \mathsf{A}_\square \hookrightarrow \mathsf{G}_\square$ corresponds under inverse image to the *domain* functor $\boldsymbol{i}^* : \mathrm{Sh}(\mathsf{A}_\square) \downarrow \boldsymbol{\alpha}^* \longrightarrow \mathrm{Pr}(\mathscr{A}_\square)$. We compute the adjunction $\boldsymbol{i}^* \dashv \boldsymbol{i}_*$ as follows:

$$\begin{aligned} \boldsymbol{i}^* &: (E, E' \longrightarrow \boldsymbol{\alpha}^* E) \longmapsto E' \\ \boldsymbol{i}_* &: E' \longmapsto (\mathbf{1}, E' \longrightarrow \boldsymbol{\alpha}^* \mathbf{1}) \end{aligned}$$

Because $\boldsymbol{\alpha}$ is an *essential* morphism of topoi, we have an additional left adjoint $\boldsymbol{i}_! \dashv \boldsymbol{i}^* \dashv \boldsymbol{i}_*$:

$$\boldsymbol{i}_! : E' \longmapsto (\boldsymbol{\alpha}_! E', E' \longrightarrow \boldsymbol{\alpha}^* \boldsymbol{\alpha}_! E')$$

Computation 56 (Open and closed modality)**.** Using the syntactic open syn (Computation 27), we can compute the open modality $\bigcirc = \boldsymbol{j}_* \boldsymbol{j}^*$ as the exponential $(\mathsf{syn} \to -)$. Likewise, syn makes another computation of the closed modality $\bullet = \boldsymbol{i}_* \boldsymbol{i}^*$ available:

$$\begin{array}{ccc} A \times \mathsf{syn} & \longrightarrow & \mathsf{syn} \\ \downarrow & & \downarrow \\ A & \longrightarrow & \bullet A \end{array}$$

Corollary 57. *From Computations 27 and 53 to 55 we may make the following observations:*

1) *The open modality $\bigcirc := \boldsymbol{j}_* \boldsymbol{j}^*$ on $\mathscr{E}$ has both a right adjoint $\boldsymbol{j}_* \boldsymbol{j}^!$ and left adjoint $\boldsymbol{j}_! \boldsymbol{j}^*$; hence the syntactic open $\mathsf{syn} : \mathscr{E}$ is a tiny object; the adjunction $\boldsymbol{j}_! \boldsymbol{j}^* \dashv \bigcirc$ can be computed as $(- \times \mathsf{syn}) \dashv (\mathsf{syn} \to -)$.*
2) *The closed modality $\bullet := \boldsymbol{i}_* \boldsymbol{i}^*$ on $\mathscr{E}$ has a left adjoint $\boldsymbol{i}_! \boldsymbol{i}^*$.*
3) *The gluing functor $\boldsymbol{\alpha}^* : \mathrm{Pr}(\mathscr{T}_\square) \longrightarrow \mathrm{Pr}(\mathscr{A}_\square)$ can be reconstructed as the composite $\boldsymbol{i}^* \boldsymbol{j}_*$.*

The following fracture theorem is from SGA 4 [10].

Lemma 58 (Fracture [10, 51])**.** *Any sheaf on $\mathsf{G}_\square$ can be reconstructed up to isomorphism from its restriction to $\mathsf{T}_\square$ and $\mathsf{A}_\square$; in particular, the following square is cartesian for any $A : \mathscr{E}$:*

$$\begin{array}{ccc} A & \longrightarrow & \bullet A \\ \downarrow & & \downarrow \\ \bigcirc A & \longrightarrow & \bullet \bigcirc A \end{array}$$

C. Explicit construction of neutral and normal forms

Our construction of the computability model of cubical type theory (Theorem 20) requires only that certain constants corresponding to the neutral and normal forms exist in $\mathscr{E}$. However, to use this computability model to establish the injectivity and (external) decidability properties of Section V, it is important to ensure that the corresponding properties hold for our normal forms.

Concretely, we define them by a family of indexed quotient inductive types (QITs [3]) valued in the modal universe $\mathscr{U}_\bullet$:

$$\begin{array}{ll} [\mathsf{tp} \ni_{\mathsf{nf}} A] : \mathscr{U}_\bullet & (A : \mathsf{M.tp}) \\ [A \ni_{\mathsf{nf}} a] : \mathscr{U}_\bullet & (A : \mathsf{M.tp}, a : \mathsf{M.tm}(A)) \\ [a \in^{\phi}_{\mathsf{ne}} A] : \mathscr{U}_\bullet & (\phi : \mathbb{F}, A : \mathsf{M.tp}, a : \mathsf{M.tm}(A)) \end{array}$$

In fact, we ensure that $[a \in^{\phi}_{\mathsf{ne}} A]$ is not only $\bigcirc$-connected but actually $(\phi \vee \mathsf{syn})$-connected, capturing the sense in which the data of a neutral form collapses to a point on its locus of instability. Then, the collections of normal and neutral forms

$$\mathfrak{sg} : \{(\textstyle\sum_{A:\mathsf{nftp}}\prod_{x:\mathsf{var}(A)}\mathsf{nftp}) \to \mathsf{nftp} \mid \mathsf{syn} \hookrightarrow \mathsf{M}.\Sigma\}$$
$$\mathfrak{pair} : \{A, B\}\ \{(\textstyle\sum_{x:\mathsf{nf}(A)}\mathsf{nf}(B(x))) \to \mathsf{nf}(\mathsf{M}.\Sigma(A,B)) \mid \mathsf{syn} \hookrightarrow \lambda(a,b).(a,b)\}$$
$$\mathfrak{split} : \{\phi, A, B\}\ \{\mathsf{ne}_\phi(\mathsf{M}.\Sigma(A,B)) \to \textstyle\sum_{x:\mathsf{ne}_\phi(A)}\mathsf{ne}_\phi(B(x)) \mid \mathsf{syn} \hookrightarrow \lambda(a,b).(a,b)\}$$

$$\Sigma : \{(\textstyle\sum_{A:\mathsf{tp}}(A \to \mathsf{tp})) \to \mathsf{tp} \mid \mathsf{syn} \hookrightarrow \mathsf{M}.\Sigma\}$$
$$[\Sigma(A,B)] \cong \textstyle\sum_{x:A}B(x)$$
$$\mathsf{hcom}^{r\to s;\phi}_{\Sigma(A,B)}p = \textbf{let } x(k) = \mathsf{hcom}^{r\to k;\phi}_{A}\lambda i.[i = r \vee_{\mathbb{F}} \phi \to \pi_1(p(i))] \textbf{ in } (x(s), \mathsf{com}^{r\to s;\phi}_{\lambda i.B(x(i))}\lambda i.[i = r \vee_{\mathbb{F}} \phi \to \pi_2(p(i))])$$
$$\mathsf{coe}^{r\to s}_{\lambda i.\Sigma(A(i),B(i))}p = \textbf{let } x(k) = \mathsf{coe}^{r\to k}_{A}\pi_1(p) \textbf{ in } (x(s), \mathsf{coe}^{r\to s}_{\lambda i.B(x(i))}\pi_2(p))$$
$$\Downarrow\Sigma(A,B) = \mathfrak{sg}(\Downarrow A, \lambda x.\Downarrow B(\uparrow^{\bot_{\mathbb{F}}}_{A}[\mathfrak{var}(x) \mid \bot_{\mathbb{F}} \hookrightarrow []_{\mathsf{M.tm}(A)}]))$$
$$\uparrow^{\phi}_{\Sigma(A,B)}[p_0 \mid \phi \hookrightarrow p] = \textbf{let } (x_0, y_0) = \mathfrak{split}(p_0); \tilde{x} = \uparrow^{\phi}_{A}[x_0 \mid \phi \hookrightarrow \pi_1(p)] \textbf{ in } (\tilde{x}, \uparrow^{\phi}_{B(\tilde{x})}[y_0 \mid \phi \hookrightarrow \pi_2(p)])$$
$$\downarrow_{\Sigma(A,B)}p = \mathfrak{pair}(\downarrow_A\pi_1(p), \downarrow_{B(\pi_1(p))}\pi_2(p))$$

Fig. 9. The cubical normalization structure for dependent sum types.

$$\mathfrak{s1} : \{\mathsf{nftp} \mid \mathsf{syn} \hookrightarrow \mathsf{M.S1}\}$$
$$\mathfrak{base} : \{\mathsf{nf}(\mathsf{M.S1}) \mid \mathsf{syn} \hookrightarrow \mathsf{M.base}\}$$
$$\mathfrak{loop} : \{\textstyle\prod_{i:\mathbb{I}}\{\mathsf{nf}(\mathsf{M.S1}) \mid \partial i \hookrightarrow \mathfrak{base}\} \mid \mathsf{syn} \hookrightarrow \mathsf{M.loop}\}$$
$$\mathfrak{fhcom} : \{\mathsf{HCom}(\mathsf{M.S1}, \mathsf{nf}(\mathsf{M.S1})) \mid \mathsf{syn} \hookrightarrow \mathsf{M.hcom}_{\mathsf{M.S1}}\}$$
$$\mathfrak{inds1} : \{\textstyle\prod_{C:\mathsf{var}(\mathsf{M.S1})\to\mathsf{nftp}}\prod_{b:\mathsf{nf}(C(\mathsf{M.base}))}\prod_{l:\prod_{i:\mathbb{I}}\{\mathsf{nf}(C(\mathsf{M.loop}(i)))\mid\partial i\hookrightarrow b\}}\prod_{x:\mathsf{ne}_\phi(\mathsf{M.S1})}\mathsf{ne}_\phi(C(x)) \mid \mathsf{syn} \hookrightarrow \mathsf{M.ind}_{\mathsf{S1}}\}$$
$$\mathfrak{lift} : \{\phi\}\ \{(\textstyle\sum_{b:\mathsf{ne}_\phi(\mathsf{M.S1})}\prod_{\_:[\phi]}\{\mathsf{nf}(\mathsf{M.S1}) \mid \mathsf{syn} \hookrightarrow b\}) \to \mathsf{nf}(\mathsf{M.S1}) \mid \mathsf{syn} \vee \phi \hookrightarrow \lambda(b,b').[\mathsf{syn} \hookrightarrow b, \phi \hookrightarrow b']\}$$

The following definition of S1 is justified by realignment: the constructors give rise to a (quotient-inductive) type that is isomorphic to M.tm(M.S1) underneath z : syn; after realignment, the constructor $\lfloor - \rfloor$ becomes the identity function.

data $[\mathsf{S1}] : \{\mathscr{U} \mid \mathsf{syn} \hookrightarrow \mathsf{M.tm}(\mathsf{M.S1})\}$ **where**
$\quad\lfloor - \rfloor : \{\mathsf{syn}\} \to \mathsf{M.tm}(\mathsf{M.S1}) \to [\mathsf{S1}]$
$\quad\mathsf{base} : \{[\mathsf{S1}] \mid \mathsf{syn} \hookrightarrow \lfloor \mathsf{M.base} \rfloor\}$
$\quad\mathsf{loop} : \prod_{i:\mathbb{I}}\{[\mathsf{S1}] \mid [\partial i] \hookrightarrow \mathsf{base}, \mathsf{syn} \hookrightarrow \lfloor \mathsf{M.loop}(i) \rfloor\}$
$\quad\mathsf{lift} : \prod_{\phi:\mathbb{F}}\prod_{x_0:\mathsf{ne}_\phi(\mathsf{M.S1})}\prod_{x:[\phi]\to\{[\mathsf{S1}]\mid\mathsf{syn}\hookrightarrow\lfloor x_0\rfloor\}}\{[\mathsf{S1}] \mid [\phi] \hookrightarrow x, \mathsf{syn} \hookrightarrow \lfloor x_0 \rfloor\}$
$\quad\mathsf{fhcom} : \prod_{r,s:\mathbb{I}}\prod_{\phi:\mathbb{F}}\prod_{c:\prod_{i:\mathbb{I}}\prod_{\_:[i=r\vee_{\mathbb{F}}\phi]}\sum_{x:\mathsf{M.tm}(\mathsf{M.S1})}\{[\mathsf{S1}]\mid\mathsf{syn}\hookrightarrow\lfloor x\rfloor\}}\{[\mathsf{S1}] \mid [r = s \vee_{\mathbb{F}} \phi] \hookrightarrow \pi_2(c(i)), \mathsf{syn} \hookrightarrow \mathsf{M.hcom}^{r\to s;\phi}_{\mathsf{M.S1}}(\lambda i.\pi_1(c(i)))\}$

$$\mathsf{S1} : \{\mathsf{tp} \mid \mathsf{syn} \hookrightarrow \mathsf{M.S1}\}$$
$$\mathsf{hcom}^{r\to s;\phi}_{\mathsf{S1}}c = \mathsf{fhcom}(r, s, \phi, \lambda i.[i = r \vee_{\mathbb{F}} \phi \hookrightarrow (c(i), c(i))])$$
$$\mathsf{coe}^{r\to s}_{\lambda\_.\mathsf{S1}}x = x$$
$$\uparrow^{\phi}_{\mathsf{S1}}[x_0 \mid \phi \hookrightarrow x] = (x_0, \mathsf{lift}(\phi, x_0, [\phi \hookrightarrow x]))$$
$$\downarrow_{\mathsf{S1}}\lfloor c \rfloor = c$$
$$\downarrow_{\mathsf{S1}}\mathsf{base} = \mathfrak{base}$$
$$\downarrow_{\mathsf{S1}}\mathsf{loop}(i) = \mathfrak{loop}(i)$$
$$\downarrow_{\mathsf{S1}}\mathsf{lift}(\phi, x_0, x) = \mathfrak{lift}_\phi[x_0 \mid \phi \hookrightarrow \downarrow_{\mathsf{S1}}x]$$
$$\downarrow_{\mathsf{S1}}\mathsf{fhcom}(r, s, \phi, c) = \mathfrak{fhcom}^{r\to s;\phi}\lambda i.[i = r \vee_{\mathbb{F}} \phi \hookrightarrow \downarrow_{\mathsf{S1}}\pi_2(c(i))]$$

Fig. 10. Definition of the cubical normalization structure for the circle.

$$\mathsf{ind_{S1}} : \left\{ \prod_{C:\mathsf{S1}\to\mathsf{tp}} \prod_{b:C(\mathsf{base})} \prod_{l:\prod_{i:\mathbb{I}}\{C(\mathsf{loop}(i))\mid\partial i\hookrightarrow b\}} \prod_{x:\mathsf{S1}} C(x) \;\middle|\; \mathsf{syn} \hookrightarrow \mathsf{M.ind_{S1}} \right\}$$
$$\begin{aligned}
\mathsf{ind_{S1}}(C,b,l,\lfloor c\rfloor) &= \mathsf{M.ind_{S1}}(C,b,l,c)\\
\mathsf{ind_{S1}}(C,b,l,\mathsf{base}) &= b\\
\mathsf{ind_{S1}}(C,b,l,\mathsf{loop}(i)) &= l(i)\\
\mathsf{ind_{S1}}(C,b,l,\mathsf{lift}(\phi,x_0,x)) &= \textbf{let } \tilde{x} = \Uparrow^{\phi}_{\mathsf{S1}}[x_0 \mid \phi \hookrightarrow (x_0,x)] \textbf{ in}\\
&\ \textbf{let } c_0 = \mathfrak{inds1}\big(\lambda y.\Downarrow C\big(\Uparrow^{\perp_{\mathbb{F}}}_{\mathsf{S1}}[\mathfrak{var}(y) \mid \perp_{\mathbb{F}} \hookrightarrow []_{\mathsf{tm(S1)}}]\big), \downarrow_{C(\mathsf{base})} b, \lambda i.\downarrow_{C(\mathsf{loop}(i))} l(i), x_0\big) \textbf{ in}\\
&\ \Uparrow^{\phi}_{C(\tilde{x})}[c_0 \mid \phi \hookrightarrow \mathsf{ind_{S1}}(C,b,l,\tilde{x})]\\
\mathsf{ind_{S1}}(C,b,l,\mathsf{fhcom}(r,s,\phi,x)) &= \mathsf{com}^{r\to s;\phi}_{\lambda i.C(\mathsf{hcom}^{r\to i;\phi}_{\mathsf{S1}}x)} \lambda i.[i=r \vee_{\mathbb{F}} \phi \hookrightarrow \mathsf{ind_{S1}}(C,b,l,\pi_2(x(i)))]
\end{aligned}$$

Fig. 11. Implementation of the induction principle for the circle.

are obtained by dependent sum and realignment as follows (noting that the fibers of each family are valued in $\mathscr{U}_\bullet$ and are thus $\bigcirc$-connected):

$$\begin{aligned}
\mathsf{nftp} &\cong \textstyle\sum_{A:\mathsf{M.tp}}[\mathsf{tp} \ni_{\mathsf{nf}} A]\\
\mathsf{nf}(A) &\cong \textstyle\sum_{a:\mathsf{M.tm}(A)}[A \ni_{\mathsf{nf}} a]\\
\mathsf{ne}_\phi(A) &\cong \textstyle\sum_{a:\mathsf{M.tm}(A)}[a \in^{\phi}_{\mathsf{ne}} A]
\end{aligned}$$

Our use of quotienting in the definition of normal forms is to impose correct *cubical boundaries* on constructors: for instance, we must have $\partial i \to \mathfrak{loop}(i) = \mathfrak{base}$. Because the theory of cofibrations is (externally) decidable, the quotient can be presented externally by an effective rewriting system that reduces size and is therefore obviously noetherian.

Remark 59. An indexed quotient inductive type in $\mathscr{U}_\bullet$ also has a universal property in $\mathscr{U}$, obtained by adding an additional quotient-inductive clause that contracts each fiber to a point under $z : \mathsf{syn}$.

In Fig. 12 we present the indexed quotient inductive definition of normal and neutral forms.

Lemma 60 (Injectivity of normal form constructors)**.** *The following formula holds in the internal logic of $\mathscr{E} = \mathrm{Sh}(\mathbf{G}_\square)$:*

$$\forall A, A', B, B'.\quad \mathfrak{pi}(A,B) = \mathfrak{pi}(A',B') \implies \bullet((A,B) = (A',B'))$$

Proof. We prove this in the same way that one proves injectivity of constructors of any inductive type, with one subtlety: we must ensure that our constructions respect the cubical boundary of $\mathfrak{glue}$, which is the only constructor of $[\mathsf{tp} \ni_{\mathsf{nf}} -]$ subject to an equational clause.

By induction on $[\mathsf{tp} \ni_{\mathsf{nf}} -]$, we define a $\bigcirc$-connected predicate $\mathsf{isPi} : \mathsf{nftp} \to \Omega_\bullet$ satisfying a universal property:

$$\forall X : \mathsf{nftp}.\mathsf{isPi}(X) \iff \bullet(\exists \mathfrak{A}, \mathfrak{B}.X = \mathfrak{pi}(\mathfrak{A},\mathfrak{B}))$$

We define isPi as follows:

$$\begin{aligned}
\mathsf{isPi}(_, \mathfrak{pi}(\mathfrak{A},\mathfrak{B})) &= \bullet\top\\
\mathsf{isPi}(_, \mathfrak{glue}\{B,A,f\}(\phi,\mathfrak{B},\mathfrak{A},\mathfrak{f})) &= \bullet\exists z : [\phi].\mathsf{isPi}(A, \mathfrak{A}(z))\\
\mathsf{isPi}(_, \mathfrak{sg}(\mathfrak{A},\mathfrak{B})) &= \bullet\perp\\
\mathsf{isPi}(_, \mathfrak{path}(\mathfrak{A},\mathfrak{a}_0,\mathfrak{a}_1)) &= \bullet\perp\\
\mathsf{isPi}(_, \mathfrak{s}1) &= \bullet\perp
\end{aligned}$$

We must verify that isPi respects the ϕ-boundary of $\mathfrak{glue} = \mathfrak{A}(z)$. But supposing that $z : [\phi]$, we can calculate that $\mathsf{isPi}(\_, \mathfrak{glue}(\phi,\mathfrak{B},\mathfrak{A},\mathfrak{f})) = \mathsf{isPi}(\_, \mathfrak{A}(z))$, which is exactly what it means for isPi to respect that equation.

The reverse implication of the universal property of isPi is immediate. We prove the forward implication by induction on X, where again the only subtlety is in the case of $\mathfrak{glue}$: if $\mathsf{isPi}(\_, \mathfrak{glue}(\phi,\mathfrak{B},\mathfrak{A},\mathfrak{f}))$ then we must have $\mathsf{isPi}(A, \mathfrak{A}(z))$, which by the inductive hypothesis implies $\mathfrak{A}(z) = \mathfrak{pi}$.

Then we define $\mathsf{dom} : \{X : \mathsf{nftp} \mid \mathsf{isPi}(X)\} \to \bullet\mathsf{nftp}$:

$$\begin{aligned}
\mathsf{dom}(_, \mathfrak{pi}(\mathfrak{A},\mathfrak{B})) &= \eta_\bullet(\mathfrak{A})\\
\mathsf{dom}(_, \mathfrak{glue}(\phi,\mathfrak{B},\mathfrak{A},\mathfrak{f})) &= \mathsf{dom}(A, \mathfrak{A})
\end{aligned}$$

Note that these are the only constructors that may satisfy isPi, and in the case of $\mathfrak{glue}$, if $\mathsf{isPi}(\_, \mathfrak{glue}(\phi,\mathfrak{B},\mathfrak{A},\mathfrak{f}))$ holds then $\phi = \top$ and $\mathsf{isPi}(A, \mathfrak{A})$. We may define cod similarly.

Finally, suppose that $\mathfrak{pi}(A,B) = \mathfrak{pi}(A',B')$. Then by applying dom to both sides of this equation, we get $\bullet(A = A')$; the equality of codomains likewise follows by applying cod. □

D. Proofs of theorems

Lemma 23. *The chosen interval structure is preserved by restriction along $\boldsymbol{\alpha} : \mathbf{A}_\square \longrightarrow \mathbf{T}_\square$; that is, we have an isomorphism $\mathsf{y}_{\mathscr{A}_\square}(\cdot.\mathbb{I}) \cong \boldsymbol{\alpha}^*\mathsf{y}_{\mathscr{T}_\square}(\mathbb{I})$.*

Proof. First we compute the representable points of $\boldsymbol{\alpha}^*\mathsf{y}_{\mathscr{T}_\square}(\mathbb{I})$ by transpose and the Yoneda lemma:

$$\begin{aligned}
&\mathrm{Hom}_{\mathrm{Pr}(\mathscr{A}_\square)}(\mathsf{y}_{\mathscr{A}_\square}(\Gamma), \boldsymbol{\alpha}^*\mathsf{y}_{\mathscr{T}_\square}(\mathbb{I}))\\
&\quad\cong \mathrm{Hom}_{\mathrm{Pr}(\mathscr{T}_\square)}(\boldsymbol{\alpha}_!\mathsf{y}_{\mathscr{A}_\square}(\Gamma), \mathsf{y}_{\mathscr{T}_\square}(\mathbb{I}))\\
&\quad\cong \mathrm{Hom}_{\mathrm{Pr}(\mathscr{T}_\square)}(\mathsf{y}_{\mathscr{T}_\square}(\boldsymbol{\alpha}(\Gamma)), \mathsf{y}_{\mathscr{T}_\square}(\mathbb{I}))\\
&\quad\cong \mathrm{Hom}_{\mathscr{T}_\square}(\boldsymbol{\alpha}(\Gamma), \mathbb{I})
\end{aligned}$$

CIRCLE TYPE

$$\frac{}{\mathfrak{s1} : [\mathsf{tp} \ni_{\mathsf{nf}} \mathsf{M.S1}]}$$

PATH TYPE

$$\frac{\mathfrak{A} : \prod_{i:\mathbb{I}}[\mathsf{tp} \ni_{\mathsf{nf}} A(i)] \qquad \mathfrak{a}_0 : [A(0) \ni_{\mathsf{nf}} a_0] \qquad \mathfrak{a}_1 : [A(1) \ni_{\mathsf{nf}} a_1]}{\mathfrak{path}\{A, a_0, a_1\}(\mathfrak{A}, \mathfrak{a}_0, \mathfrak{a}_1) : [\mathsf{tp} \ni_{\mathsf{nf}} \mathsf{M.path}(A, a_0, a_1)]}$$

GLUE TYPE

$$\frac{\phi : \mathbb{F} \qquad \mathfrak{B} : [\mathsf{tp} \ni_{\mathsf{nf}} B] \qquad \mathfrak{A} : \prod_{z:[\phi]}[\mathsf{tp} \ni_{\mathsf{nf}} A(z)] \qquad \mathfrak{f} : \prod_{z:[\phi]}[\mathsf{M.Equiv}(A(z), B) \ni_{\mathsf{nf}} f(z)]}{\mathfrak{glue}\{B, A, f\}(\phi, \mathfrak{B}, \mathfrak{A}, \mathfrak{f}) : [\mathsf{tp} \ni_{\mathsf{nf}} \mathsf{M.glue}(\phi, B, A, f)]}$$

GLUE TYPE BOUNDARY

$$\frac{\phi : \mathbb{F} \qquad \mathfrak{B} : [\mathsf{tp} \ni_{\mathsf{nf}} B] \qquad \mathfrak{A} : \prod_{z:[\phi]}[\mathsf{tp} \ni_{\mathsf{nf}} A(z)] \qquad \mathfrak{f} : \prod_{z:[\phi]}[\mathsf{M.Equiv}(A(z), B) \ni_{\mathsf{nf}} f(z)] \qquad z : [\phi]}{\mathfrak{glue}\{B, A, f\}(\phi, \mathfrak{B}, \mathfrak{A}, \mathfrak{f}) = \mathfrak{A}(z) : [\mathsf{tp} \ni_{\mathsf{nf}} A(z)]}$$

PI/SG TYPE

$$\frac{\mathfrak{A} : [\mathsf{tp} \ni_{\mathsf{nf}} A] \qquad \mathfrak{B} : \prod_{x:\mathsf{var}(A)}[\mathsf{tp} \ni_{\mathsf{nf}} B(x)]}{\mathfrak{pi}\{A, B\}(\mathfrak{A}, \mathfrak{B}) : [\mathsf{tp} \ni_{\mathsf{nf}} \mathsf{M}.\Pi(A, B)] \qquad \mathfrak{sg}\{A, B\}(\mathfrak{A}, \mathfrak{B}) : [\mathsf{tp} \ni_{\mathsf{nf}} \mathsf{M}.\Sigma(A, B)]}$$

UNSTABLE

$$\frac{z : [\phi]}{\star(z) : [a \in^{\phi}_{\mathsf{ne}} A]}$$

UNSTABLE COLLAPSE

$$\frac{z : [\phi] \qquad \mathfrak{a} : [a \in^{\phi}_{\mathsf{ne}} A]}{\mathfrak{a} = \star(z) : [a \in^{\phi}_{\mathsf{ne}} A]}$$

VARIABLE

$$\frac{x : \mathsf{var}(A)}{\mathfrak{var}\{A\}(x) : [x \in^{\perp}_{\mathsf{ne}} A]}$$

FUNCTION APPLICATION

$$\frac{\mathfrak{f} : [f \in^{\phi}_{\mathsf{ne}} \mathsf{M}.\Pi(A, B)] \qquad \mathfrak{a} : [A \ni_{\mathsf{nf}} a]}{\mathfrak{app}\{A, B, f, a\}(\mathfrak{f}, \mathfrak{a}) : [a \in^{\phi}_{\mathsf{ne}} A]}$$

PAIR PROJECTION

$$\frac{\mathfrak{p} : [p \in^{\phi}_{\mathsf{ne}} \mathsf{M}.\Sigma(A, B)]}{\mathfrak{fst}\{A, B, p\}(\mathfrak{p}) : [\pi_1(p) \in^{\phi}_{\mathsf{ne}} A] \qquad \mathfrak{snd}\{A, B, p\}(\mathfrak{p}) : [\pi_2(p) \in^{\phi}_{\mathsf{ne}} B(\pi_1(p))]}$$

PATH APPLICATION

$$\frac{\mathfrak{p} : [p \in^{\phi}_{\mathsf{ne}} \mathsf{M.path}(A, a_0, a_1)] \qquad r : \mathbb{I}}{\mathfrak{papp}\{A, a_0, a_1\}(\mathfrak{p}, r) : [p(r) \in^{\phi \vee_{\mathbb{F}} \partial r}_{\mathsf{ne}} A(r)]}$$

UNGLUE DESTRUCTOR

$$\frac{\mathfrak{g} : [g \in^{\psi}_{\mathsf{ne}} \mathsf{M.glue}(\phi, B, A, f)]}{\mathfrak{unglue}\{B, A, f, g\}(\phi, \mathfrak{g}) : [\mathsf{M.unglue}(g) \in^{\psi \vee_{\mathbb{F}} \phi}_{\mathsf{ne}} B]}$$

CIRCLE INDUCTION

$$\frac{\mathfrak{C} : \prod_{x:\mathsf{var}(\mathsf{M.S1})}[\mathsf{tp} \ni_{\mathsf{nf}} C(x)] \qquad \mathfrak{b} : [C(\mathsf{M.base}) \ni_{\mathsf{nf}} b] \qquad \mathfrak{l} : \prod_{i:\mathbb{I}}[C(\mathsf{M.loop}(i)) \ni_{\mathsf{nf}} l(i)] \qquad \mathfrak{s} : [s \in^{\phi}_{\mathsf{ne}} \mathsf{M.S1}]}{\mathfrak{ind}\{C, b, l, s\}(\mathfrak{C}, \mathfrak{b}, \mathfrak{l}, \mathfrak{s}) : [\mathsf{M.ind}_{\mathsf{S1}}(C, b, l, s) \in^{\phi}_{\mathsf{ne}} C(s)]}$$

CIRCLE NEUTRAL LIFT

$$\frac{\mathfrak{s}_0 : [s \in^{\phi}_{\mathsf{ne}} \mathsf{M.S1}] \qquad \mathfrak{s}_\phi : \prod_{\_:[\phi]}[\mathsf{M.S1} \ni_{\mathsf{nf}} s]}{\mathfrak{lift}\{s\}(\mathfrak{s}_0, \mathfrak{s}_\phi) : [\mathsf{M.S1} \ni_{\mathsf{nf}} s]}$$

CIRCLE NEUTRAL LIFT BOUNDARY

$$\frac{\mathfrak{s}_0 : [s \in^{\phi}_{\mathsf{ne}} \mathsf{M.S1}] \qquad \mathfrak{s}_\phi : \prod_{\_:[\phi]}[\mathsf{M.S1} \ni_{\mathsf{nf}} s] \qquad z : [\phi]}{\mathfrak{lift}\{s\}(\mathfrak{s}_0, \mathfrak{s}_\phi) = \mathfrak{s}_\phi(z) : [\mathsf{M.S1} \ni_{\mathsf{nf}} s]}$$

CIRCLE BASE

$$\frac{}{\mathfrak{base} : [\mathsf{M.S1} \ni_{\mathsf{nf}} \mathsf{M.base}]}$$

CIRCLE LOOP

$$\frac{r : \mathbb{I}}{\mathfrak{loop}(r) : [\mathsf{M.S1} \ni_{\mathsf{nf}} \mathsf{M.loop}(r)]}$$

CIRCLE LOOP BOUNDARY

$$\frac{r : \mathbb{I} \qquad \_ : [\partial r]}{\mathfrak{loop}(r) = \mathfrak{base} : [\mathsf{M.S1} \ni_{\mathsf{nf}} \mathsf{M.base}]}$$

CIRCLE FORMAL HOMOGENEOUS COMPOSITION

$$\frac{r, s : \mathbb{I} \qquad \phi : \mathbb{F} \qquad \mathfrak{a} : \prod_{i:\mathbb{I}}\prod_{z:[i=r \vee_{\mathbb{F}} \phi]}[\mathsf{M.S1} \ni_{\mathsf{nf}} a(i, z)]}{\mathfrak{fhcom}\{a\}(r, s, \phi, \mathfrak{a}) : [\mathsf{M.S1} \ni_{\mathsf{nf}} \mathsf{M.hcom}(\mathsf{M.S1}, r, s, \phi, a)]}$$

FUNCTION ABSTRACTION

$$\frac{\mathfrak{f} : \prod_{x:\mathsf{var}(A)}[B(x) \ni_{\mathsf{nf}} f(x)]}{\mathfrak{lam}\{A, B, f\}(\mathfrak{f}) : [\mathsf{M}.\Pi(A, B) \ni_{\mathsf{nf}} \lambda x.f(x)]}$$

PAIR CONSTRUCTOR

$$\frac{\mathfrak{a} : [A \ni_{\mathsf{nf}} a] \qquad \mathfrak{b} : [B(a) \ni_{\mathsf{nf}} b]}{\mathfrak{pair}\{A, B, a, b\}(\mathfrak{a}, \mathfrak{b}) : [\mathsf{M}.\Sigma \ni_{\mathsf{nf}} (a, b)]}$$

PATH ABSTRACTION

$$\frac{\mathfrak{p} : \prod_{i:\mathbb{I}}[A(i) \ni_{\mathsf{nf}} p(i)]}{\mathfrak{plam}\{A, a_0, a_1, p\}(\mathfrak{p}) : [\mathsf{M.path}(A, a_0, a_1) \ni_{\mathsf{nf}} \lambda i.p(i)]}$$

ENGLUE CONSTRUCTOR

$$\frac{\mathfrak{a} : \prod_{z:[\phi]}[A \ni_{\mathsf{nf}} a] \qquad \mathfrak{b} : [B \ni_{\mathsf{nf}} b] \qquad \forall z : [\phi].\mathfrak{b} = \mathfrak{a}(z)}{\mathfrak{englue}\{B, A, f, a, b\}(\phi, \mathfrak{a}, \mathfrak{b}) : [\mathsf{M.glue}(\phi, B, A, f) \ni_{\mathsf{nf}} \mathsf{M.glue/tm}(a, b)]}$$

ENGLUE CONSTRUCTOR BOUNDARY

$$\frac{\mathfrak{a} : \prod_{z:[\phi]}[A \ni_{\mathsf{nf}} a] \qquad \mathfrak{b} : [B \ni_{\mathsf{nf}} b] \qquad \forall z : [\phi].\mathfrak{b} = \mathfrak{a}(z) \qquad z : [\phi]}{\mathfrak{englue}\{B, A, f, a, b\}(\phi, \mathfrak{a}, \mathfrak{b}) = \mathfrak{a}(z) : [A \ni_{\mathsf{nf}} a]}$$

Fig. 12. The explicit indexed quotient-inductive definition of normal and neutral forms. The UNSTABLE and UNSTABLE COLLAPSE rules ensure that $\mathsf{ne}_\phi(A)$ collapses to $\mathsf{M.tm}(A)$ within the locus of instability ϕ.

We see by induction on the definition of the objects and hom sets of $\mathscr{A}_{\square}$ that this is equivalent to $\mathrm{Hom}_{\mathscr{A}_{\square}}(\Gamma, \cdot.\mathbb{I})$. ◻

Lemma 31. *Let $f : \mathbf{Y} \longrightarrow \mathbf{U}$ be a morphism of topoi. Write $\mathbf{X}$ for the Artin gluing of the inverse image functor f^*, and write $j : \mathbf{U} \hookrightarrow \mathbf{X}$ and $i : \mathbf{Y} \hookrightarrow \mathbf{X}$ for the respective open and closed immersions of topoi. Suppose that $X : \mathrm{Sh}(\mathbf{X})$ is a sheaf such that j^*X is a tiny object in $\mathrm{Sh}(\mathbf{U})$ and i^*X is a tiny object in $\mathrm{Sh}(\mathbf{Y})$; then X is tiny.*

Proof. We must check that the exponential functor $(X \to -)$ preserves colimits. Fixing a diagram $E_\bullet : I \longrightarrow \mathrm{Sh}(\mathbf{X})$, we may compute the exponential $(X \to \mathrm{colim}_I\, E_\bullet)$ in the language of $\mathrm{Sh}(\mathbf{Y})$ as follows; first, the standard computation that glues a function from the open subtopos onto a function from the closed subtopos [36]:

$$\begin{array}{ccc} i^*(X \to \mathrm{colim}_I\, E_\bullet) & \longrightarrow & i^*X \to i^*\,\mathrm{colim}_I\, E_\bullet \\ \vdots & \lrcorner & \downarrow \\ f^*(j^*X \to j^*\,\mathrm{colim}_I\, E_\bullet) & \longrightarrow & i^*X \to f^*j^*\,\mathrm{colim}_I\, E_\bullet \end{array}$$

Commute cocontinuous functors past colimits.

$$\begin{array}{ccc} i^*(X \to \mathrm{colim}_I\, E_\bullet) & \longrightarrow & i^*X \to \mathrm{colim}_I\, i^*E_\bullet \\ \vdots & \lrcorner & \downarrow \\ f^*(j^*X \to \mathrm{colim}_I\, j^*E_\bullet) & \longrightarrow & i^*X \to \mathrm{colim}_I\, f^*j^*E_\bullet \end{array}$$

Use the tininess of i^*X, j^*X and the cocontinuity of f^*.

$$\begin{array}{ccc} i^*(X \to \mathrm{colim}_I\, E_\bullet) & \longrightarrow & \mathrm{colim}_I\, (i^*X \to i^*E_\bullet) \\ \vdots & \lrcorner & \downarrow \\ \mathrm{colim}_I\, f^*(j^*X \to j^*E_\bullet) & \longrightarrow & \mathrm{colim}_I\, (i^*X \to f^*j^*E_\bullet) \end{array}$$

Hence by the universality of colimits we have:

$$\begin{array}{ccc} \mathrm{colim}_I\, i^*(X \to E_\bullet) & \longrightarrow & \mathrm{colim}_I\, (i^*X \to i^*E_\bullet) \\ \vdots & \lrcorner & \downarrow \\ \mathrm{colim}_I\, f^*(j^*X \to j^*E_\bullet) & \longrightarrow & \mathrm{colim}_I\, (i^*X \to f^*j^*E_\bullet) \end{array}$$ ◻

Lemma 61. *If $\Gamma : \mathscr{A}_{\square}$ is a formal context, then the hom set $\mathrm{Hom}_{\mathscr{T}_{\square}}(\boldsymbol{\alpha}(\Gamma), [0 = 1])$ is empty.*

Proof. Formal contexts do not induce assumptions of false cofibrations. This can be seen by a model construction in which $\mathscr{T}_{\square}$ is interpreted into cubical sets, where the interpretation of cofibrations is standard but each type is interpreted as an *inhabited* type. Such an argument accommodates types that are "weakly empty" (e.g. the void type lacking an η-law), because the abort eliminator can simply return the basepoint of its motive. ◻

Lemma 62. *In the lattice of opens of $\mathbf{G}_{\square}$, we have $\bot_{\mathbb{F}} \leq \mathsf{syn}$.*

Proof. We recall that the interval $\mathbb{I}$ is purely syntactic (Lemma 23), hence we have $\bot_{\mathbb{F}} = \boldsymbol{j}_*\mathsf{y}_{\mathscr{T}_{\square}}([0 = 1])$ and therefore $\boldsymbol{i}^*\bot_{\mathbb{F}} = \boldsymbol{\alpha}^*\mathsf{y}_{\mathscr{T}_{\square}}([0 = 1])$. To show the inequality $\bot_{\mathbb{F}} \leq \mathsf{syn}$, we need to exhibit a square of the following kind in $\mathrm{Pr}(\mathscr{A}_{\square})$:

$$\begin{array}{ccc} \boldsymbol{\alpha}^*\mathsf{y}_{\mathscr{T}_{\square}}([0 = 1]) & \cdots\to & \mathbf{0} \\ {\scriptstyle \bot_{\mathbb{F}}}\downarrow & & \downarrow{\scriptstyle \mathsf{syn}} \\ \boldsymbol{\alpha}^*\mathsf{y}_{\mathscr{T}_{\square}}([0 = 1]) & \longrightarrow & \mathbf{1} \end{array}$$

It suffices to show that $\boldsymbol{\alpha}^*\mathsf{y}_{\mathscr{T}_{\square}}([0 = 1])$ is the initial object of $\mathrm{Pr}(\mathscr{A}_{\square})$, but this follows from Lemma 61. ◻

Lemma 63. *A type $A : \mathscr{U}$ in $\mathscr{E}$ is $\bot_{\mathbb{F}}$-connected if and only $\bigcirc A$ is $\bot_{\mathbb{F}}$-connected.*

Proof. If A is $\bot_{\mathbb{F}}$-connected, it is immediate that $\bigcirc A$ is $\bot_{\mathbb{F}}$-connected. Conversely, assume that $\bigcirc A$ is $\bot_{\mathbb{F}}$-connected; Lemma 62 implies that A is also $\bot_{\mathbb{F}}$-connected. ◻

Lemma 33. *A type $A : \mathscr{U}$ in $\mathscr{E}$ is $\mathbb{F}$-local if and only if $\bigcirc A$ is $\mathbb{F}$-local.*

Proof. For $\bot_{\mathbb{F}}$-connectedness, we use Lemma 63. Then, fixing $\phi, \psi : \mathbb{F}$ we must check that a partial element $[\phi \hookrightarrow a_\phi, \psi \hookrightarrow a_\psi] : [\phi] \vee [\psi] \longrightarrow A$ can be extended to a unique partial element $[\phi \vee_{\mathbb{F}} \psi] \longrightarrow A$.

We assume a proof of $[\phi \vee_{\mathbb{F}} \psi]$; by Construction 10 we have $\mathsf{M}.[\phi\, \mathsf{M}.\vee_{\mathbb{F}}\, \psi]$ and $\bullet([\phi] \vee [\psi])$, which is the same as $[\phi] \vee [\psi] \vee \mathsf{syn}$. Hence we may form the following partial element, using the fact that $\bigcirc A$ is $\mathbb{F}$-local:

$$\left[\begin{array}{l} \phi \hookrightarrow a_\phi \\ \psi \hookrightarrow a_\psi \\ \mathsf{syn} \hookrightarrow [\phi \hookrightarrow a_\phi, \psi \hookrightarrow a_\psi]_{\bigcirc A} \end{array}\right.$$ ◻

Lemma 36. *For any $X : \mathscr{E}$, we have a canonical isomorphism $[(\!|-|\!), X] \cong \boldsymbol{i}^*X : \mathrm{Pr}(\mathscr{A}_{\square})$ determined by adjoint transpose and the Yoneda lemma.*

Proof. Fix an atomic context $\Gamma : \mathscr{A}_{\square}$ and compute:

$$\begin{aligned} [(\!|-|\!), X](\Gamma) &\cong \mathrm{Hom}_{\mathscr{E}}((\!|\Gamma|\!), X) \\ &= \mathrm{Hom}_{\mathscr{E}}(\boldsymbol{i}_!\mathsf{y}_{\mathscr{A}_{\square}}(\Gamma), X) \\ &\cong \mathrm{Hom}_{\mathrm{Pr}(\mathscr{A}_{\square})}(\mathsf{y}_{\mathscr{A}_{\square}}(\Gamma), \boldsymbol{i}^*X) \\ &\cong \boldsymbol{i}^*X(\Gamma) \end{aligned}$$ ◻

Lemma 40. *The pointwise vertical natural transformation $\mathsf{atom} : (\!|-|\!) \longrightarrow [\![-]\!]$ induces by precomposition a vertical map $\mathsf{atom}^*_X : X_{\mathsf{M}'} \longrightarrow X$ for any sheaf $X : \mathscr{E}$.*

Proof. The vertical map is computed in the language of the comma category as the following square:

$$\begin{array}{ccc} [\llbracket - \rrbracket, X] & \xrightarrow{[\mathsf{atom}, X]} & [(\!|-|\!), X] \cong \boldsymbol{i}^* X \\ {\scriptstyle X_{\mathsf{M}'}} \downarrow & & \downarrow {\scriptstyle X} \\ \boldsymbol{\alpha}^* \boldsymbol{j}^* X & \xrightarrow[\boldsymbol{\alpha}^* \mathbf{id}_{\boldsymbol{j}^* X}]{} & \boldsymbol{\alpha}^* \boldsymbol{j}^* X \end{array}$$

To see that the diagram commutes, we chase an element $x : \llbracket \Gamma \rrbracket \longrightarrow X$, using the fact that each component $\mathsf{atom}_\Gamma : (\!|\Gamma|\!) \longrightarrow \llbracket \Gamma \rrbracket$ is vertical. □

Theorem 41 (The normalization function)**.** *The functor* $\mathsf{M}' : \mathscr{T}_{\square} \longrightarrow \mathscr{E}$ *induces a vertical map* $\mathsf{M.tp} \longrightarrow (\mathsf{M'.tp})_{\mathsf{M}'}$. *Composing this with the vertical maps* $\mathsf{atom}^*_{\mathsf{M'.tp}}$ *and* $\Downarrow$*, we obtain a vertical* normalization *map* nbe *sending a syntactic type to the normal form chosen by its normalization structure in* M'*:*

$$\mathsf{M.tp} \dashrightarrow (\mathsf{M'.tp})_{\mathsf{M}'} \xrightarrow{\mathsf{atom}^*_{\mathsf{M'.tp}}} \mathsf{M'.tp} \xrightarrow{\Downarrow} \mathsf{nftp}$$

Proof. Unfolding things more precisely, the vertical map $\mathsf{M.tp} \longrightarrow (\mathsf{M'.tp})_{\mathsf{M}'}$ must be a square of the following form:

$$\begin{array}{ccc} \boldsymbol{\alpha}^* \mathsf{y}_{\mathscr{T}_{\square}}(\mathsf{tp}) & \dashrightarrow & [\llbracket - \rrbracket, \mathsf{M'.tp}] \\ {\scriptstyle \mathsf{M.tp} = \boldsymbol{j}_* \mathsf{y}_{\mathscr{T}_{\square}}(\mathsf{tp})} \downarrow & & \downarrow {\scriptstyle (\mathsf{M'.tp})_{\mathsf{M}'}} \\ \boldsymbol{\alpha}^* \mathsf{y}_{\mathscr{T}_{\square}}(\mathsf{tp}) & \xrightarrow[\boldsymbol{\alpha}^* \mathbf{id}_{\mathsf{y}_{\mathscr{T}_{\square}}(\mathsf{tp})}]{} & \boldsymbol{\alpha}^* \mathsf{y}_{\mathscr{T}_{\square}}(\mathsf{tp}) \end{array}$$

The upstairs map is defined by functoriality of the computability interpretation, taking a type $A : \boldsymbol{\alpha}(\Gamma) \longrightarrow \mathsf{tp}$ to its chosen normalization structure $\mathsf{M}'(A) : \llbracket \Gamma \rrbracket \longrightarrow \mathsf{M'.tp}$. □