NORMALIZATION FOR MULTIMODAL TYPE THEORY

DANIEL GRATZER

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

ABSTRACT. We prove normalization for MTT, a general multimodal dependent type theory capable of expressing modal type theories for guarded recursion, internalized parametricity, and various other prototypical modal situations. We prove that deciding type checking and conversion in MTT can be reduced to deciding the equality of modalities in the underlying modal situation, immediately yielding a type checking algorithm for all instantiations of MTT in the literature. This proof uses a generalization of *synthetic Tait computability*—an abstract approach to gluing proofs—to account for modalities. This extension is based on MTT itself, so that this proof also constitutes a significant case study of MTT.

1. INTRODUCTION

If type theory is classically the study of objects invariant under change of context, modal type theory is the study of adding non-invariant connectives—*modalities*—to type theory. Given that many natural features of particular models of type theory are not invariant under substitution, modal type theories have sparked considerable interest. By nature, however, modal type theories must thread the needle of presenting modalities in such a way that the classical substitution theorems of type theory still hold.

Typically, modal type theories require modifications to the apparatus of contexts and substitutions. Unfortunately, these tweaks are often more art than science, with expert attention required even to make the most trivial modification to the modal structure of a type theory. In order to address this complexity, *general* modal type theories have been introduced [LSR17, GKNB20a]. These theories can be instantiated by a description of a modal situation to produce a system enjoying the theorems usually proved by experts.

1.1. **Multimodal type theory.** We focus on one such general modal type theory: MTT [GKNB20a]. MTT can be instantiated with an arbitrary collection of modalities and transformations between them to yield a highly usable syntax. The modalities in MTT behave like (weak) dependent right adjoints (DRAs) [BCM+20] so that MTT can be used to internalize nearly any right adjoint. This flexibility allows MTT to encode calculi for guarded recursion, internalized parametricity, and other handcrafted calculi.

More precisely, MTT can be instantiated by a *mode theory*, a strict 2-category describing modes, modalities, and natural transformations between these modalities. This 2-categorical structure is then reflected into the structure of substitutions in MTT, ensuring that e.g., a transformation between two modalities μ and ν gives rise to a function $\langle \mu \mid A \rangle \to \langle \nu \mid A \rangle$.

While this flexibility allows MTT to accommodate many interesting calculi, it becomes proportionally more challenging to prove metatheoretic results about MTT. In particular, the rich substitution structure inherited from the mode theory can introduce subtle equations between terms. The proof that the crisp induction principles can be reconstructed in MTT [GKNB21, Theorem 10.4], for instance, exemplifies this and hinges on many such calculations. In fact, the metatheoretic results established by Gratzer et al. [GKNB20a] (soundness and canonicity) are results on closed terms in MTT, allowing their proofs to avoid the majority of the substitution apparatus.

Crucially, it remained open whether MTT admitted a normalization algorithm and, consequently, whether type checking was decidable. Even in the presence of a normalization algorithm MTT cannot admit an unconditional type checking algorithm: it is not only necessary to have a decision procedure for terms in the language, but also for modalities and 2-cells as both appear in terms for MTT.

In this paper we show the best possible result holds: MTT admits an unconditional normalization algorithm and conversion of normal forms is decidable if conversion is decidable in the mode theory.[1] As corollaries, we show that type constructors in MTT are always injective and that type checking is decidable when the mode theory is decidable.[2]

1.2. **Normalization-by-evaluation.** A normalization algorithm must begin by defining *normal forms*. Their precise formulation depends on the situation but they always satisfy two crucial properties. First, the equality of normal forms $u = v$ is clearly decidable—often no more than structural equality—and there is a function $\mathbf{dec}(u)$ decoding a normal form to a term of the same type.

Relative to a notion of normal form, a normalization algorithm sends a term $\Gamma \vdash M : A$ to a normal form $\mathbf{nf}_\Gamma(M, A)$ such that $(\mathbf{nf}_\Gamma(-, A), \mathbf{dec}(-))$ lifts to an isomorphism between equivalence classes of terms of A and normal forms [Abe13]. Typically one breaks the condition that $(\mathbf{nf}_\Gamma(-, A), \mathbf{dec}(-))$ forms an isomorphism into three conditions:

(1) *Completeness*: if $\Gamma \vdash M = N : A$ then $\mathbf{nf}_\Gamma(M, A) = \mathbf{nf}_\Gamma(N, A)$.
(2) *Soundness*: $\Gamma \vdash \mathbf{dec}(\mathbf{nf}_\Gamma(M, A)) = M : A$.
(3) *Idempotence*: $u = \mathbf{nf}_\Gamma(\mathbf{dec}(u), A)$.

Remark 1.1. We warn the reader that this terminology is not entirely standard. Various sources use the opposite conventions of soundness and completeness [AK16, AK17]. Such sources often refer to the final condition as *stability*.

Proving normalization is an involved affair. Traditionally, one begins by fixing a strongly normalizing confluent rewriting system presenting the equational theory of the type theory. The normal forms are then exactly the terms of the theory which cannot be further reduced. This approach does not scale, however, to type theories with *type-directed* equations such as the unicity principles of dependent sums and the unit type. These equations defy attempts to present them in a rewriting system and require type-directed algorithms.

The preeminent type-directed technique for normalization is *normalization-by-evaluation (NbE)* [Abe13]. Proving that an NbE algorithm works, however, is an extremely intricate affair involving a variety of complex constructions. After the algorithm is defined, the

---

[1]The converse is almost, but not quite, true. Decidability of conversion for normal forms implies that the 1- and 2-cells of the mode theory have decidable equality, as these appear in normal forms.

[2]This requirement is potentially nontrivial e.g., the word problem for groups is known to be undecidable and is subsumed by the problem for 2-categories.

proof of correctness typically proceeds by establishing properties (1)-(3) in order. Each property, moreover, requires a separate argument. Completeness is established through a PER model, soundness through a cross-language logical relation, and idempotence through a final inductive argument. The first two properties in particular are time-consuming to verify; recent work by Gratzer et al. [GSB19a] extended NbE to a type theory with an idempotent comonad but even in this minimal case the correctness proof occupied a 90 page technical report [GSB19b].

These difficulties are not unique to modal type theories, and a long line of research focuses on taming the complexity of NbE through *gluing* [AHS95, Str98, Fio02, AK16, Coq19, Ste21]. This line of work recasts normalization algorithms as the construction of models of type theory in categories defined by Artin gluing.

1.3. **Normalization-by-gluing.** Stepping back from type theory and normalization, fix a functor $F : \mathcal{C} \longrightarrow \mathcal{D}$ between a pair of categories. The *gluing* of F (written $\mathbf{Gl}(F)$) is a category whose objects triples $\big(C : \mathcal{C}, D : \mathcal{D}, f : D \longrightarrow F(C)\big)$. Morphisms in this category are given by pairs of morphisms (x_0, x_1) fitting into a commuting square, e.g.:

$$
\begin{array}{ccc}
D_0 & \xrightarrow{\ \ \ \ \ \ \ \ x_1 \ \ \ \ \ \ \ \ } & D_1 \\
{\scriptstyle f_0}\downarrow & & \downarrow{\scriptstyle f_1} \\
F(C_0) & \xrightarrow[\ \ \ \ \ F(x_0) \ \ \ \ \ ]{} & F(C_1)
\end{array}
$$

We note that there are evident projection functors $\pi_0 : \mathbf{Gl}(F) \longrightarrow \mathcal{C}$ and $\pi_1 : \mathbf{Gl}(F) \longrightarrow \mathcal{D}$.

We will view $\mathbf{Gl}(F)$ as a category of proof-relevant predicates on $\mathcal{C}$. To illustrate this, consider $\mathcal{E} = \mathbf{Gl}(\Gamma)$ where $\Gamma = [\mathbf{1}, -] : \mathcal{C} \longrightarrow \mathbf{Set}$ is the global sections map on a cartesian closed category $\mathcal{C}$ sending each object to the set of its global points. Objects in $\mathcal{E}$ then correspond to an object $C : \mathcal{C}$ equipped with a map of sets $\pi : X \longrightarrow [\mathbf{1}, C]$. Shifting perspective, we can view π as a (proof-relevant) predicate on the global points of C by setting $\Phi(c) = \pi^{-1}(c)$.

Remarkably, $\mathcal{E}$ inherits much of the structure of $\mathcal{C}$ so that $\mathcal{E}$ is also a Cartesian closed category and π_0 preserves finite products and exponentials. This is a recurrent pattern with Artin gluing; if $F : \mathcal{C} \longrightarrow \mathcal{D}$ is a nice functor between categories closed under (co)limits, exponentials, etc., then $\mathbf{Gl}(F)$ will be closed under the same operations in such a way that π_0 preserves them. In fact, unfolding the construction of e.g. binary products and exponentials in $\mathcal{E}$ yields the definition familiar from logical relations.

Example 1.2. Viewing objects of $\mathcal{E}$ as proof-relevant predicates as described above, the exponential $(C, \Phi)^{(D, \Psi)}$ is given by the following pair (C^D, Ξ) where Ξ is defined as follows (writing ϵ for the evaluation map associated with C^D):

$$
\Xi(f) = \prod\nolimits_{d \in [\mathbf{1}, D]} \Psi(d) \to \Phi(\epsilon\langle f, d\rangle)
$$

Informally, therefore, we view $\mathbf{Gl}(F : \mathcal{C} \longrightarrow \mathcal{D})$ as the category of $\mathcal{D}$-valued predicates on $\mathcal{C}$ and the construction of exponentials, products, etc. within $\mathbf{Gl}(F)$ corresponds to defining a logical relation on $\mathcal{C}$. See Mitchell and Scedrov [MS93] for an exposition on this perspective.

Carrying out a normalization-by-gluing proof, therefore, turns the classical approach on its head. Originally one defined the normalization algorithm then showed it to be sound, complete, and idempotent. When carrying out the proof by gluing, the algorithm is not defined up front. Instead, one carefully constructs a gluing category $\mathbf{Gl}(F)$ built on a functor out of the category of contexts of the initial model $\mathcal{I}$. Concretely, this is the category of syntactic contexts and simultaneous substitutions between them up to definitional equality. The heart of the argument then breaks down into three steps:

(1) We show that $\mathbf{Gl}(F)$ supports a particular model of type theory $\mathcal{G}$.
(2) We define a *reify* operation which sends terms from $\mathcal{G}$ to normal forms.
(3) We show that the projection π_0 induces a morphism of models $\mathcal{G} \longrightarrow \mathcal{I}$ and that for a given term x in $\mathcal{G}$ reifying x yields a normal form for $\pi_0(x)$.

In particular, types in $\mathcal{G}$ will be chosen such that they consist of a type from the initial model along with a proof-relevant predicate carving out those terms which have (suitably hereditary) normal forms. A term in this model is then a term from the syntactic model together with a witness for the proof-relevant predicate associated with the type.

The first step and the universal property of the initial model produces a morphism of models $i : \mathcal{I} \longrightarrow \mathcal{G}$ and the second step ensures that $\pi_0 \circ i = \mathsf{id}$. Remarkably, this already defines a sound and complete normalization algorithm. The algorithm simply takes a syntactic term $M : A$, regards it as an element of the initial model, and then reifies $i(M)$ to obtain the normal form. Moreover, because $\pi_0 \circ i = \mathsf{id}$ we conclude that this yields a normal form for the supplied M.

To a coarse approximation, the construction of $\mathcal{G}$ and reification specifies the normalization algorithm and proves its soundness in a single step. The attentive reader will notice, however, that the completeness requirement from Section 1.2 seems to be absent from this new story. In fact, in this approach completeness is automatic and no proof is required. Indeed, terms and types within the initial model are realized by equivalences classes of syntactic terms and types taken up to definitional equality. Accordingly, the morphism i—and therefore the normalization algorithm—cannot distinguish between definitional equal terms.

One might suspect that working with equivalence classes of terms when defining $\mathcal{G}$ simply causes the burden to shift so that—while there is no need to prove completeness separately—the work of such a proof is spread throughout the construction of $\mathcal{G}$. In fact the opposite is the case: working with terms up to definitional equality substantially simplifies the construction of $\mathcal{G}$. Connectives in type theory only have universal properties up to definitional equality. Only when working with equivalences classes therefore, can we use these universal properties and benefit from existing results. For instance, we shall see that our construction of dependent products in our gluing model is essentially mechanical.

The gluing approach yields other unexpected advantages. Recall that $\mathbf{Gl}(F)$ intuitively consists of *proof-relevant* predicates. This proof relevance is crucial to an elegant treatement of universes in the model [Coq19]. We are able to define the predicate associated with an element of a universe to consist not only of an appropriate normal form but to also contain the data of the type it encodes within the model. In proof-irrelevant settings, universes were a frequent source of difficulty which necessitated laborious techniques to encode [All87].

1.4. **Synthetic Tait computability.** Using gluing to prove normalization is certainly an improvement over 'free-hand' proofs of normalization-by-evaluation, but the picture is not as

rosy as it may first appear. Models of type theory are subject to a variety of strict equations (see Item 3 on page 4) which often force external constructions, where naturality obligations can be prohibitive. Worse, the passage between mathematics internal to the gluing category and external constructions is difficult and the boundary frequently raises mismatches.

We follow Sterling and Harper [SH21] and adopt a synthetic approach to gluing. We begin with two crucial observations. First, while models of type theory are strangely behaved objects, one can often embed a model into a presheaf topos and thereby work in an extremely rich setting. Second, when gluing together presheaf topoi along a nice functor $\mathbf{Gl}(F : \mathbf{PSh}(\mathcal{C}) \longrightarrow \mathbf{PSh}(\mathcal{D}))$, the result is another presheaf topos and the internal language of this topos contains lex idempotent monads ($\bigcirc, \bullet$) allowing one to recover both $\mathbf{PSh}(\mathcal{C})$ and $\mathbf{PSh}(\mathcal{D})$.

Sterling and collaborators have then shown that it is possible to work exclusively within the internal language of $\mathbf{Gl}(F)$ to construct the normalization model and have termed this approach *synthetic Tait computability (STC)*. Experience has shown that working internally simplifies constructions involved in the gluing model, making it practical to prove metatheorems for even extremely complex type theories like cubical type theory [SH21, SA21, Ste21, GB22, SH22].

Proofs using STC construct the model within $\mathbf{Gl}(F)$ by defining a sequence of constants within the internal language. Accordingly, the heart of the normalization proof is realized by a series of programming exercises in extensional type theory. This alone does not remove the strict equations that cause trouble with typical gluing proofs but it does provide a systematic approach to handling them. Concretely, within an STC proof, all the required strict equations have a particular form: for some type operator in the object theory, we are given an element $\mathsf{op} : \bigcirc \mathsf{Ty}$ corresponding to the operator in the syntactic model, and we must extend this to an element of Ty. Within the internal language, the two components of this problem (the element of Ty and the proof that it extends op) can be represented by an element of the following dependent sum:[3]

$$\textstyle\sum_{A : \mathsf{Ty}} x \leftarrow \mathsf{op}; \bigcirc(A = x)$$

The second component in particular represents the aforementioned strict equation. In practice, it is easy to obtain an element of Ty which extends op up to isomorphism i.e. an element of the following type:

$$\textstyle\sum_{A : \mathsf{Ty}} x \leftarrow \mathsf{op}; \bigcirc(\mathsf{Tm}(A) \cong \mathsf{Tm}(x))$$

Remarkably, this proves to be enough. The internal language of $\mathbf{Gl}(F)$ supports a strictification axiom [OP18] which provides a section to the canonical projection from the first type to the second. We are therefore able to construct various connectives which agree only up to isomorphism with their syntactic counterparts and correct them to construct the model. For instance, a dependent product is determined by a universal property and it is possible to construct a type in $\mathbf{Gl}(F)$ with this property by virtue of general categorical theorems. However, the result will only satisfy the required equation up to isomorphism. The strictification axiom allows STC proofs to benefit from the general categorical result without resorting to unfolding the construction supplied by the abstract argument.

---

[3]Here we have used standard syntactic sugar to represent the monadic operations of $\bigcirc$.

1.4.1. *Synthetic Tait computability for* MTT. Unlike Martin-Löf type theory or cubical type theory, a model of MTT is not a single category equipped with additional structure. Rather, a model is a network of categories, each supporting their own individual model of type theory which are then connected by various adjoints and natural transformations. The internal language of any of these categories is insufficient to construct the gluing model, so it is necessary to generalize from working in the extensional type theory of a topos to working in all topoi simultaneously using extensional MTT. Each topos then comes equipped with the structure of STC: a pair of lex monads and a strictification axiom. We prove that this mode-local structure is respected by the MTT modalities between topoi and call the resulting language *multimodal synthetic Tait computability*. The smooth interaction between MTT modalities and the lex monads $\bigcirc$ and $\bullet$ ensures that the key techniques of STC proofs can be generalized to multimodal STC.

With this machinery, we are able to give a concise and conceptual construction of the gluing model and extract the first normalization algorithm for multimodal type theory. In practice, this internal proof is necessary; removing the simplifying assumption on substitutions used in the canonicity proof given by Gratzer et al. [GKNB21] is already nearly intractable.

1.5. **Contributions.** We contribute a normalization algorithm for MTT equipped with the full suite of connectives: dependent sums, products, booleans, intensional identity types, a universe, and modal types. In addition to the usual corollaries of normalization (decidability of type checking, injectivity of type constructors, etc.), this sharpens the canonicity result of Gratzer et al. [GKNB20a]. This algorithm applies to any choice of mode theory and therefore simultaneously establishes normalization results for many specialized modal calculi.

In order to prove this result, we advance modern gluing techniques to apply to modal type theories and demonstrate that extensional MTT itself is a suitable metalanguage for carrying out the proof of normalization-by-gluing. We further argue that these techniques scale by extending the proof to a version of MTT supplemented with crisp induction principles and deduce that e.g., normalization continues to hold.

Section 2 gives a brief tutorial on MTT and introduces normal forms for this type theory. In Section 3, we discuss the models of MTT and relax the definition of a model of MTT to obtain *MTT cosmoi*. We prove that the syntactic cosmos enjoys a privileged position among MTT cosmoi (Theorem 3.9). Section 4 introduces *multimodal synthetic Tait computability* and shows that gluing together a network of topoi results in a model of extensional MTT equipped with STC structure in each mode (Theorem 4.17). Finally, in Section 5 we construct the normalization cosmos (Theorem 5.12) and extract the normalization function in Section 6 (Theorem 6.4). Section 7 discusses an extension of this proof to support crisp induction.

2. A primer on MTT

We collect the key ideas of MTT [GKNB21]. First, as mentioned in Section 1, MTT is parametrized by a mode theory: a strict 2-category $\mathcal{M}$ whose objects are modes, morphisms are modalities, and 2-cells are natural transformations between modalities. Henceforth, we will work with MTT over a fixed mode theory $\mathcal{M}$.

MTT plays two distinct roles in this paper. First, it is the object theory under consideration and the subject of our normalization theorem. However, as the proof of normalization uses MTT as an internal language to construct the normalization model MTT is also used as a metalanguage. These two different uses invite two very distinct perspectives on the

type theory. In order to crystallize MTT precisely enough for the normalization result, we will view MTT as a particular generalized algebraic theory (GAT). Accordingly, binding is handled by De Bruijn indices and the theory uses explicit substitutions [ML92]. On the other hand, we will not use De Brujin indices and explicit substitutions when working with MTT as a metalanguage. In these instances, we will treat MTT as a normal type theory and avail ourselves of conveniences similar to what a proof assistant like Agda might provide.

As a compromise, we introduce MTT in Sections 2.1 and 2.2 as a formal theory but go through several important constructions in Section 2.3 using the informal surface-language employed by much of Section 5. For a comprehensive account of both perspectives, we refer the reader to Gratzer et al. [GKNB21].

2.1. **Mode-local connectives in MTT.** Each mode in MTT constitutes its own separate type theory. In fact, each mode m is equipped with its own copy the of judgments of type theory e.g., $\Gamma \mathrel{\mathsf{cx}} @ m$, $\Gamma \vdash A @ m$, $\Gamma \vdash M : A @ m$. Much of the theory of MTT is *mode-local* and only mentions a single copy of these judgments at a time. For these connectives the rules are precisely the standard rules from MLTT, replicated for each mode. The connectives of type theory—dependent sums, intensional identity types, booleans—are all incorporated in this fashion. Each mode also contains a *weak* universe à la Tarski. Explicitly, this means that there are separate codes and an $\mathsf{El}(-)$ operation decoding a code to a type, but the decoding operation only commutes with connectives up to isomorphism. While the restriction to weak universes is not fundamental, it simplifies the proof and recent implementations have shown them to be practical [Red20].

2.2. **Modalities in MTT.** The novelty of MTT comes from those connectives which mix two modes: the modalities. MTT draws inspiration from Fitch-style type theories [Clo18, BCM^{+}20] and defines each modality together with an adjoint action on contexts. Accordingly, each $\mu : n \longrightarrow m$ defines a context former sending contexts in mode m to contexts in mode n and this is then used to define modal types $\langle \mu \mid A \rangle$:

$$\frac{\Gamma \mathrel{\mathsf{cx}} @ m}{\Gamma.\{\mu\} \mathrel{\mathsf{cx}} @ n} \qquad \frac{\Gamma.\{\mu\} \vdash A @ n}{\Gamma \vdash \langle \mu \mid A \rangle @ m} \qquad \frac{\Gamma.\{\mu\} \vdash M : A @ n}{\Gamma \vdash \mathsf{mod}_\mu(M) : \langle \mu \mid A \rangle @ m}$$

These context operations assemble into a 2-functor $m \mapsto \mathsf{Cx}_m$ from $\mathcal{M}^{\mathsf{coop}}$ to the category of categories, selecting the various categories of contexts.[4] Concretely, a substitution $\Delta \vdash \gamma : \Gamma @ m$ lifts to a substitution $\Delta.\{\mu\} \vdash \gamma.\{\mu\} : \Gamma.\{\mu\} @ n$ and each 2-cell $\alpha : \nu \longrightarrow \mu$ induces a substitution $\Gamma.\{\mu\} \vdash \{\alpha\} : \Gamma.\{\nu\} @ n$. These operations satisfy several equations to organize them into a 2-functor e.g., $\Gamma.\{\mu\} \vdash \mathsf{id}.\{\mu\} = \mathsf{id} : \Gamma.\{\mu\} @ n$ and $\Gamma.\{\mu\}.\{\xi\} = \Gamma.\{\mu \circ \xi\} \mathrel{\mathsf{cx}} @ o$. We record these rules in Figure 1.

Two basic questions remain: what is the elimination principle for $\langle \mu \mid A \rangle$ and which terms can be constructed in the context $\Gamma.\{\mu\}$? Both of these problems are addressed through the same idea, the final component of MTT. We generalize the context extension $\Gamma.A$ from MLTT to annotate each variable with a modality:

$$\frac{\Gamma \mathrel{\mathsf{cx}} @ m \qquad \Gamma.\{\mu\} \vdash A @ n}{\Gamma.(\mu \mid A) \mathrel{\mathsf{cx}} @ m}$$

---

[4]Given a 2-category $\mathcal{C}$, recall that $\mathcal{C}^{\mathsf{coop}}$ is a 2-category with the same objects as $\mathcal{C}$ but with 1- and 2-cells reversed.

$$\frac{\mu : n \longrightarrow m \qquad \Gamma \ \mathsf{cx} @ m}{\Gamma.\{\mu\} \ \mathsf{cx} @ n} \qquad\qquad \frac{\mu : n \longrightarrow m \qquad \Gamma \vdash \delta : \Delta @ m}{\Gamma.\{\mu\} \vdash \delta.\{\mu\} : \Delta.\{\mu\} @ n}$$

$$\frac{\mu : n \longrightarrow m \qquad \Gamma \vdash \delta_0 : \Delta_0 @ m \qquad \Delta_0 \vdash \delta_1 : \Delta_1 @ m}{\Gamma.\{\mu\} \vdash (\delta_1 \circ \delta_0).\{\mu\} = \delta_1.\{\mu\} \circ \delta_0.\{\mu\} : \Delta_1.\{\mu\} @ n} \qquad \frac{\mu : n \longrightarrow m \qquad \Gamma \ \mathsf{cx} @ m}{\Gamma.\{\mu\} \vdash \mathsf{id}.\{\mu\} = \mathsf{id} : \Gamma.\{\mu\} @ n}$$

$$\frac{\nu : o \longrightarrow n \qquad \mu : n \longrightarrow m \qquad \Gamma \ \mathsf{cx} @ m}{\Gamma.\{\mu \circ \nu\} = \Gamma.\{\mu\}.\{\nu\} \ \mathsf{cx} @ o}$$

$$\frac{\nu : o \longrightarrow n \qquad \mu : n \longrightarrow m \qquad \Gamma \vdash \delta : \Delta @ m}{\Gamma.\{\mu \circ \nu\} \vdash \delta.\{\mu\}.\{\nu\} = \delta.\{\mu \circ \nu\} : \Delta.\{\mu \circ \nu\} @ o}$$

$$\frac{\mu, \nu : n \longrightarrow m \qquad \alpha : \nu \longrightarrow \mu \qquad \Gamma \vdash \delta : \Delta @ m}{\Gamma.\{\mu\} \vdash \{\alpha\}_\Gamma : \Gamma.\{\nu\} @ n} \qquad \frac{\mu : n \longrightarrow m \qquad \Gamma \ \mathsf{cx} @ m}{\Gamma.\{\mu\} \vdash \mathsf{id} = \{\mathsf{id}\}_\Gamma : \Gamma.\{\mu\} @ n}$$

$$\frac{\Gamma, \Delta \ \mathsf{cx} @ m \qquad \mu, \nu : n \longrightarrow m \qquad \Gamma \vdash \delta : \Delta @ m \qquad \alpha : \nu \longrightarrow \mu}{\Gamma.\{\mu\} \vdash \{\alpha\}_\Gamma \circ (\delta.\{\mu\}) = (\delta.\{\nu\}) \circ \{\alpha\}_\Delta : \Delta.\{\nu\} @ n}$$

$$\frac{\Gamma \ \mathsf{cx} @ m \qquad \mu_0, \mu_1, \mu_2 : n \longrightarrow m \qquad \alpha_0 : \mu_0 \longrightarrow \mu_1 \qquad \alpha_1 : \mu_1 \longrightarrow \mu_2}{\Gamma.\{\mu_2\} \vdash \{\alpha_1 \circ \alpha_0\}_\Gamma = \{\alpha_0\}_\Gamma \circ \{\alpha_1\}_\Gamma : \Gamma.\{\mu_0\} @ n}$$

$$\frac{\Gamma \ \mathsf{cx} @ m \qquad \nu_0, \nu_1 : o \longrightarrow n \qquad \mu_0, \mu_1 : n \longrightarrow m \qquad \beta : \nu_0 \longrightarrow \nu_1 \qquad \alpha : \mu_0 \longrightarrow \mu_1}{\Gamma.\{\mu_1 \circ \nu_1\} \vdash \{\alpha \bullet \beta\}_\Gamma = \{\alpha\}_\Gamma.\{\nu_0\} \circ \{\beta\}_{\Gamma.\{\mu_1\}} : \Gamma.\{\mu_0 \circ \nu_0\} @ o}$$

Figure 1: Key rules for contexts and substitutions in MTT

Intuitively, $\Gamma.(\mu \mid A)$ plays the same role as $\Gamma.\langle \mu \mid A\rangle$ and comes equipped with a similar universal property: a substitution $\Delta \vdash \gamma : \Gamma.(\mu \mid A) @ m$ is precisely determined by a substitution $\Delta \vdash \gamma' : \Gamma @ m$ and a term $\Delta.\{\mu\} \vdash M : A[\gamma'.\{\mu\}] @ n$. The ordinary context extension $\Gamma.A$ is recovered by taking $\mu = \mathsf{id}$; the equation $\Gamma.\{\mathsf{id}\} = \Gamma$ ensures that the universal properties of $\Gamma.A$ and $\Gamma.(\mathsf{id} \mid A)$ match.

Despite the similarities between $\Gamma.(\mu \mid A)$ and $\Gamma.(\mathsf{id} \mid \langle \mu \mid A\rangle)$, they occupy different positions in the theory. The variable rule of MTT is adjusted to take into account modal annotations and require that the modalities in the context must cancel a variable's annotation:

$$\frac{\Gamma \ \mathsf{cx} @ m \qquad \Gamma.\{\mu\} \vdash A @ n}{\Gamma.(\mu \mid A).\{\mu\} \vdash \mathbf{v}_0 : A[\uparrow.\{\mu\}] @ n}$$

As in Martin-Löf type theory, it is necessary to apply a weakening substitution $\uparrow$ to A when describing the type of $\mathbf{v}_0$. The normal variable rule arises again as a special case after setting $\mu = \mathsf{id}$. Note that attempting to state such a variable rule for $\Gamma.(\mathsf{id} \mid \langle \mu \mid A\rangle)$ would quickly introduce issues around substitution within the theory, so these two contexts behave quite differently in practice.

Remark 2.1. From the view of Fitch-style type theories where $-.\{\mu\}$ is left adjoint to the modal type, this rule plays the role of the counit; it allows us to pass from $L(R(A))$ to A.

The addition of modal annotations creates a redundancy in our system: we may hypothesize of $\langle \mu \mid A \rangle$ with annotation ν or directly hypothesize over A with annotation $\nu \circ \mu$. There is a substitution navigating in one direction, but not the other:

$$\Gamma.(\nu \circ \mu \mid A) \vdash \uparrow.\mathsf{mod}_\mu(\mathbf{v}_0) : \Gamma.(\nu \mid \langle \mu \mid A \rangle) @ o$$

This mismatch is addressed through elimination for $\langle \mu \mid - \rangle$. Informally, this rule ensures that these two contexts are isomorphic 'from the perspective of a type':[5]

$$\frac{\begin{array}{cc} \nu : m \longrightarrow o \qquad \mu : n \longrightarrow m \\ \Gamma \; \mathsf{cx} @ o \qquad \Gamma.\{\nu\}.\{\mu\} \vdash A @ n \qquad \Gamma.(\nu \mid \langle \mu \mid A \rangle) \vdash B @ m \\ \Gamma.\{\nu\} \vdash M_0 : \langle \mu \mid A \rangle @ m \qquad \Gamma.(\nu \circ \mu \mid A) \vdash M_1 : B[\uparrow.\mathsf{mod}_\mu(\mathbf{v}_0)] @ o \end{array}}{\Gamma \vdash \mathsf{let}_\mu \; \mathsf{mod}_\nu(\_) \leftarrow M_0 \; \mathsf{in} \; M_1 : B[\mathsf{id}.M_0] @ o}$$

$$\mathsf{let}_\mu \; \mathsf{mod}_\nu(\_) \leftarrow \mathsf{mod}_\nu(M_0) \; \mathsf{in} \; M_1 = M_1[\mathsf{id}.M_0]$$

Notice that the elimination rule for the modal type $\langle \mu \mid - \rangle$ is parameterized by an additional modality ν. We refer to μ as the *main modality* and ν as the *framing modality*.

Remark 2.2. Fitch-style type theories require $\Gamma.(\nu \circ \mu \mid A) \vdash \uparrow.\mathsf{mod}_\mu(\mathbf{v}_0) : \Gamma.(\nu \mid \langle \mu \mid A \rangle) @ o$ to be invertible. Such an inverse, however, again disrupts substitution in the presence of multiple modalities. For an extended discussion of this point and various potential solutions, see Gratzer et al. [GCK$^+$22].

In addition to modal types, dependent products in MTT are also modalized so that $A \rightarrow B$ is replaced by $(\mu \mid A) \rightarrow B$:

$$\frac{\Gamma.(\mu \mid A) \vdash M : B @ m}{\Gamma \vdash \lambda(M) : (\mu \mid A) \rightarrow B @ m} \qquad \frac{\Gamma \vdash M : (\mu \mid A) \rightarrow B @ m \qquad \Gamma.\{\mu\} \vdash N : A @ n}{\Gamma \vdash M(N) : B[\mathsf{id}.N] @ m}$$

This feature is a useful convenience; it ensures that many functions avoid the need to accept an argument of modal type only to immediately apply the elimination rule. We will see frequent examples of this pattern later as MTT is used as a metalanguage.

2.3. **Standard combinators within MTT.** As the assignment $\Gamma \mapsto \Gamma.\{\mu\}$ is pseudo-functorial, its adjoint action on types is likewise functorial *up to propositional equality*. In particular, there are equivalences $\mathsf{triv} : \langle \mathsf{id} \mid A \rangle \rightarrow A$ and $\mathsf{comp} : \langle \mu \mid \langle \nu \mid A \rangle \rangle \rightarrow \langle \mu \circ \nu \mid A \rangle$:

$$\mathsf{triv}(x) = \mathsf{let}_{\mathsf{id}} \; \mathsf{mod}_{\mathsf{id}}(y) \leftarrow x \; \mathsf{in} \; y$$

$$\mathsf{triv}^{-1}(x) = \mathsf{mod}_{\mathsf{id}}(x)$$

$$\mathsf{comp}(x) = \mathsf{let}_{\mathsf{id}} \; \mathsf{mod}_\mu(y_0) \leftarrow x \; \mathsf{in} \; \mathsf{let}_\mu \; \mathsf{mod}_\nu(y_1) \leftarrow y_0 \; \mathsf{in} \; \mathsf{mod}_{\mu \circ \nu}(y_1)$$

$$\mathsf{comp}^{-1}(x) = \mathsf{let}_{\mathsf{id}} \; \mathsf{mod}_{\mu \circ \nu}(y) \leftarrow x \; \mathsf{in} \; \mathsf{mod}_\mu(\mathsf{mod}_\nu(y))$$

Each modality $\langle \mu \mid - \rangle$ also satisfies the modal principle referred to as *axiom K* i.e., they preserve finite products. In practice, this property serves as an internalization of functoriality as it provides a canonical comparison map $\langle \mu \mid A \rightarrow B \rangle \rightarrow \langle \mu \mid A \rangle \rightarrow \langle \mu \mid B \rangle$. In fact, we can prove a dependent version of this map as in Birkedal et al. [BCM$^+$20]:

$$(\circledast) : \langle \mu \mid (x : A) \rightarrow B(x) \rangle \rightarrow (a : \langle \mu \mid A \rangle) \rightarrow \mathsf{let} \; \mathsf{mod}_\mu(a_0) \leftarrow a \; \mathsf{in} \; \langle \mu \mid B(a_0) \rangle$$

---

[5]Formally, this rule ensures that, among others, this map is anodyne in the sense of Awodey [Awo18].

$$f \circledast a = \mathsf{let}\ \mathsf{mod}_\mu(f_0) \leftarrow f\ \mathsf{in}\ \mathsf{let}\ \mathsf{mod}_\mu(a_0) \leftarrow a\ \mathsf{in}\ \mathsf{mod}_\mu(f_0(a_0))$$

In functional programming parlance, modalities are *applicative functors* though without an operation $A \to \langle \mu \mid A \rangle$ [MP08].

While it is far less useful, one can also define a version of $\circledast$ using the modalized dependent product rather than accepting elements of $\langle \mu \mid - \rangle$:

$$(\circledast') : (\mu \mid (x : A) \to B(x)) \to (\mu \mid a : A) \to \langle \mu \mid B(a) \rangle$$
$$f \circledast' a = \mathsf{mod}_\mu(f(a))$$

This is indicative of a common pattern; it is typically far more concise to use the modalized dependent product instead of accepting $\langle \mu \mid - \rangle$ in order to avoid needing to immediately eliminate arguments.

2.4. **Normal and neutral forms in MTT.** As mentioned in Section 1.2, the starting point for normalization is the definition of normal form. In MTT—as in other type theories—normal forms are presented together with a class of neutral forms. Intuitively, normal forms capture terms in β-normal and η-long form while neutrals are chains of eliminations applied to a variable.

We define normal and neutral forms as separate syntactic classes, equipped with their own family of typing judgments and decoding functions sending them to terms. Dependency complicates this definition as various typing rules require substitution in the types of premises or the conclusion. Unfortunately, it is just as hard to define substitution on normal forms as it is to define normalization in general [WCPW04]. Accordingly, a normal form (resp. neutral, normal type) is typed by the judgment $\Gamma \vdash^{\mathsf{nf}} u : A @ m$ (resp. $\Gamma \vdash^{\mathsf{ne}} e : A @ m$, $\Gamma \vdash^{\mathsf{nf}} \tau @ m$) where A is not required to be any sort of normal form. Furthermore, these judgments are defined inductive-recursively with decoding functions $|u|$ (resp. $|e|$, $|\tau|$) which send a normal form (resp. neutral, normal type) to its corresponding piece of syntax. Normal and neutral forms for mode-local connectives are unchanged from their standard presentation in type theory:

$$
\begin{array}{llll}
(\textit{Normals}) & u & ::= & \lambda(u) \mid \mathsf{up}(e) \mid \mathsf{mod}_\mu(u) \mid \dots \\
(\textit{Neutral}) & e & ::= & \mathbf{v}_k^\alpha \mid e(u) \mid \mathsf{letmod}(\mu; \nu; \tau; e; u) \mid \dots \\
(\textit{Normal types}) & \tau & ::= & (\mu \mid \tau) \to \sigma \mid \langle \mu \mid \tau \rangle \mid \mathsf{El}(u) \mid \dots
\end{array}
$$

We defer a more complete presentation of the judgments and decoding function to Figure 3, but remark that the neutral form for variables is annotated with a 2-cell and index, decoding to $\mathbf{v}_0$ together with a combination of weakening and 2-cell substitutions $\uparrow$ and $\{\alpha\}$. Note that we require that $\mathsf{El}(-)$ commute with type formers only up to isomorphism (weak Tarski universes) we must include neutral and normal forms for e.g., $\mathsf{El}(\widehat{\langle \mu \mid A \rangle})$ as well as other type connectives. We include only those for $\widehat{\langle \mu \mid - \rangle}$ as they are representative of the general pattern.

To ensure that normal forms are η-long, neutrals can only be 'injected' into normals by $\mathsf{up}(-)$ for types without an η law e.g., at modal types but not at dependent products. Finally, we emphasize that normal forms are freely generated so their equality is decidable if equality of modalities and 2-cells is decidable. This is more subtle than it may appear at first blush, and we return to this point in Section 6.2.

$$\overline{\Gamma \vdash\, ! : \mathbf{1} \,@\, m} \qquad \overline{|!| = !} \qquad \overline{\Gamma.(\mu \mid A) \vdash\, \uparrow\, : \Gamma \,@\, m} \qquad \overline{|\uparrow| = \uparrow} \qquad \overline{\Gamma \vdash \mathsf{id} : \Gamma \,@\, m} \qquad \overline{|\mathsf{id}| = \mathsf{id}}$$

$$\frac{\Gamma_0 \vdash r : \Gamma_1 \,@\, m \qquad \Gamma_1 \vdash s : \Gamma_2 \,@\, m}{\Gamma_0 \vdash s \circ r : \Gamma_2 \,@\, m} \qquad \overline{|s \circ r| = |s| \circ |r|} \qquad \frac{\Gamma \vdash r : \Delta \,@\, m}{\Gamma.\{\mu\} \vdash r.\{\mu\} : \Delta.\{\mu\} \,@\, n} \qquad \overline{|r.\{\mu\}| = |r|.\{\mu\}}$$

$$\frac{\mu, \nu : n \longrightarrow m \qquad \alpha : \nu \longrightarrow \mu}{\Gamma.\{\mu\} \vdash \{\alpha\}_\Gamma : \Gamma.\{\nu\} \,@\, n \qquad |\{\alpha\}_\Gamma| = \{\alpha\}_\Gamma}$$

$$\frac{\Gamma \vdash r : \Delta \,@\, m \qquad \Gamma.\{\mu\} \vdash^{\mathsf{ne}} \mathbf{v}_k^\alpha : A[|r|.\{\mu\}] \,@\, n}{\Gamma \vdash r.\mathbf{v}_k^\alpha : \Delta.(\mu \mid A) \,@\, m \qquad |r.\mathbf{v}_k^\alpha| = |r|.|\mathbf{v}_k^\alpha|}$$

Figure 2: Complete definition of renamings

Renamings. While normal and neutral forms are not stable under substitution, they are stable under the restricted class of *renamings*. The formal definition of renamings is presented in Figure 2. Intuitively, they are the smallest class of substitutions closed under weakening, composition, identity, modal substitutions $(-.\{\mu\}, \{\alpha\})$, and extension by variables $\mathbf{v}_k^\alpha$.

Renamings are easily seen to act on normal forms, neutral forms, and normal types. Unlike normals and neutrals, however, renamings are taken up to a definitional equality which ensures that e.g., composition is associative and that modal substitutions organize into a 2-functor. This poses no issue as the action of renamings on normals and neutrals send definitionally equal renamings to identical normals and neutrals, ensuring that the action lifts to equivalences classes.

A nontrivial definitional equality on renamings is essential, however, as it ensures that the class of contexts of mode m and renamings between them organizes into a category Ren_m and that the assignments $m \mapsto \mathsf{Ren}_m$, $\mu \mapsto -.\{\mu\}$, and $\alpha \mapsto \{\alpha\}$ define a 2-functor $\mathcal{M}^{\mathsf{coop}} \longrightarrow \mathbf{Cat}$.

Lemma 2.3. *The decoding of renamings to substitutions gives a 2-natural transformation* $\mathbf{i}[-] : \mathsf{Ren}_- \longrightarrow \mathsf{Cx}_-$.

3. Models and cosmoi

Gratzer et al. [GKNB21] introduced MTT as a generalized algebraic theory so that MTT is automatically equipped with a category of models. A standard result of GATs ensures that the syntax of MTT organizes into an initial model which opens the possibility of semantic methods for proving results about syntax. Gratzer et al. [GKNB21] then repackages the definition of models in the language of natural models [Awo18].

3.1. **Natural models of MTT.** We begin by recalling the presentation of a model of MTT given by Gratzer et al. [GKNB21]. Recall that a natural model of type theory [Awo18] is a pair of a category $\mathcal{C}$—representing a category of contexts—together with a representable natural transformation $\tau : \mathcal{T}^\bullet \longrightarrow \mathcal{T}$:

Definition 3.1. A natural transformation $f : X \longrightarrow Y : \mathbf{PSh}(\mathcal{C})$ is *representable* when each fiber of f over a representable point of Y is itself representable i.e., $\mathbf{y}(C) \times_Y X$ is representable for each $\mathbf{y}(C) \longrightarrow Y$.

$$\frac{}{\Gamma \vdash^{\mathsf{nf}} \mathsf{bool} @ m} \qquad \frac{}{\Gamma \vdash^{\mathsf{nf}} \mathsf{U} @ m} \qquad \frac{\Gamma \vdash^{\mathsf{nf}} \tau @ m \qquad \Gamma.(\mu \mid |\tau|) \vdash^{\mathsf{nf}} \sigma @ m}{\Gamma \vdash^{\mathsf{nf}} (\mu \mid \tau) \to \sigma @ m}$$

$$\frac{\Gamma \vdash^{\mathsf{nf}} \tau @ m \qquad \Gamma.(\mathsf{id} \mid |\tau|) \vdash^{\mathsf{nf}} \sigma @ m}{\Gamma \vdash^{\mathsf{nf}} \Sigma(\tau,\sigma) @ m} \qquad \frac{\Gamma \vdash^{\mathsf{nf}} \tau @ m \qquad \Gamma \vdash^{\mathsf{nf}} u,v : |\tau| @ m}{\Gamma \vdash^{\mathsf{nf}} \mathsf{Id}_\tau(u,v) @ m}$$

$$\frac{\Gamma.\{\mu\} \vdash^{\mathsf{nf}} \tau @ n}{\Gamma \vdash^{\mathsf{nf}} \langle \mu \mid \tau \rangle @ m} \qquad \frac{\Gamma \vdash^{\mathsf{nf}} u : \mathsf{U} @ m}{\Gamma \vdash^{\mathsf{nf}} \mathsf{El}(u) @ m}$$

$$\frac{\Gamma(k) = (\mu \mid A) \qquad \mathsf{locks}(\Gamma,k) = \nu \qquad \alpha : \mu \longrightarrow \nu}{\Gamma \vdash^{\mathsf{ne}} \mathbf{v}_k^\alpha : A[\{\alpha\} \circ (\uparrow.\{\nu_{k-1}\}) \cdots \circ (\uparrow.\{\nu_0\})] @ m}$$

$$\frac{}{\Gamma \vdash^{\mathsf{nf}} \mathsf{tt} : \mathsf{bool} @ m} \qquad \frac{}{\Gamma \vdash^{\mathsf{nf}} \mathsf{ff} : \mathsf{bool} @ m} \qquad \frac{\Gamma \vdash^{\mathsf{ne}} e : \mathsf{bool} @ m}{\Gamma \vdash^{\mathsf{nf}} \mathsf{up}(e) : \mathsf{bool} @ m}$$

$$\frac{\Gamma \vdash^{\mathsf{ne}} e : \mathsf{bool} @ m \qquad \begin{array}{c} \Gamma.(\mathsf{id}_m \mid \mathsf{bool}) \vdash^{\mathsf{nf}} \tau @ m \\ \Gamma \vdash^{\mathsf{nf}} v_1 : |\tau|[\mathsf{id}.\mathsf{tt}] @ m \qquad \Gamma \vdash^{\mathsf{nf}} v_2 : |\tau|[\mathsf{id}.\mathsf{ff}] @ m \end{array}}{\Gamma \vdash^{\mathsf{ne}} \mathsf{if}(\tau; e; v_1; v_2) : |\tau|[\mathsf{id}.|e|] @ m}$$

$$\frac{\Gamma \vdash^{\mathsf{nf}} u : A @ m}{\Gamma \vdash^{\mathsf{nf}} \mathsf{refl}(u) : \mathsf{Id}_A(|u|,|u|) @ m} \qquad \frac{\Gamma \vdash M_0, M_1 : A @ m \qquad \Gamma \vdash^{\mathsf{ne}} e : \mathsf{Id}_A(M_0,M_1) @ m}{\Gamma \vdash^{\mathsf{nf}} \mathsf{up}(e) : \mathsf{Id}_A(M_0,M_1) @ m}$$

$$\frac{\begin{array}{c} \Gamma \vdash M_0, M_1 : A @ m \\ \Gamma \vdash^{\mathsf{ne}} e : \mathsf{Id}_A(M_0,M_1) @ m \qquad \Gamma.(\mathsf{id}_m \mid A).(\mathsf{id}_m \mid A).(\mathsf{id}_m \mid \mathsf{Id}_{A[\uparrow^2]}(\mathbf{v}_1,\mathbf{v}_0)) \vdash^{\mathsf{nf}} \tau @ m \\ \Gamma.(\mathsf{id} \mid A) \vdash^{\mathsf{nf}} u : |\tau|[\mathsf{id}.\mathbf{v}_0.\mathbf{v}_0.\mathsf{refl}(\mathbf{v}_0)] @ m \end{array}}{\Gamma \vdash^{\mathsf{ne}} \mathsf{J}(\tau; u; e) : |\tau|[\mathsf{id}.M_0.M_1.P] @ m}$$

$$\frac{\Gamma.(\mu \mid A) \vdash^{\mathsf{nf}} u : B @ m}{\Gamma \vdash^{\mathsf{nf}} \lambda(u) : (\mu \mid A) \to B @ m} \qquad \frac{\Gamma \vdash^{\mathsf{ne}} e : (\mu \mid A) \to B @ m \qquad \Gamma \vdash^{\mathsf{nf}} u : A @ m}{\Gamma \vdash^{\mathsf{ne}} e(u) : B[\mathsf{id}.|u|] @ m}$$

$$\frac{\Gamma.\{\mu\} \vdash^{\mathsf{nf}} u : A @ n}{\Gamma \vdash^{\mathsf{nf}} \mathsf{mod}_\mu(u) : \langle \mu \mid A \rangle @ m} \qquad \frac{\Gamma \vdash^{\mathsf{ne}} e : \langle \mu \mid A \rangle @ m}{\Gamma \vdash^{\mathsf{nf}} \mathsf{up}(e) : \langle \mu \mid A \rangle @ m}$$

$$\frac{\begin{array}{c} \Gamma.\{\mu\} \vdash^{\mathsf{ne}} u : \langle \nu \mid A \rangle @ n \\ \Gamma.(\mu \mid \langle \nu \mid A \rangle) \vdash^{\mathsf{nf}} \tau @ m \qquad \Gamma.(\mu \circ \nu \mid A) \vdash^{\mathsf{nf}} u : |\tau|[\uparrow.\mathsf{mod}_\nu(\mathbf{v}_0)] @ m \end{array}}{\Gamma \vdash^{\mathsf{ne}} \mathsf{letmod}(\mu; \nu; \tau; e; u) : |\tau|[\mathsf{id}.|u|] @ m} \qquad \frac{\Gamma \vdash^{\mathsf{ne}} e : \mathsf{U} @ m}{\Gamma \vdash^{\mathsf{nf}} \mathsf{up}(e) : \mathsf{U} @ m}$$

$$\frac{\Gamma.\{\mu\} \vdash^{\mathsf{nf}} u : \mathsf{U} @ m}{\Gamma \vdash^{\mathsf{nf}} \widehat{\langle \mu \mid u \rangle} : \mathsf{U} @ m} \qquad \frac{\Gamma \vdash^{\mathsf{ne}} e : \mathsf{U} @ m \qquad \Gamma \vdash^{\mathsf{ne}} f : \mathsf{El}(|e|) @ m}{\Gamma \vdash^{\mathsf{nf}} \mathsf{up}(f) : \mathsf{El}(|e|) @ m}$$

$$\frac{\Gamma.\{\mu\} \vdash A : \mathsf{U} @ n \qquad \Gamma \vdash^{\mathsf{ne}} e : \mathsf{El}(\widehat{\langle \mu \mid A \rangle}) @ m}{\Gamma \vdash^{\mathsf{ne}} \mathsf{dec}^\rhd(e) : \langle \mu \mid \mathsf{El}(A) \rangle @ m} \qquad \frac{\Gamma \vdash^{\mathsf{nf}} u : \langle \mu \mid \mathsf{El}(A) \rangle @ m}{\Gamma \vdash^{\mathsf{nf}} \mathsf{dec}^\lhd(u) : \mathsf{El}(\widehat{\langle \mu \mid A \rangle}) @ m}$$

Figure 3: Definition of selected normals, neutrals, and normal types

Intuitively, τ displays pairs of terms with their types over types. These two objects organize into presheaves through substitution on terms and types. With this in mind, the representability condition encodes context extension.

In order to adapt this to MTT, we can no longer consider just a category of contexts. The existence of multiple modes mandates that we consider a 2-functor of contexts $F : \mathcal{M}^{\mathsf{coop}} \longrightarrow \mathbf{Cat}$. The action of modalities $F(\mu) : F(m) \longrightarrow F(n)$ gives the semantic equivalent of $-.\{\mu\}$, while the 2-cell component $F(\alpha)$ interprets $\{\alpha\}$.

Each mode $m : \mathcal{M}$ is equipped with a morphism $\tau_m : \mathcal{T}_m^\bullet \longrightarrow \mathcal{T}_m : \mathbf{PSh}(F(m))$ representing the terms and types of mode m and each modality $\mu : n \longrightarrow m$ induces a functor which acts by precomposition $F(\mu)^*$.

Definition 3.2. A model of MTT without any type constructors is a strict 2-functor $F : \mathcal{M}^{\mathsf{coop}} \longrightarrow \mathbf{Cat}$ together with a collection of morphisms $\tau_m : \mathcal{T}_m^\bullet \longrightarrow \mathcal{T}_m : \mathbf{PSh}(F(m))$ such that $F(\mu)^*(\tau_n)$ is representable for each $\mu : n \longrightarrow m$.

Connectives are individually specified on top of this structure. For instance, the following pullback square in $\mathbf{PSh}(F(m))$ for each mode m ensures closure under dependent sums:

$$
\begin{array}{ccc}
\sum_{A:\mathcal{T}_m} \sum_{B:\tau_m[A]\to\mathcal{T}_m} \sum_{a:\tau_m[A]} \tau_m[B(a)] & \longrightarrow & \mathcal{T}_m^\bullet \\
\downarrow & & \downarrow \\
\sum_{A:\mathcal{T}_m} \prod_{-:\tau_m[A]} \mathcal{T}_m & \longrightarrow & \mathcal{T}_m
\end{array}
\tag{3.1}
$$

Diagram 3.1 takes advantage of the model of extensional MLTT in a presheaf topos [Hof97] and we have written $\tau_m[A]$ to denote the specialization of τ_m (viewed as a dependent type over $\mathcal{T}_M$) with A. We will freely take advantage of this model and use our assumption of a hierarchy of Grothendieck universes to equip it with an infinite hierarchy of cumulative universes [HS97]. We refer to a family of presheaves as *small* if it is classified by a universe.

Dependent products $(\mu \mid A) \to B$ are specified by a similar pullback square but their encoding in MTT presents a slight complication. Recall that dependent products include a modality $(\mu \mid A) \to B$. In order to account for μ, we use $F(\mu)^*$; if elements of $\mathcal{T}_m(X)$ represent types from mode m in context $X : F(m)$, elements $F(\mu)^*(\mathcal{T}_n)(X)$ represent types from mode n but in context $F(\mu)(X)$. Accordingly, the presence of dependent products is encoded by the following pullback square:

$$
\begin{array}{ccc}
\sum_{A:F(\mu)^*(\mathcal{T}_n)} \sum_{B:F(\mu)^*(\tau_n)[A]\to\mathcal{T}_m} \prod_{a:F(\mu)^*(\tau_n)[A]} \tau_m[B(a)] & \longrightarrow & \mathcal{T}_m^\bullet \\
\downarrow & & \downarrow \\
\sum_{A:F(\mu)^*(\mathcal{T}_n)} F(\mu)^*(\tau_n)[A] \to \mathcal{T}_m & \longrightarrow & \mathcal{T}_m
\end{array}
\tag{3.2}
$$

Given $\mu : n \longrightarrow m$, we can specify the formation and introduction rules of $\langle \mu \mid - \rangle$ with another commuting square:

$$
\begin{array}{ccc}
F(\mu)^*\mathcal{T}_n^\bullet & \longrightarrow & \mathcal{T}_m^\bullet \\
\downarrow & & \downarrow \\
F(\mu)^*\mathcal{T}_n & \longrightarrow & \mathcal{T}_m
\end{array}
\tag{3.3}
$$

Unlike dependent sums or products, modal types do not have a universal property—an η law—so they cannot be encoded by a single pullback. Instead we must describe the elimination principle separately. Following Gratzer et al. [GKNB21], we encode the elimination principle as an internal lifting structure.

Definition 3.3 Definition 18 [Awo18]. An internal lifting structure $s : i \pitchfork \tau$ between a pair of morphisms $i : A \longrightarrow B$ and $\tau : X \longrightarrow Y$ is a section of canonical map $X^B \longrightarrow Y^B \times_{Y^A} X^A$.

Fix a pair of modalities $\mu : n \longrightarrow m$ and $\nu : o \longrightarrow n$ and write c for the comparison map $F(\nu)^*(\mathcal{T}_o^\bullet) \longrightarrow F(\nu)^*(\mathcal{T}_o) \times_{\mathcal{T}_n} \mathcal{T}_n^\bullet$ induced by Diagram 3.3. The elimination principle for ν-modal types with a framing modality μ is encoded by a lifting structure of the following type:

$$F(\mu)^*(c) \pitchfork F(\mu \circ \nu)^*(\mathcal{T}_o) \times \tau_m : \mathbf{PSh}(F(o))/F(\mu \circ \nu)^*(\mathcal{T}_o)$$

This definition is somewhat obstruse, but we will soon be in a position to formulate a far more intuitive version of it by taking advantage of a richer version of the internal language in Section 3.3.

As models of a particular GAT, models of MTT assemble into a category. A morphism between models F and G is given by a 2-natural transformation $F \longrightarrow G$ along with natural assignments of terms and types of F to the terms and types of G. All of these operations are required to strictly preserve term, type, and context formers. We refer the reader to Gratzer et al. [GKNB21] for a precise description.

Finally, a standard result of GATs is that the *syntactic model* occupies a distinguished place in the category of models:

Theorem 3.4. *Syntax is the initial model of MTT.*

3.2. **MTT cosmoi.** As mentioned in Section 1, normalization is proven through the construction of a model of MTT together with a map from this model to syntax. Models of MTT and morphisms between them are difficult to construct, however, because of the extreme strictness of morphisms and the requirement that each τ_m be a representable natural transformation. Prior to normalization, therefore, we introduce a weakened notion of model: an MTT cosmos. An MTT cosmos is an axiomatization of a natural model of MTT, but rather than working in presheaf topoi and requiring that τ_m is a representable natural transformation a cosmos requires only that τ_m be a morphism in a locally cartesian closed category equipped with structure such as Diagrams 3.2 and 3.3.

Definition 3.5. A *cosmos* is a pseudofunctor $F : \mathcal{M} \longrightarrow \mathbf{Cat}$ such that each $F(m)$ is a locally cartesian closed category and each $F(\mu)$ has a left adjoint $F_!(\mu) \dashv F(\mu)$.

One should imagine a cosmos F as arising from some model of MTT F_0 with $F(m) = \mathbf{PSh}(F_0(m))$. The adjunction $F(\mu)_! \dashv F(\mu)$ is then recording the adjunction given by precomposition and left Kan extension $F_0(\mu)_! \dashv F_0(\mu)^*$. In particular, the left adjoint to $F(\mu)$ allows us to capture the left adjoint action of a modality on contexts $(-.\{\mu\})$ while $F(\mu)$ is more intended to record the modality itself. While this example is strictly 2-functorial, we allow a general cosmos to be pseudofunctorial. The formal connection between models and cosmoi is given by the following example:

Example 3.6. A model of MTT F assembles into a cosmos G by taking $G(m) = \mathbf{PSh}(F(m))$ and $G(\mu) = F(\mu)^*$. In particular, we write $\mathcal{S} : \mathcal{M} \longrightarrow \mathbf{Cat}$ for the cosmos induced by the initial model of MTT specified by Theorem 3.4.

The additional requirements imposed by natural models of MTT to encode various connectives can be transferred *mutatis mutandis* to a cosmos; they are all stated within the language of locally cartesian closed categories.

Definition 3.7. An cosmos F is an MTT cosmos when equipped with the following structure:

(1) In $F(m)$, there is a universe $\tau_m : \mathcal{T}_m^\bullet \longrightarrow \mathcal{T}_m$ with a choice of codes witnessing its closure under dependent sums and products, identity types, and booleans. For instance, a choice of pullback square of the following shape:

$$
\begin{array}{ccc}
\sum_{A:F(\mu)(\mathcal{T}_m)} \sum_{B:F(\mu)(\tau_n)[A]\to\mathcal{T}_m} \prod_{a:F(\mu)(\tau_n)[A]} \tau_m[B(a)] & \xrightarrow{\ \mathbf{lam}\ } & \mathcal{T}_m^\bullet \\
\downarrow & & \downarrow \\
\sum_{A:F(\mu)(\mathcal{T}_n)} F(\mu)(\tau_n)[A] \to \mathcal{T}_m & \xrightarrow[\mathbf{Prod}]{} & \mathcal{T}_m
\end{array}
$$

(2) For each μ, there exists a chosen commuting square

$$
\begin{array}{ccc}
F(\mu)(\mathcal{T}_n^\bullet) & \longrightarrow & \mathcal{T}_m^\bullet \\
\downarrow & & \downarrow \\
F(\mu)(\mathcal{T}_n) & \xrightarrow[\mathbf{Mod}]{} & \mathcal{T}_m
\end{array}
\tag{3.4}
$$

(3) For each $\mu : n \longrightarrow m$ and $\nu : o \longrightarrow n$, there is a chosen lifting structure $F(\mu)(m) \pitchfork F(\mu \circ \nu)(\mathcal{T}_o) \times \tau_m$, where $m : F(\nu)(\mathcal{T}_o^\bullet) \longrightarrow F(\nu)(\mathcal{T}_o) \times_{\mathcal{T}_n} \mathcal{T}_n^\bullet$ is the comparison map induced by Diagram 3.4.

(4) τ_m contains a subuniverse also closed under all these connectives.

Definition 3.8. A morphism between MTT cosmoi $\alpha : F \longrightarrow G$ is a 2-natural transformation α such that α_m is an LCCC functor and preserves all connectives strictly.

Furthermore, we require that α satisfies the Beck-Chevalley condition so that there is a natural isomorphism $\beta_\mu : \alpha_n \circ F(\mu)_! \cong G(\mu)_! \circ \alpha_m$ commuting with transposition. Precisely, if $a : X \longrightarrow F(\mu)(Y) : F(m)$ the transposition of $\alpha_\mu \circ \alpha_m(a)$ is $\alpha_n(\widehat{a}) \circ \beta_\mu^{-1}$.

Definition 3.8 uses a number of concepts from 2-category theory and we take a moment to recall and discuss them here. First, a 2-natural transformation α between pseudofunctors $F, G : \mathcal{M} \longrightarrow \mathbf{Cat}$ consists of a collection of functors $\alpha_m : F(m) \longrightarrow G(m)$ along with a family of natural isomorphisms α_μ witnessing the commutativity of the following diagrams up to natural isomorphism:

$$
\begin{array}{ccc}
F(n) & \xrightarrow{\ \alpha_n\ } & G(n) \\
F(\mu) \downarrow & & \downarrow G(\mu) \\
F(m) & \xrightarrow[\alpha_m]{} & G(m)
\end{array}
$$

The collection of natural isomorphisms α_μ satisfy a number of coherence conditions forcing them to behave as expected with respect to composition and identity in $\mathcal{M}$ as well as to force them to be natural with respect to 2-cells in $\mathcal{M}$. Fortunately, these higher conditions will not generally factor into what follows, so we refer the reader to Johnson and Yau [JY20] where this notion is detailed under the name *strong transformation*.

Note that $F(m)$ and $G(m)$ are both LCC and equipped with universes closed under various connectives. The next part of Definition 3.8 requires that α_μ respects this additional structure. Finally, since $F(\mu)$ and $G(\mu)$ are both right adjoints, one can ask whether there is a natural isomorphism witnessing $\alpha_m \circ F_!(\mu) = G_!(\mu) \circ \alpha_n$. The final requirement—that α_μ satisfy the Beck-Chevalley condition—essentially states that there is such a natural isomorphism and that it is canonically induced from α_μ. In particular, this ensures that transposing a morphism along $F_!(\mu) \dashv F(\mu)$ and then applying α_m produces the same result as applying α_n and transposing along $G_!(\mu) \dashv G(\mu)$.

A morphism of MTT cosmoi is both more and less restrictive than a morphism of MTT models. While a morphism of models need not induce an LCC functor between the relevant presheaf categories, a morphism of cosmoi is not required to strictly preserve context extension or the choice of terminal context. It so happens that the only map of consequence in this paper is locally cartesian closed, so the additional structure of morphisms of cosmoi poses no issue. Not requiring the strict preservation of context extension and dropping the representability requirements from MTT cosmoi, however, ensures that cosmoi are far easier to construct.

Merely defining a normalization cosmos $\mathcal{G}$ and projection $\pi : \mathcal{G} \longrightarrow \mathcal{S}$, however, is not enough to prove normalization; we also need a section to π. In the category of models, this section would exist as a consequence of initiality, but $\mathcal{S}$ is not initial in the category of MTT cosmoi.[6] Accordingly, we cannot easily obtain a section of a map into $\mathcal{S}$ and in fact sections rarely exist. Any such map, however, is essentially surjective on definable terms e.g., for any syntactic context Γ there exists some object in $X : G(m)$ along with $\alpha : \pi(X) \cong \mathbf{y}(\Gamma)$. Similar statements hold for terms, types, etc. While these choices need not assemble into a morphism of cosmoi, such piecemeal liftings suffice for the normalization algorithm in Section 6.

Theorem 3.9. *Fix an MTT cosmos G and $\pi : G \longrightarrow \mathcal{S}$.*
(1) *For Γ cx @ m, there exists $[\![\Gamma]\!] : G(m)$ and a canonical isomorphism $\alpha_\Gamma : \mathbf{y}(\Gamma) \cong \pi([\![\Gamma]\!])$.*
(2) *For every $\Gamma \vdash A @ m$, there exists $[\![A]\!] : [\![\Gamma]\!] \longrightarrow \mathcal{T}_m$ such that $\pi([\![A]\!]) \circ \alpha_\Gamma = \lfloor A \rfloor$.*
(3) *For every $\Gamma \vdash M : A @ m$, there exists $[\![M]\!] : [\![\Gamma]\!] \longrightarrow \mathcal{T}_m^\bullet$ lying over $[\![A]\!]$ such that $\pi([\![M]\!]) \circ \alpha_\Gamma = \lfloor M \rfloor$.*
Here $\lfloor - \rfloor$ is the isomorphism induced by the Yoneda lemma. Moreover, each lift $[\![-]\!]$ respects definitional equality.

Remark 3.10. While we have proven this result quite generally, we will apply it only in the special case where π is a 2-natural transformation between strict 2-functors and required isomorphisms of left adjoints are likewise identities. The reader may accordingly safely ignore these coherences when reading the proof without consequence.

Remark 3.11. Both Theorem 3.4 and 3.9 are categorical abstractions of *rule induction*. Indeed, 3.4 is used to prove 3.9—via the construction of an appropriate displayed

---

[6]2-monad theory [KPT99, GS20] yields an initial cosmos $\mathcal{I}$ but we work with $\mathcal{S}$ because—unlike $\mathcal{I}$—it is known to adequately represent syntax.

model [KKA19]—and the latter takes the place of rule induction in the proof of normalization (see Theorem 6.4).

Proof. We write $\mathbf{El}_m$, $\mathbf{Ty}_m$ and $\mathbf{Tm}_m$ instead of τ_m, $\mathcal{T}_m$, and $\mathcal{T}_m^\bullet$ in the syntactic model, reserving the latter exclusively for G. We write $[\![\mu]\!]$ for the functor sending Γ to $\Gamma.\{\mu\}$. We begin by replacing G by an equivalent strict 2-functor so that π becomes strictly 2-natural.

We construct a displayed model of MTT [KKA19] which lies over the syntactic model. Using the existing coherence result for MTT [GKNB20b], we only ensure that $\Gamma.\{\mu\}.\{\nu\}$ and $\Gamma.\{\mu \circ \nu\}$ agree up to pseudonatural isomorphism.

- A context in m is a triple $X : G(m)$, $\Gamma\ \mathsf{cx} @ m$, and $\alpha : \pi(X) \cong \mathbf{y}(\Gamma)$.
- A type in a context (X, Γ, α) is a pair of $\bar{A} : X \longrightarrow \mathcal{T}_m$ and $\Gamma \vdash A @ m$ such that $\pi(\bar{A}) = \lfloor A \rfloor \circ \alpha$.
- A term in a context (X, Γ, α) of type $(\bar{A}, A)$ is a pair $\bar{M} : X \longrightarrow \tau_m[\bar{A}]$ and $\Gamma \vdash M : A @ m$ such that $\pi(\bar{M}) = \lfloor M \rfloor \circ \alpha$.
- A substitution $(X, \Gamma, \alpha) \longrightarrow (Y, \Delta, \beta)$ is a pair $f : X \longrightarrow Y$ and $\Gamma \vdash \delta : \Delta @ m$ satisfying $\beta \circ \pi(f) = \mathbf{y}(\delta) \circ \alpha$

Once this model is constructed, the result follows from Theorem 3.4. The construction of contexts, substitutions, terms, and types is straightforward as π is a 2-natural transformation which preserves finite limits, and commutes with all connectives. We show two cases.

The action of a modality on a context. Given a triple (X, Γ, α) at mode m and a modality $\mu : n \longrightarrow m$, we define the 'locked' context to be the following:

$$(G(\mu)_!(X), \Gamma.\{\mu\}, \gamma \circ [\![\mu]\!]_! \alpha \circ \beta)$$

Here $\beta : \pi(G(\mu)_! X) \cong [\![\mu]\!]_! \pi(X)$ and $\gamma : [\![\mu]\!]_! \mathbf{y}(\Gamma) \cong \mathbf{y}(\Gamma.\{\mu\})$ are the canonical isomorphisms.

Modal types. Suppose we are given a context (X, Γ, α) and a type $(\bar{A}, A)$ in the context $(G(\mu)_!(\mu)(X), \Gamma.\{\mu\}, \gamma \circ [\![\mu]\!]^*(\alpha) \circ \beta_\mu)$. Writing $\bar{B}$ for the transpose of $\bar{A}$, we form the modal type as

$$(\mathbf{Mod}_\mu(\bar{B}), \langle \mu \mid A \rangle)$$

It remains to check that these types are coherent i.e.:

$$\pi(\mathbf{Mod}_\mu(\bar{B})) = \lfloor \langle \mu \mid A \rangle \rfloor \circ \alpha$$

By assumption, $\pi(\bar{B}) = \lfloor A \rfloor \circ \gamma \circ [\![\mu]\!]^*(\alpha) \circ \beta$. By our assumption that π satisfies Beck-Chevalley $\pi(\bar{B}) = \widehat{\lfloor A \rfloor \circ \gamma} \circ \alpha$. The result follows from the fact that π preserves $\mathbf{Mod}_\mu$. $\square$

3.3. Presheaf cosmoi.
Example 3.6 shows that each model of MTT induces an MTT cosmos. In fact, such cosmoi are particularly well-behaved as they are comprised of presheaf topoi connected by adjoint triples. These cosmoi enjoy a privileged role in our proof and we observe some of their unique behavior.

Definition 3.12. A presheaf cosmos F is a cosmos where F is a strict 2-functor, each $F(m)$ is a presheaf topos, and each right adjoint $F(\mu)$ sends small families to small families.

What distinguishes presheaf cosmoi from other cosmoi is the rich internal language they offer. Gratzer et al. [GKNB21] have proven that such a cosmos F supports a model of *extensional* MTT with the same mode theory where $\langle \mu \mid - \rangle$ is interpreted by $F(\mu)$. We will now use extensional MTT as a *multimodal metalanguage* to specify the structure of

an MTT cosmos as a sequence of constants, thereby reducing its construction to a series of programming exercises. It is this characterization of MTT-cosmoi that we will use in Section 5 to construct the normalization cosmos.

Remark 3.13. Some caution is required here, as a presheaf cosmos will frequently host more than one interpretation of MTT, with different universes of types. In particular, if we consider the collection of presheaf categories $E = \mathbf{PSh}(F(-))$ where F is a strict 2-functor coming from a model of MTT, we may interpret MTT into E either by choosing types to be arbitrary families of presheaves, or locally representable families of presheaves. This is comparable to Diagram 3.1, where type theory is used to describe a model of type theory.

Within this internal language, the universe $\tau_m : \mathcal{T}_m^\bullet \longrightarrow \mathcal{T}_m$ is encoded by a pair of types:

$$\mathsf{Ty}_m : \mathsf{U}_0 \qquad \mathsf{Tm}_m : (A : \mathsf{Ty}_m) \to \mathsf{U}_0$$

Each of the diagrams discussed in Sections 3.1 and 3.2 can then be translated into constants within this language with the use of dependent types automatically encoding commutativity. For instance, Diagram 3.4 becomes the following pair of constants:

$$\mathsf{Mod}_\mu : (\mu \mid \mathsf{Ty}_n) \to \mathsf{Ty}_m \qquad \mathsf{m}_\mu : (\mu \mid A : \mathsf{Ty}_n)(\mu \mid \mathsf{Tm}_n(A)) \to \mathsf{Tm}_m(\mathsf{Mod}_\mu(A))$$

In this language it is far easier to specify the modal elimination principle:

$$\begin{aligned}
&\mathsf{letmod}_{\mu;\nu} : \\
&\quad (\nu \circ \mu \mid A : \mathsf{Ty}_n)\,(B : (\nu \mid \mathsf{Tm}_n(\mathsf{Mod}_\mu(A))) \to \mathsf{Ty}_o) \\
&\quad \big(b : \big(\nu \circ \mu \mid x : \mathsf{Tm}_n(A)\big) \to \mathsf{Tm}_o\big(B(\mathsf{m}_\mu(A, x))\big)\big) \\
&\quad \to (\nu \mid a : \mathsf{Tm}_m(\mathsf{Mod}_\mu(A))) \to \mathsf{Tm}_o(B(a))
\end{aligned}$$

Each argument to $\mathsf{letmod}_{\mu;\nu}$ corresponds directly to a premise of the rule given in Section 2. The hypothetical judgment is encoded by the dependent products in the language and each occurrence of $-.\{-\}$ is replaced with an occurrence of the corresponding modal type within the metalanguage. The β-rule for this elimination principle is encoded by another constant inhabiting the equality type:

$$\begin{aligned}
&\mathsf{Mod/beta}_{\mu;\nu} : \\
&\quad (\nu \circ \mu \mid A : \mathsf{Ty}_n)\,(B : (\nu \mid \mathsf{Tm}_n(\mathsf{Mod}_\mu(A))) \to \mathsf{Ty}_o) \\
&\quad \big(b : \big(\nu \circ \mu \mid x : \mathsf{Tm}_n(A)\big) \to \mathsf{Tm}_o\big(B(\mathsf{m}_\mu(A, x))\big)\big) \\
&\quad \to (\nu \circ \mu \mid a : \mathsf{Tm}_m(A)) \to \mathsf{letmod}_{\mu;\nu}(A, B, b, \mathsf{m}_\mu(A, a)) = b(a)
\end{aligned}$$

The remaining connectives are detailed in Figure 4.

4. Multimodal Synthetic Tait computability

In light of Section 3, we revise the proof outlined in Section 1: instead of constructing a glued *model* of MTT, we will construct a glued MTT *cosmos*. In fact, we will construct a glued presheaf cosmos, and take advantage of the internal language discussed in Section 3.3 to upgrade it to an MTT cosmos with a projection onto $\mathcal{S}$. Prior to this, however, we must show that (1) a pair of cosmoi can be glued together and (2) that each mode of the internal language of the resulting cosmos can be extended with synthetic Tait computability primitives compatible with the already-present MTT modalities.

$\mathsf{Prod} : (\mu \mid A : \mathsf{Ty}_m)\,(B : (\mu \mid \mathsf{Tm}_m(A)) \to \mathsf{Ty}_m) \to \mathsf{Ty}_m$

$\alpha_{\mathsf{Prod}} : (\mu \mid A : \mathsf{Ty}_m)\,(B : (\mu \mid \mathsf{Tm}_m(A)) \to \mathsf{Ty}_m)$

$\quad \to \mathsf{Tm}_m(\mathsf{Prod}(A,B)) \cong [(\mu \mid a : \mathsf{Tm}_m(A)) \to \mathsf{Tm}_m(B(a))]$

$\mathsf{Sig} : (A : \mathsf{Ty}_m) \to (\mathsf{Tm}_m(A) \to \mathsf{Ty}_m) \to \mathsf{Ty}_m$

$\alpha_{\mathsf{Sig}} : (A : \mathsf{Ty}_m)(B : \mathsf{Tm}_m(A) \to \mathsf{Ty}_m)$

$\quad \to \mathsf{Tm}_m(\mathsf{Sig}(A,B)) \cong \left[\sum_{a:\mathsf{Tm}_m(A)} \mathsf{Tm}_m(B(a)) \right]$

$\mathsf{Bool} : \mathsf{Ty}_m$

$\mathsf{true}, \mathsf{false} : \mathsf{Tm}_m(\mathsf{Bool})$

$\mathsf{if} : (A : \mathsf{Tm}_m(\mathsf{Bool}) \to \mathsf{Ty}_m)$

$\quad \to \mathsf{Tm}_m(A(\mathsf{true})) \to \mathsf{Tm}_m(A(\mathsf{false})) \to (b : \mathsf{Tm}_m(\mathsf{Bool})) \to \mathsf{Tm}_m(A(b))$

$\_ : (A : \mathsf{Tm}_m(\mathsf{Bool}) \to \mathsf{Ty}_m)\,(t : \mathsf{Tm}_m(A(\mathsf{true})))\,(f : \mathsf{Tm}_m(A(\mathsf{false})))$

$\quad \to (\mathsf{if}(A,t,f,\mathsf{true}) = t) \times (\mathsf{if}(A,t,f,\mathsf{false}) = f)$

$\mathsf{Id} : (A : \mathsf{Ty}_m)(a_0, a_1 : \mathsf{Tm}_m(A)) \to \mathsf{Ty}_m$

$\mathsf{refl} : (A : \mathsf{Ty}_m)(a : \mathsf{Tm}_m(A)) \to \mathsf{Tm}_m(\mathsf{Id}(A,a,a))$

$\mathsf{J} : (A : \mathsf{Ty}_m)\,(B : (a_0, a_1 : \mathsf{Tm}_m(A))(p : \mathsf{Tm}_m(\mathsf{Id}(A,a_0,a_1))) \to \mathsf{Ty}_m)$

$\quad \to ((a : \mathsf{Tm}_m(A)) \to \mathsf{Tm}_m(B(a,a,\mathsf{refl}(a))))$

$\quad \to (a_0, a_1 : \mathsf{Tm}_m(A))(p : \mathsf{Tm}_m(\mathsf{Id}(A,a_0 a_1))) \to \mathsf{Tm}_m(B(a_0,a_1,p))$

$\_ : (A : \mathsf{Ty}_m)\,(B : (a_0, a_1 : \mathsf{Tm}_m(A))(p : \mathsf{Tm}_m(\mathsf{Id}(A,a_0,a_1))) \to \mathsf{Ty}_m)$

$\quad \to (b : (a : \mathsf{Tm}_m(A)) \to \mathsf{Tm}_m(B(a,a,\mathsf{refl}(a))))$

$\quad \to (a : \mathsf{Tm}_m(A)) \to \mathsf{J}(A,B,b,a,a,\mathsf{refl}(a)) = b(a)$

$\mathsf{Uni} : \mathsf{Ty}_m$

$\mathsf{El} : \mathsf{Tm}_m(\mathsf{Uni}) \to \mathsf{Ty}_m$

$\widehat{\mathsf{Sig}} : (A : \mathsf{Tm}_m(\mathsf{Uni})) \to (\mathsf{Tm}_m(\mathsf{El}(A)) \to \mathsf{Tm}_m(\mathsf{Uni})) \to \mathsf{Tm}_m(\mathsf{Uni})$

$\widehat{\mathsf{Prod}} : (\mu \mid A : \mathsf{Tm}_n(\mathsf{Uni})) \to ((\mu \mid \mathsf{Tm}_n(\mathsf{El}(A))) \to \mathsf{Ty}_m) \to \mathsf{Tm}_m(\mathsf{Uni})$

$\widehat{\mathsf{Bool}} : \mathsf{Tm}_m(\mathsf{Uni})$

$\widehat{\mathsf{Mod}} : (\mu \mid \mathsf{Tm}_n(\mathsf{Uni})) \to \mathsf{Tm}_m(\mathsf{Uni})$

$\mathsf{dec}_{\widehat{\mathsf{Sig}}} : (A : \mathsf{Tm}_m(\mathsf{Uni}))(B : \mathsf{Tm}_m(\mathsf{El}(A)) \to \mathsf{Tm}_m(\mathsf{Uni}))$

$\quad \to \mathsf{Tm}_m(\mathsf{El}(\widehat{\mathsf{Sig}}(A,B))) \cong \mathsf{Tm}_m(\mathsf{Sig}(\mathsf{El}(A), \mathsf{El} \circ B))$

$\mathsf{dec}_{\widehat{\mathsf{Prod}}} : (\mu \mid A : \mathsf{Tm}_n(\mathsf{Uni}))(B : (\mu \mid \mathsf{Tm}_n(\mathsf{El}(A))) \to \mathsf{Tm}_m(\mathsf{Uni}))$

$\quad \to \mathsf{Tm}_m(\mathsf{El}(\widehat{\mathsf{Prod}}(A,B))) \cong \mathsf{Tm}_m(\mathsf{Prod}(\mathsf{El}(A), \mathsf{El} \circ B))$

$\mathsf{dec}_{\widehat{\mathsf{Bool}}} : \mathsf{Tm}_m(\mathsf{El}(\widehat{\mathsf{Bool}})) \cong \mathsf{Tm}_m(\mathsf{Bool})$

$\mathsf{dec}_{\widehat{\mathsf{Mod}}} : (\mu \mid A : \mathsf{Tm}_m(\mathsf{Uni})) \to \mathsf{Tm}_m(\mathsf{El}(\widehat{\mathsf{Mod}}(A))) \cong \mathsf{Tm}_m(\mathsf{Mod}_\mu(\mathsf{El}(A)))$

Figure 4: Internal presentation of an MTT cosmos

4.1. **Synthetic Tait computability.** For this subsection, fix two presheaf topoi $\mathcal{E}$ and $\mathcal{F}$ along with a continuous functor $\rho : \mathcal{E} \longrightarrow \mathcal{F}$.

Definition 4.1. The *Artin gluing* $\mathbf{Gl}(\rho)$ is a category whose objects are triples (E, F, f) of an object from $\mathcal{E}$, an object from $\mathcal{F}$, and a morphism $F \longrightarrow \rho(E)$. Morphisms in $\mathbf{Gl}(\rho)$ are commuting squares:

$$
\begin{array}{ccc}
F_0 & \xrightarrow{\ \ \alpha\ \ } & F_1 \\
{\scriptstyle f_0}\downarrow & & \downarrow{\scriptstyle f_1} \\
\rho(E_0) & \xrightarrow[\rho(\beta)]{} & \rho(E_1)
\end{array}
$$

Projection induces functors $\pi_0 : \mathbf{Gl}(\rho) \longrightarrow \mathcal{E}$ and $\pi_1 : \mathbf{Gl}(\rho) \longrightarrow \mathcal{F}$.

Example 4.2. Intuitively $\mathbf{Gl}(\rho)$ is a category of proof-relevant $\mathcal{F}$-predicates on ρ-elements of $\mathcal{E}$. To cultivate this intuition, consider $\mathcal{F} = \mathbf{Set}$ and $\rho = [\mathbf{1}, -]$. An object of $\mathbf{Gl}([\mathbf{1}, -])$ is a triple of (S, E, f) which induces a proof-relevant predicate $\Phi(e) = f^{-1}(e)$ on the global points of E. Following Tait [Tai67], we refer to elements in the image of f as *computable elements*. Morphisms are then morphisms of $\mathcal{E}$ equipped with additional structure ensuring that computable elements are sent to computable elements.

We now reap the first reward from considering proof-*relevant* predicates: $\mathbf{Gl}(\rho)$ is extremely well-behaved.

Theorem 4.3 [AGV72, CJ95]**.** $\mathbf{Gl}(\rho)$ *is a presheaf topos and* π_0 *is a logical functor with left and right adjoints.*

As a presheaf topos, $\mathbf{Gl}(\rho)$ enjoys a model of extensional type theory with a strictly cumulative hierarchy of universes and a universe of propositions Ω. We can use this language to *synthetically* build logical relations models [SH21]. In order to effectively construct such models, however, we must supplement type theory with primitives specific to $\mathbf{Gl}(\rho)$. The most fundamental of these is a proposition:

Definition 4.4. The *syntactic proposition* $\mathbf{syn} : \Omega$ is interpreted in $\mathbf{Gl}(\rho)$ as the subterminal object $(\mathbf{1}_{\mathcal{E}}, \mathbf{0}_{\mathcal{F}}, !)$.

Recalling the correspondence between objects of $\mathbf{Gl}(\rho)$ and predicates, $\mathbf{syn}$ is the predicate on $\mathbf{1}_{\mathcal{E}}$ with no computable elements. What makes this proposition useful is its ability to wipe out the obligation to track computable elements. A morphism $f : \mathbf{syn} \times A \longrightarrow B$ must contain a morphism $\pi_0(f) : \pi_0(\mathbf{syn} \times A) \cong \pi_0(A) \longrightarrow \pi_0(B)$, but there are no computable elements of $\mathbf{syn} \times A$ so $\pi_0(f)$ entirely determines f; there is a bijection $[\mathbf{syn} \times A, B]_{\mathbf{Gl}(\rho)} \cong [\pi_0(A), \pi_0(B)]_{\mathcal{E}}$. Internally, hypothesizing $\mathbf{syn}$ collapses the category to $\mathcal{E}$:

Lemma 4.5. *There is an equivalence* $\mathcal{E} \simeq \mathbf{Gl}(\rho)/\mathbf{syn}$.

In topos-theoretic terms, $\mathcal{E}$ is an open subtopos of $\mathbf{Gl}(\rho)$. As an open subtopos, we can present $\mathcal{E}$ internally to $\mathbf{Gl}(\rho)$ through a lex idempotent monad $\bigcirc A = \mathbf{syn} \to A$ [RSS20]. This modality has a strongly disjoint lex idempotent modality, $\bullet A$ [RSS20, Section 3.4]. While we could work with $\bullet$ entirely through this characterization, it is helpful to fix a

definition:

$$
\begin{array}{ccc}
\mathbf{syn} \times A & \longrightarrow & A \\
\downarrow & & \downarrow \\
\mathbf{syn} & \longrightarrow & \bullet A
\end{array}
\tag{4.1}
$$

Intuitively, $\bullet A$ is the portion of A with a trivial $\mathcal{E}$ component. This is even clearer if one calculates the behavior of $\bullet$ on a closed type $A = (E, F, f)$ as $\bullet A = (\mathbf{1}, F, !)$. Just as hypothesizing $\mathbf{syn}$ i.e., working under $\bigcirc$, recovers $\mathcal{E}$ internally to $\mathbf{Gl}(\rho)$, working under $\bullet$ recovers $\mathcal{F}$. Phrased in topos-theoretic terms, $\mathcal{F}$ is a *closed* subtopos of $\mathbf{Gl}(\rho)$.

The final ingredient we must add to our type theory is the *realignment axiom* [OP18, BBC$^+$19, SH21], stating that the following canonical map has an inverse re for any $B : \mathsf{U}$:

$$
\left(\sum\nolimits_{A:\mathsf{U}} [A \cong B] \right) \to \left(\sum\nolimits_{A:\mathbf{syn}\to\mathsf{U}} \prod\nolimits_{z:\mathbf{syn}} A(z) \cong B \right)
\tag{4.2}
$$

Unfolding these conditions yields the following:

Definition 4.6. Fix $B : \mathsf{U}$, $A : \bigcirc\mathsf{U}$, and $\alpha : \prod_{z:\mathbf{syn}} A(z) \cong B$. The *realignment* $\mathsf{re}(B, A, \alpha)$ of B along α is a term of type $\sum_{A^*:\mathsf{U}} A^* \cong B$ satisfying the following condition:

$$
\prod\nolimits_{z:\mathbf{syn}} \mathsf{re}(B, A, \alpha) = (A(z), \alpha(z))
$$

More intuitively, realignment states that a predicate lying over an object in $\mathcal{E}$ can be shifted to lie over an isomorphic object. A proper motivation of realignment is deferred to its use in Section 5, but broadly realignment will be used to satisfy the strict equalities demanded by Definition 3.8 where a priori two constants might agree only up to isomorphism.

Theorem 8.4 of Orton and Pitts [OP18] shows that a Hofmann–Streicher universe satisfies realignment for levelwise decidable propositions. Using the presentation of $\mathbf{Gl}(\rho)$ as a presheaf topos [CJ95], $\mathbf{syn}$ is clearly levelwise decidable and so realignment at $\mathbf{syn}$ is constructively valid. Indeed, for this proposition realignment has a simple and intuitive meaning. To a first approximation, it allows us to take an object in a gluing topos $X \longrightarrow \rho(Y)$ along with an isomorphism $Y \cong Y'$ and perturb the first object to $X \longrightarrow \rho(Y')$. Making this precise (e.g., allowing re to act in an arbitrary context) is only marginally more complex.

Definition 4.7. The language of synthetic Tait computability is extensional type theory with a cumulative hierarchy of universes and a universe of propositions equipped with a distinguished proposition $\mathbf{syn} : \Omega$ such that each universe satisfies the realignment axiom for $\mathbf{syn}$.

This subsection is summarized by the following result, which might be termed the 'fundamental lemma' of STC:

Theorem 4.8. $\mathbf{Gl}(\rho)$ *is a model of STC.*

4.2. **Gluing together cosmoi.** While a model in $\mathbf{Gl}(\rho)$ for a carefully chosen $\mathcal{E}$, $\mathcal{F}$, and ρ is sufficient to prove many results of MLTT [Coq19] the situation for MTT is more complex. Rather than gluing along a single functor, it is necessary to glue along an entire 2-natural transformation of continuous functors between 2-functors of presheaf topoi. We begin by

considering a pair of presheaf cosmoi for the mode theory $\{\mu : n \longrightarrow m\}$ and a 2-natural transformation of right adjoints between them:

$$
\begin{array}{ccc}
\mathcal{E}_n & \xrightarrow{\ \rho_n\ } & \mathcal{F}_n \\
{\scriptstyle f}\downarrow & & \downarrow{\scriptstyle g} \\
\mathcal{E}_m & \xrightarrow[\ \rho_m\ ]{} & \mathcal{F}_m
\end{array}
\tag{4.3}
$$

For simplicity and since we do not require the additional generality, we shall assume that F and G are strict 2-functors and that the 2-natural transformation between them is likewise strict. Let us further assume that f and g preserve finite colimits.

Gluing 'horizontally', we obtain a pair of categories $\mathbf{Gl}(\rho_n)$ and $\mathbf{Gl}(\rho_m)$ and by Theorems 4.3 and 4.8 both are presheaf topoi and models of STC. Artin gluing is functorial, and Diagram 4.3 induce a functor $\mathbf{Gl}(f,g) : \mathbf{Gl}(\rho_n) \longrightarrow \mathbf{Gl}(\rho_m)$ sending (E_n, F_n, x) to $(f(E_n), g(F_n), g(x))$.

Lemma 4.9. $\mathbf{Gl}(f,g) : \mathbf{Gl}(\rho_n) \longrightarrow \mathbf{Gl}(\rho_m)$ *is a right adjoint.*

Proof. While this follows classically from the special adjoint functor theorem, an explicit construction is useful. There is a comparison $\beta : g_! \circ \rho_m \longrightarrow \rho_n \circ f_!$ induced by transposition and the unit of the $f_! \dashv f$. The left adjoint $\mathbf{Gl}(f,g)_!$ sends $f : F \longrightarrow \rho_m(E)$ to $\beta \circ g_!(f) : g_!(F) \longrightarrow \rho_n(f_!(E))$. The isomorphism $[[f,g]_!(X), Y] \cong [X, [f,g](Y)]$ is given component-wise by the isomorphisms associated with $f_! \dashv f$ and $g_! \dashv g$. $\qquad\square$

Remark 4.10. This explicit calculation show that $\pi_n : \mathbf{Gl}(\rho_n) \longrightarrow \mathcal{E}_n$ and $\pi_m : \mathbf{Gl}(\rho_m) \longrightarrow \mathcal{E}_m$ assemble into a natural transformation which satisfies Beck-Chevalley.

Since each $\mathbf{Gl}(\rho_-)$ is a presheaf topos, it supports a model of extensional type theory. We wish to stitch these models together into a single model of MTT with mode theory $\{n \longrightarrow m\}$ using the results of Gratzer et al. [GKNB21]. To do so, we must show that $\mathbf{Gl}(f,g)$ induces a dependent right adjoint between models of MLTT in $\mathbf{Gl}(\rho_n)$ and $\mathbf{Gl}(\rho_m)$. Next, we show this holds if we take the models of extensional type theory in $\mathbf{Gl}(\rho_-)$ as each having universes of types given by a sufficiently large Hofmann–Streicher universe:

Lemma 4.11. *The adjunction $\mathbf{Gl}(f,g)_! \dashv \mathbf{Gl}(f,g)$ induces a dependent right adjoint with respect to sufficiently large Hofmann–Streicher universe $\mathcal{U}$.*

Proof. It suffices to argue that $\mathbf{Gl}(f,g)$ sends a $\mathcal{U}$-small family in $\mathbf{Gl}(\rho_n)$ to a $\mathcal{U}$-small in $\mathbf{Gl}(\rho_m)$. This is proven by e.g., Gratzer et al. [GSS22, Lemma 3.3.7]. $\qquad\square$

As a consequence of Lemma 4.11, we obtain a model of MTT with the mode theory $\{\mu : n \longrightarrow m\}$ which interprets n, m, and μ as $\mathbf{Gl}(\rho_n)$, $\mathbf{Gl}(\rho_m)$, and $\mathbf{Gl}(f,g)$ respectively. This model of MTT is particularly well-behaved: equality is extensional and $\mathbf{Gl}(f,g)$ validates the strong transposition-style elimination rules specified by Birkedal et al. [BCM$^+$20].

Lemma 4.12. *In this model of MTT, $\langle \mu \mid \mathbf{syn}_n \rangle \cong \mathbf{syn}_m$*

Proof. Externally, $\mathbf{syn}_n = (\mathbf{1}, \mathbf{0}, !)$ but g preserves $\mathbf{0}$ while f preserves $\mathbf{1}$, so $\mathbf{Gl}(f,g)(\mathbf{syn}_n) \cong (\mathbf{1}, \mathbf{0}, !) = \mathbf{syn}_m$. $\qquad\square$

Lemma 4.13. *In this model of MTT, $\bigcirc\langle \mu \mid A \rangle \cong \langle \mu \mid \bigcirc A \rangle$ and $\bullet\langle \mu \mid A \rangle \cong \langle \mu \mid \bullet A \rangle$.*

Proof. We consider the only case of $\bigcirc$, as the argument for $\bullet$ is identical. First, we observe that $\mathbf{Gl}(f,g)$ preserves $\bigcirc$ *externally.* That is, there is an isomorphism $\alpha : \mathbf{Gl}(f,g) \circ \bigcirc \cong \bigcirc \circ \mathbf{Gl}(f,g)$. It remains to show that this isomorphism can be internalized. Let us write $\tau_m : \mathcal{T}_m^\bullet \longrightarrow \mathcal{T}_m$ for the universe of types in $\mathbf{Gl}(\rho_m)$ and write τ_n for its counterpart in $\mathbf{Gl}(\rho_n)$. Let us further write i, $\hat{\bigcirc}_m$, and $\hat{\bigcirc}_n$ for the cartesian natural transformations $\mathbf{Gl}(f,g)(\tau_n) \longrightarrow \tau_m$, $\bigcirc\tau_m \longrightarrow \tau_m$, and $\bigcirc\tau_n \longrightarrow \tau_n$ that are used to interpret $\langle \mu \mid - \rangle$ and $\bigcirc$ in both $\mathbf{Gl}(\rho_n)$ and $\mathbf{Gl}(\rho_m)$, respectively.

Unfolding this statement into the model, we must argue that the following pair of maps classify isomorphic families:

$$\mathbf{Gl}(f,g)(\bigcirc\mathcal{T}_n) \xrightarrow{\mathbf{Gl}(f,g)(\hat{\bigcirc})} \mathbf{Gl}(f,g)(\mathcal{T}_n) \xrightarrow{\hspace{2em} i \hspace{2em}} \mathcal{T}_m$$

$$\mathbf{Gl}(f,g)(\bigcirc\mathcal{T}_n) \xrightarrow{\hspace{1.5em} \bigcirc i \circ \alpha \hspace{1.5em}} \bigcirc\mathcal{T}_m \xrightarrow{\hspace{2em} \hat{\bigcirc} \hspace{2em}} \mathcal{T}_m$$

We check that both classify $\mathbf{Gl}(f,g)(\bigcirc\tau_n)$ as both $\mathbf{Gl}(f,g)$ and $\bigcirc$ preserve finite limits. $\square$

Remark 4.14. Technically, $\mathbf{syn}$, $\bigcirc$, and $\bullet$ should be always annotated with a mode. In light of these results, however, we shall omit this annotation and systematically *identify* $\mathbf{syn}_m$ and $\langle \mu \mid \mathbf{syn}_n \rangle$. As both are subterminal, there are no coherence issues in this identification.

Definition 4.15. The language of *multimodal STC* (MSTC) is extensional MTT with a cumulative hierarchy of universes and a universe of propositions such that

- Each mode is equipped with a proposition $\mathbf{syn}$.
- Each universe satisfies the realignment axiom for $\mathbf{syn}$.
- MTT modalities commute with $\mathbf{syn}$, $\bigcirc$, and $\bullet$.

Summarizing the preceding discussion:

Theorem 4.16. $\mathbf{Gl}(\rho_n)$, $\mathbf{Gl}(\rho_m)$, *and* $\mathbf{Gl}(f,g)$ *assemble into a presheaf cosmos and a model of MSTC.*

In fact, it is only a small step from this result to the full fundamental lemma of multimodal STC:

Theorem 4.17. *Given a pair of cosmoi* $F, G : \mathcal{M} \longrightarrow \mathbf{Cat}$ *and a 2-natural transformation* $\rho : F \longrightarrow G$ *such that each* $F(\mu), G(\mu)$ *preserves finite colimits and each* ρ_m *is continuous,* $\mathbf{Gl}(\rho) : \mathcal{M} \longrightarrow \mathbf{Cat}$ *both a presheaf cosmos and a model of MSTC. Furthermore* $\pi_0 : \mathbf{Gl}(\rho) \longrightarrow F$ *is a morphism of cosmoi.*

5. The normalization cosmos

Recall from Section 2.4 the 2-functor of categories of renamings Ren_-. By an identical construction to Example 3.6, we obtain the cosmos of renamings $\mathcal{R}(-) = \mathbf{PSh}(\mathsf{Ren}_-)$ and the 2-natural transformation $\mathbf{i}[-] : \mathsf{Ren}_- \longrightarrow \mathsf{Cx}_-$ acts by precomposition to yield a 2-natural transformation $\mathbf{i}[-]^* : \mathcal{S} \longrightarrow \mathcal{R}$. Theorem 4.17 then yields the following:

Definition 5.1. The normalization cosmos $\mathcal{G}$ is a presheaf cosmos and model of MSTC where $\mathcal{G}(m) = \mathbf{Gl}(\mathbf{i}[m]^*)$.

Remark 5.2. One may explicitly present $\mathbf{Gl}(\mathbf{i}[m]^*)$ as a presheaf category over the *collage* of Ren_m and Cx_m [CJ95]. This is a category whose objects are given by the disjoint union of $\mathsf{Ren}_m \coprod \mathsf{Cx}_m$ and with morphisms defined as follows:

$$[\iota_0(\Delta), \iota_0(\Gamma)] = [\Delta, \Gamma]_{\mathsf{Ren}_m} \qquad\qquad [\iota_1(\Delta), \iota_1(\Gamma)] = [\Delta, \Gamma]_{\mathsf{Cx}_m}$$
$$[\iota_1(\Delta), \iota_0(\Gamma)] = [\Delta, i(\Gamma)]_{\mathsf{Cx}_m} \qquad\qquad [\iota_0(\Delta), \iota_1(\Gamma)] = \emptyset$$

As a further consequence of Theorem 4.17, the projection map $\pi_0 : \mathcal{G} \longrightarrow \mathcal{S}$ is a morphism of cosmoi. In this section, we equip $\mathcal{G}$ with the structure of an MTT cosmos and show that π_0 extends to a morphism of MTT cosmoi.

5.1. **Prerequisites for the normalization cosmos.** Before we extend $\mathcal{G}$ to an MTT cosmos, we import features of $\mathcal{G}$ into the language of MSTC to specialize the latter to this situation. In this section, we begin using the interpretation of MTT to work internally to $\mathcal{G}$ and explicitly record the extensions to MSTC required for the normalization proof.

Notation 5.3 (Dependent open modality). As $\bigcirc A = \mathbf{syn} \to A$, we will write $\bigcirc_z A(z) = (z : \mathbf{syn}) \to A(z)$ for the *dependent* version of the open modality.

Notation 5.4 (Extension types). Given a type A, a proposition ϕ, and an element $a : \phi \to A$, we write $\{A \mid x : \phi \mapsto a(x)\}$ for subtype of A of elements equal to a under ϕ. Formally:

$$\{A \mid x : \phi \mapsto a(x)\} = \textstyle\sum_{a':A} (x : \phi) \to a' = a(x)$$

We treat the coercion $\{A \mid x : \phi \mapsto a(x)\} \to A$ as silent and refer to the equation $a' = a(x)$ as a *boundary condition*.

Recall from Example 3.6 that $\mathcal{S}$ already contains the structure of an MTT cosmos. As a presheaf cosmos, this manifests through a series of constants in the internal language of $\mathcal{S}$. Using Lemma 4.5 we import these constants into $\mathcal{G}$.

Extension 1. *For each $m : \mathcal{M}$, there is a pair of constants $z : \mathbf{syn} \vdash \mathsf{Ty}_m(z) : \mathsf{U}_0 @ m$ and $z : \mathbf{syn}, A : \mathsf{Ty}_m(z) \vdash \mathsf{Tm}_m(z, A) : \mathsf{U}_0 @ m$. These constants are further equipped with operations à la Figure 4 closing them under dependent sums, dependent products, modal types, etc.*

Next, observe that normals, neutrals, and normal types are equipped with an action by renamings, so that they can be structured as presheaves over Ren_-. The decoding operations further organize them into proof-relevant predicates over terms and types e.g., the presheaf of normal types as an object of $\mathcal{G}$ lying over the presheaf of types from $\mathcal{S}(m)$. In fact, because renamings map variables to variables, the collection of variables of a given type organizes into a presheaf over Ren_- and part of an object in $\mathcal{G}$. We import these objects into the internal language as additional constants:

Extension 2. *Given $m : \mathcal{M}$ and $A : \bigcirc_z \mathsf{Ty}_m(z)$, we have constants $\mathsf{Nf}_m(A), \mathsf{Ne}_m(A), \mathsf{V}_m(A) : \{\mathsf{U}_0 \mid z : \mathbf{syn} \mapsto \mathsf{Tm}_m(z, A(z))\}$ and $\mathsf{NfTy}_m : \{\mathsf{U}_0 \mid z : \mathbf{syn} \mapsto \mathsf{Ty}_m(z)\}$.*
We treat the coercion from $\mathsf{V}_m(A)$ to $\mathsf{Ne}_m(A)$ as silent.

Notation 5.5. We frequently omit explicitly passing $z : \mathbf{syn}$ as an argument to $M : \bigcirc X$. For instance, given $A, B : \bigcirc \mathsf{Ty}_m$ we write $\mathsf{Nf}_m(\mathsf{Prod}(A, B))$ not $\mathsf{Nf}_m(\lambda z. \mathsf{Prod}(z, A(z), B(z)))$.

Following Hofmann [Hof99], the constructors for normal forms, neutrals, and normal types can be realized in $\mathbf{PSh}(\mathsf{Ren}_-)$ by a form of higher-order abstract syntax. As $\mathsf{Nf}_m(A)$, $\mathsf{Ne}_m(A)$, and NfTy_m lie over $\mathsf{Tm}_m(A)$ and Ty_m, one can extend this higher-order abstract syntax presentation to $\mathcal{G}$ and realize each normal form, neutral, and normal type as a constant of $\mathsf{Nf}_m(A)$, $\mathsf{Ne}_m(A)$, or NfTy_m which collapses to the appropriate syntactic constant under $z : \mathbf{syn}$. As a simple example, the normal form type for booleans along with the ordinary boolean type former induce maps $\mathsf{bool} : \mathbf{1} \longrightarrow \pi_1(\mathsf{NfTy}_m)$ and $\mathsf{bool} : \mathbf{1} \longrightarrow \mathsf{Ty}_m$ in $\mathbf{PSh}(\mathsf{Ren}_m)$ and $\mathbf{PSh}(\mathsf{Cx}_m)$ respectively. These maps pair together to introduce a morphism $[\![\mathbf{Bool}]\!] : \mathbf{1} \longrightarrow [\![\mathsf{NfTy}_m]\!]$ in $\mathcal{G}(m)$ where we rely on the equation $|\mathsf{bool}| = \mathsf{bool}$ to ensure that these morphisms fit into the commutative square required by $\mathcal{G}(m)$. The full collection of constants is specified in Figure 5.

Extension 3. *There are constants internalizing normals, neutrals, and normal types.*

Finally, inspecting Definition 5.1 reveals that modalities are interpreted by functors which are both left and right adjoints as they preserve all (co)limits. As a result, modalities preserve coproducts:

Extension 4. $\langle \mu \mid A + B \rangle \cong \langle \mu \mid A \rangle + \langle \mu \mid B \rangle$

5.2. **The MTT cosmos.** We now extend $\mathcal{G}$ to an MTT cosmos. To ensure that π_0 induces a morphism of MTT cosmoi, it suffices to ensure that each constant we add to $\mathcal{G}$ is equal to the corresponding piece of $\mathcal{S}$ as internalized by Extension 1 under $z : \mathbf{syn}$.

The universe of computable types and terms. We begin with the definition of types and terms in this cosmos. Concretely, we require the following for each $m : \mathcal{M}$:

$$\mathsf{Ty}_m^* : \{\mathsf{U}_2 \mid z : \mathbf{syn} \mapsto \mathsf{Ty}_m(z)\}$$
$$\mathsf{Tm}_m^* : (A : \mathsf{Ty}_m^*) \to \{\mathsf{U}_1 \mid z : \mathbf{syn} \mapsto \mathsf{Tm}_m(z, A)\}$$

We start with the following putative definition of types:

$$\begin{aligned}&\mathbf{record}\ T : \mathsf{U}_2\ \mathbf{where} &&(5.1)\\&\quad \mathsf{code} : \mathsf{NfTy}_m\\&\quad \mathsf{pred} : \{\mathsf{U}_1 \mid z : \mathbf{syn} \mapsto \mathsf{Tm}_m(z, \mathsf{code})\}\\&\quad \mathsf{reflect} : \{\mathsf{Ne}_m(\mathsf{code}) \to \mathsf{pred} \mid \mathbf{syn} \mapsto \mathsf{id}\}\\&\quad \mathsf{reify} : \{\mathsf{pred} \to \mathsf{Nf}_m(\mathsf{code}) \mid \mathbf{syn} \mapsto \mathsf{id}\}\end{aligned}$$

In prose, $A : T$ contains the code of a normal type $A.\mathsf{code}$ as well as a proof-relevant predicate on the elements of $A.\mathsf{code}$.

The last two fields ensure that (1) all elements tracked by this predicate can be assigned normal forms, and (2) all neutrals lie within the predicate. We write $\downarrow_A$ and $\uparrow_A$ for $A.\mathsf{reify}$ and $A.\mathsf{reflect}$. Of the two, the reify is the crucial operation needed for the normalization algorithm: it ensures that computable elements can be given normal forms. Tait [Tai67], however, has shown that the pair of operations is necessary to close all type formers under just reify.

We cannot simply define $\mathsf{Ty}_m^* = T$, as T does not satisfy the equation $z : \mathbf{syn} \vdash T = \mathsf{Ty}_m(z)$. It does, however, satisfy this condition up to isomorphism: under $z : \mathbf{syn}$, the types

$\mathbf{Prod} : (A : \mathsf{NfTy}_m)(B : \mathsf{V}_m(A) \to \mathsf{NfTy}_m) \to \mathsf{NfTy}_m$

$\mathbf{Sum} : (A : \mathsf{NfTy}_m)(B : \mathsf{V}_m(A) \to \mathsf{NfTy}_m) \to \mathsf{NfTy}_m$

$\mathbf{Id} : (A : \mathsf{NfTy}_m) \to \mathsf{Nf}_m(A) \to \mathsf{Nf}_m(A) \to \mathsf{NfTy}_m$

$\mathbf{Bool} : \mathsf{NfTy}_m$

$\mathbf{Mod}_\mu : (\mu \mid \mathsf{NfTy}_n) \to \mathsf{NfTy}_m$

$\mathbf{lam} : (A : \bigcirc\mathsf{Ty}_m)(B : \bigcirc\mathsf{Tm}_m(A) \to \bigcirc\mathsf{Ty}_m)$
  $\to ((a : \mathsf{V}_m(A)) \to \mathsf{Nf}_m(B(a))) \to \mathsf{Nf}_m(\mathsf{Prod}(A, B))$

$\mathbf{app} : (\mu \mid A : \bigcirc\mathsf{Ty}_m)(B : \bigcirc\mathsf{Tm}_m(A) \to \bigcirc\mathsf{Ty}_m)$
  $\to \mathsf{Ne}_m(\mathsf{Prod}(A, B)) \to (\mu \mid a : \mathsf{Nf}_m(A)) \to \mathsf{Ne}_m(B(a))$

$\mathbf{up} : \mathsf{Ne}_m(\mathsf{Bool}) \to \mathsf{Nf}_m(\mathsf{Bool})$

$\mathbf{tt}, \mathbf{ff} : \mathsf{Nf}_m(\mathsf{Bool})$

$\mathbf{if} : (A : \mathsf{V}_m(\mathsf{Bool}) \to \mathsf{NfTy}_m)$
  $\to \mathsf{Nf}_m(A(\mathsf{true})) \to \mathsf{Nf}_m(A(\mathsf{false})) \to (b : \mathsf{Ne}_m(\mathsf{Bool})) \to \mathsf{Ne}_m(A(b))$

$\mathbf{up} : (A : \bigcirc\mathsf{Ty}_m)(a_0, a_1 : \bigcirc\mathsf{Tm}_m(A))$
  $\to \mathsf{Ne}_m(\mathsf{Id}(A, a_0, a_1)) \to \mathsf{Nf}_m(\mathsf{Id}(A, a_0, a_1))$

$\mathbf{refl} : (A : \bigcirc_z\mathsf{Ty}_m(z))(a : \bigcirc_z\mathsf{Tm}_m(z, A(z))) \to \mathsf{Nf}_m(\mathsf{Id}(A, a, a))$

$\mathbf{J} : (A : \bigcirc\mathsf{Ty}_m)\,(B : (a_0, a_1 : \mathsf{V}_m(A))(p : \mathsf{V}_m(\mathsf{Id}(A, a_0, a_1))) \to \mathsf{NfTy}_m)$
  $\to ((a : \mathsf{V}_m(A)) \to \mathsf{Nf}_m(B(a, a, \mathsf{refl}(a))))\,(a_0, a_1 : \bigcirc_z\mathsf{Tm}_m(A))(p : \mathsf{Ne}_m(\mathsf{Id}(A, a_0, a_1)))$
  $\to \mathsf{Ne}_m(B(a_0, a_1, p))$

$\mathbf{up} : (\mu \mid A : \mathsf{Ty}_n) \to \mathsf{Ne}_m(\mathsf{Mod}_\mu(A)) \to \mathsf{Nf}_m(\mathsf{Mod}_\mu(A))$

$\mathbf{mod}_\mu : (\mu \mid A : \bigcirc\mathsf{Ty}_n)(\mu \mid \mathsf{Nf}_n(A)) \to \mathsf{Nf}_m(\lambda z.\, \mathsf{Mod}_\mu(z, A(z)))$

$\mathbf{letmod}_{\mu;\nu} : (\nu \circ \mu \mid A : \bigcirc\mathsf{Ty}_n)\,(B : (\nu \mid a : \mathsf{V}_m(\mathsf{Mod}_\mu(A))) \to \mathsf{NfTy}_o)$
  $\to ((\nu \circ \mu \mid a : \mathsf{V}_n(A)) \to \mathsf{Nf}_o(B(\mathsf{m}_\mu(a)))) \to (\nu \mid a : \mathsf{Ne}_m(\mathsf{Mod}_\mu(A))) \to \mathsf{Ne}_o(B(a))$

$\mathbf{Uni} : \mathsf{NfTy}_m$

$\mathbf{El} : \mathsf{Nf}_m(\mathsf{Uni}) \to \mathsf{NfTy}_m$

$\mathbf{up} : \mathsf{Ne}_m(\mathsf{Uni}) \to \mathsf{Nf}_m(\mathsf{Uni})$

$\widehat{\mathbf{Mod}}_\mu : (\mu \mid \mathsf{Nf}_n(\mathsf{Uni})) \to \mathsf{Nf}_m(\mathsf{Uni})$

$\mathbf{dec}^{\rhd}_{\widehat{\mathbf{Mod}}_\mu} : (\mu \mid A : \mathsf{Nf}_n(\mathsf{Uni})) \to \mathsf{Nf}_m(\mathsf{Mod}_\mu(A)) \to \mathsf{Nf}_m(\mathsf{El}(\widehat{\mathsf{Mod}}(A)))$

$\mathbf{dec}^{\lhd}_{\widehat{\mathbf{Mod}}_\mu} : (\mu \mid A : \mathsf{Nf}_n(\mathsf{Uni})) \to \mathsf{Ne}_m(\mathsf{El}(\widehat{\mathsf{Mod}}(A))) \to \mathsf{Ne}_m(\mathsf{Mod}_\mu(A))$

Figure 5: Neutral and normal forms, internally

of pred, reflect, and reify collapse to singletons, while the type of code collapses to $\mathsf{Ty}_m(z)$ by Extension 2:

$$\alpha_{\bigcirc} : \prod_{z:\mathbf{syn}} T \cong \mathsf{Ty}_m(z)$$

$$\alpha_{\bigcirc}(z, A) = A.\mathsf{code}$$

Observe $(\mathsf{Ty}_m, \alpha_{\bigcirc}) : \sum_{A:\bigcirc\mathsf{U}} \prod_{z:\mathbf{syn}} A(z) \cong T$, so the realignment axiom of Definition 4.6 applies and we can define

$$(\mathsf{Ty}_m^*, \alpha) = \mathsf{re}(T, \mathsf{Ty}_m, \alpha_{\bigcirc}) \tag{5.2}$$

The equation $z : \mathbf{syn} \vdash \mathsf{Ty}_m^* = \mathsf{Ty}_m(z)$ follows immediately from the second half of Definition 4.6. On elements $A : \mathsf{Ty}_m^*$, this implies $z : \mathbf{syn} \vdash A = \alpha(A).\mathsf{code}$. For readability, we continue to use record notation to manipulate Ty_m^*.

Given $A : \mathsf{Ty}_m^*$, we define $\mathsf{Tm}_m^*(A)$:

$$\mathsf{Tm}_m^*(A) = A.\mathsf{pred} : \{\mathsf{U}_1 \mid z : \mathbf{syn} \mapsto \mathsf{Tm}_m(z, A)\} \tag{5.3}$$

To see that this is well-typed, we must show $\mathsf{Tm}_m^*(A) = \mathsf{Tm}_m(z, A)$ given $z : \mathbf{syn}$. The type of $A.\mathsf{code}$ in Construction 5.1 ensures $\mathsf{Tm}_m^*(A) = \mathsf{Tm}_m(z, A.\mathsf{code})$. We have observed that $A = A.\mathsf{code}$ under $z : \mathbf{syn}$ so $\mathsf{Tm}_m^*(A) = \mathsf{Tm}_m(z, A)$.

Type connectives. It remains only to close $(\mathsf{Ty}_m^*, \mathsf{Tm}_m^*)$ under all connectives in such a way that each connective lies over the corresponding one in $(\mathsf{Ty}_m, \mathsf{Tm}_m)$. For mode-local connectives, these constructions are very similar to those given by Sterling [Ste21] (Lemmas 5.8, 5.9, 5.10, and 5.11). Modal types and dependent products, however, involve modalities and thus are different than the other connectives (Lemmas 5.6 and 5.7).

Lemma 5.6. $(\mathsf{Ty}_m^*, \mathsf{Tm}_m^*)$ *is closed under dependent products and the relevant constants lift those of* Ty_m *(i.e., under an assymption* $z : \mathbf{syn}$, *they agree with those of* Ty_m *and* Tm_m):

$$\mathsf{Prod}^* : (\mu \mid A : \mathsf{Ty}_n^*)(B : (\mu \mid \mathsf{Tm}_n^*(A)) \to \mathsf{Ty}_m^*) \to \mathsf{Ty}_m^*$$

$$\alpha_{\mathsf{Prod}^*} : (\mu \mid A : \mathsf{Ty}_n^*)(B : (\mu \mid \mathsf{Tm}_n^*(A)) \to \mathsf{Ty}_m^*)$$
$$\to \mathsf{Tm}_m^*(\mathsf{Prod}^*(A, B)) \cong [(\mu \mid a : \mathsf{Tm}_n^*(A)) \to \mathsf{Tm}_m^*(B(a))]$$

Proof. We must define two constants (Prod^* and α_{Prod^*}) with the aforementioned types. We begin by fixing $(\mu \mid A : \mathsf{Ty}_m^*)$ and $B : (\mu \mid \mathsf{Tm}_n^*(A)) \to \mathsf{Ty}_m^*$ and define Φ as follows:

$$\Phi = (\mu \mid a : \mathsf{Tm}_n^*(A)) \to \mathsf{Tm}_m^*(B(a))$$

Observe under $z : \mathbf{syn}$, the following equality holds:

$$\Phi = (\mu \mid a : \mathsf{Tm}_n(z, A)) \to \mathsf{Tm}_m(B(z, a))$$

We may apply realignment using $\alpha_{\mathsf{Prod}}(z) : \mathsf{Tm}_m(z, \mathsf{Prod}(z, A, B)) \cong \Phi$. This realignment yields a type Ψ and isomorphism $\beta : \Psi \cong \Phi$. Under $z : \mathbf{syn}$, these restrict to $\mathsf{Tm}_m(z, \mathsf{Prod}(z, A, B))$ and $\alpha_{\mathsf{Prod}}(z)$ respectively.

With these to hand we define Prod^* and α_{Prod^*} as follows:

$$\mathsf{Prod}^*(A, B).\mathsf{code} = \mathbf{Prod}(A.\mathsf{code}, \lambda v.\ B(\uparrow_A v).\mathsf{code})$$

$$\mathsf{Prod}^*(A, B).\mathsf{pred} = \Psi$$

$$\mathsf{Prod}^*(A, B).\mathsf{reflect} = \lambda e.\ \beta^{-1}(\lambda a.\ \uparrow_{B(a)} \mathbf{app}(e, \downarrow_A a))$$

$$\mathsf{Prod}^*(A, B).\mathsf{reify} = \lambda f.\ \mathbf{lam}(\lambda v.\ \downarrow_{B(\uparrow_A v)} \beta(f)(\uparrow_A v))$$

$$\alpha_{\mathsf{Prod}^*} = \beta$$

It remains to check a variety of boundary conditions under $z : \mathbf{syn}$. In particular, we must show that $\mathsf{Prod}^*(A, B) = \mathsf{Prod}(z, A, B)$ and that $\mathsf{reflect}$ and reify become the identity. These follow directly from assumptions about A, B, and the boundaries of various constructors. For instance

$$\mathsf{Prod}^*(A, B) = \mathsf{Prod}^*(A, B).\mathsf{code}$$
$$= \mathbf{Prod}(A.\mathsf{code}, \lambda v. \ B(\downarrow_A v).\mathsf{code})$$
$$= \mathsf{Prod}(z, A.\mathsf{code}, \lambda v. \ B(\downarrow_A v).\mathsf{code})$$
$$= \mathsf{Prod}(z, A, \lambda v. \ B(\downarrow_A v))$$
$$= \mathsf{Prod}(z, A, B) \qquad\qquad \square$$

Lemma 5.7. $(\mathsf{Ty}^*_m, \mathsf{Tm}^*_m)$ *is closed under modal types and the four relevant constants* $(\mathsf{Mod}^*_\mu,$ $\mathsf{m}^*_\mu, \mathsf{letmod}^*_{\mu;\nu},$ *and* $\mathsf{Mod/beta}^*_{\mu;\nu})$ *lift those of their counterparts in* Ty_m *and* Tm_m.

Proof. Fix a modality $\mu : n \longrightarrow m$. In this case we define the four constants Mod_μ, m_μ, $\mathsf{letmod}_{\mu;\nu}$, and $\mathsf{Mod/beta}_{\mu;\nu}$ described in Section 3.1, subject to the expected boundary conditions. Fix a variable $A : \mathsf{Ty}^*_n$ under the modal annotation μ i.e., $(\mu \mid A : \mathsf{Ty}^*_n)$. We define the unaligned predicate as follows:

$$\begin{aligned} &\mathbf{record} \ \Phi : \mathsf{U}_1 \ \mathbf{where} \\ &\quad \mathsf{tm} : \mathsf{Nf}_m(\mathsf{Mod}_\mu(A)) \\ &\quad \mathsf{prf} : \bullet \left(\begin{array}{l} \sum_{e:\mathsf{Ne}_m(\mathsf{Mod}_\mu(A))} \mathsf{tm} = \mathbf{up}(e) \\ + \sum_{a:\langle \mu \mid A.\mathsf{pred}\rangle} \mathsf{tm} = \mathbf{mod}_\mu(\downarrow_A a) \end{array} \right) \end{aligned}$$

For the first time, we have used the closed modality $\bullet$ to explicitly tweak the proof-relevant predicate. Intuitively, Φ is a predicate on $\mathsf{Tm}_m(z, \mathsf{Mod}_\mu(z, A))$ and tm ensures that this predicate tracks elements with normals forms. The second field, moreover, ensures that these normal are either neutral or $\mathsf{mod}_\mu(a)$ where a is computable. Without the closed modality shielding the second field of Φ, however, this could never have the correct extent along $z : \mathbf{syn}$. Using $\bigcirc \bullet X \cong \mathbf{1}$ and the boundary of $\mathsf{Nf}_m(\mathsf{Mod}_\mu(A))$, we can now define the following isomorphism:

$$\alpha_\bigcirc(z, p) = p.\mathsf{tm} : \prod_{z:\mathbf{syn}} \Phi \cong \mathsf{Tm}_m(z, \mathsf{Mod}_\mu(z, A))$$

Realigning Φ along $\alpha_\bigcirc$, we obtain Ψ and $\alpha : \Psi \cong \Phi$ which under $z : \mathbf{syn}$ become $\mathsf{Tm}_m(z, \mathsf{Mod}_\mu(z, A))$ and $\alpha_\bigcirc$.

We now define Mod^*_μ:

$$\mathsf{Mod}^*_\mu(A).\mathsf{code} = \mathbf{Mod}_\mu(A.\mathsf{code})$$
$$\mathsf{Mod}^*_\mu(A).\mathsf{pred} = \Psi$$
$$\mathsf{Mod}^*_\mu(A).\mathsf{reflect} = \lambda e. \ \alpha^{-1}\langle \mathbf{up}(e), \eta_\bullet \iota_1 \langle e, \star\rangle\rangle$$
$$\mathsf{Mod}^*_\mu(A).\mathsf{reify} = \lambda m. \ \alpha(m).\mathsf{tm}$$

Unlike Lemma 5.6, the introduction and elimination principles are not automatically obtained from α and they must be constructed separately:

$$\mathsf{m}^*_\mu(A, a) = \alpha^{-1}\langle \mathbf{mod}_\mu(\downarrow_A a), \eta_\bullet \iota_2 \langle a, \star\rangle\rangle$$

It remains to define the elimination principle $\mathsf{letmod}^*_{\mu;\nu}$. This is an involved affair and we describe it step-by-step. Begin by fixing $\nu : m \longrightarrow o$ along with the following:

$$B : (\nu \mid \mathsf{Tm}^*_m(\mathsf{Mod}^*_\mu(A))) \to \mathsf{Ty}_o$$

$$b : (\nu \circ \mu \mid x : \mathsf{Tm}^*_n(A)) \to \mathsf{Tm}^*_o(B(\mathsf{m}^*_\mu(A, x)))$$
$$(\nu \mid p : \mathsf{Tm}^*_m(\mathsf{Mod}^*_\mu(A)))$$

We must construct an element of $\mathsf{Tm}^*_o(B(a))$. We begin by inspecting p. As MTT modalities in extensional MTT commute with dependent sums, equality, $\bullet$, and—by Extension 4—with finite coproducts, p can be decomposed into the following:

$$(\nu \mid \mathsf{tm} : \mathsf{Nf}_m(\mathsf{Mod}_\mu(A)))$$

$$\mathsf{prf} : \bullet \left(\begin{array}{l} \sum_{e:\langle \nu \mid \mathsf{Ne}_m(\mathsf{Mod}_\mu(A))\rangle} \mathsf{mod}_\nu(\mathsf{tm}) = \mathbf{up} \circledast e \\ + \sum_{a:\langle \nu \circ \mu \mid A.\mathsf{pred}\rangle} \mathsf{mod}_\nu(\mathsf{tm}) = (\mathbf{mod}_\mu \circ \downarrow_A) \circledast a \end{array} \right)$$

Recall from Diagram 4.1 that $\bullet X$ is a pushout of $\mathbf{syn}$ and X. To define a map out of $\bullet X$, therefore, it suffices to define a map out of X which is constant assuming $z : \mathbf{syn}$. We conclude by scrutinizing prf:

$$\begin{cases} \uparrow \mathbf{letmod}_{\mu;\nu}(A, \lambda v.\ B(\uparrow v).\mathsf{code}, \lambda x.\ \downarrow b(\uparrow x), e) & \text{if } \mathsf{prf} = \iota_1(\mathsf{mod}_\nu(e), \_) \\ b(a) & \text{if } \mathsf{prf} = \iota_2(\mathsf{mod}_\nu(a), \_) \end{cases}$$

Given $z : \mathbf{syn}$, both branches collapse to $\mathsf{letmod}_{\mu;\nu}(z, A, B, b, \mathsf{tm})$ so this yields a well-defined map. The boundary conditions follow from routine computations. □

Lemma 5.8. $(\mathsf{Ty}^*_m, \mathsf{Tm}^*_m)$ *is closed under dependent sums via:*

$$\mathsf{Sig}^*(A, B) : \mathsf{Ty}^*_m$$
$$\alpha_{\mathsf{Sig}^*} : \mathsf{Tm}_m(\mathsf{Sig}^*(A, B)) \cong \sum_{a:\mathsf{Tm}^*_m(A)} \mathsf{Tm}^*_m(B(a))$$

Moreover, assuming $z : \mathbf{syn}$ *then* $\mathsf{Sig}^* = \mathsf{Sig}$ *and* $\alpha_{\mathsf{Sig}^*} = \alpha_{\mathsf{Sig}}$.

Proof. Fixing $A : \mathsf{Ty}^*_m$ and $B : \mathsf{Tm}^*_m(A) \to \mathsf{Ty}^*_m$. We begin by applying realignment to the following:

$$\left(\sum_{a:A.\mathsf{pred}} B(a).\mathsf{pred}, \alpha_{\mathsf{Sig}(z)} \right)$$

This produces $\Psi : \mathsf{U}_1$ and $\alpha_{\mathsf{Sig}^*} : \Psi \cong \sum_{a:A.\mathsf{pred}} B(a).\mathsf{pred}$ such that under the assumption $z : \mathbf{syn}$ the following holds:

$$\Psi = \mathsf{Tm}_m(\mathsf{Sig}(z, A, B)) \qquad \alpha_{\mathsf{Sig}^*} = \alpha_{\mathsf{Sig}}(z)$$

We now define $\mathsf{Sig}^*(A, B)$ as follows:

$$\mathsf{Sig}^*(A, B).\mathsf{code} = \mathbf{Sum}(A.\mathsf{code}, \lambda v.\ B.\mathsf{code}(\uparrow_A v))$$
$$\mathsf{Sig}^*(A, B).\mathsf{pred} = \Psi$$
$$\mathsf{Sig}^*(A, B).\mathsf{reflect} = \lambda e.\ \alpha^{-1}_{\mathsf{Sig}^*}\langle \uparrow_A(\mathbf{proj}_0(e)), \uparrow_{B(\uparrow_A(\mathbf{proj}_0(e)))}(\mathbf{proj}_1(e)) \rangle$$
$$\mathsf{Sig}^*(A, B).\mathsf{reify} = \lambda p.\ \mathbf{pair}(\downarrow_A(\alpha_{\mathsf{Sig}^*}p.0), \downarrow_{B(\alpha_{\mathsf{Sig}^*}p.0)}(\alpha_{\mathsf{Sig}^*}p.1))$$

The fact that $\downarrow$ and $\uparrow$ lie over the identity follows directly from the β and η laws of dependent sums in MTT. We show the calculations for $\uparrow$. Fix $z : \mathbf{syn}$:

$$\begin{aligned} \uparrow_{\mathsf{Sig}^*(A,B)}(e) &= \alpha^{-1}_{\mathsf{Sig}^*}\langle \uparrow_A(\mathbf{proj}_0(e)), \uparrow_{B(\uparrow_A(\mathbf{proj}_0(e)))}(\mathbf{proj}_1(e)) \rangle \\ &= \alpha^{-1}_{\mathsf{Sig}}\langle \mathbf{proj}_0(e), \mathbf{proj}_1(e) \rangle \\ &= \alpha^{-1}_{\mathsf{Sig}}\langle \alpha_{\mathsf{Sig}(A,B)}(e)_0, \alpha_{\mathsf{Sig}(A,B)}(e)_1 \rangle \\ &= e \end{aligned}$$

The fact that $\mathsf{Sig}^*(A,B).\mathsf{code}$ and $\mathsf{Sig}^*(A,B).\mathsf{pred}$ lie over $\mathsf{Sig}(A,B)$ and $\mathsf{Tm}_m(z,\mathsf{Sig}(z,A,B))$ follows from their definition and realignment. □

Lemma 5.9. $(\mathsf{Ty}_m^*, \mathsf{Tm}_m^*)$ *is closed under booleans and the relevant constants lie over their counterparts in* $(\mathsf{Ty}_m, \mathsf{Tm}_m)$.

Proof. We must implement the following constants:

$$\mathsf{Bool}^* : \{\mathsf{Ty}_m^* \mid z : \mathbf{syn} \mapsto \mathsf{Bool}(z)\}$$
$$\mathsf{true}^* : \{\mathsf{Tm}_m^*(\mathsf{Bool}^*) \mid z : \mathbf{syn} \mapsto \mathsf{true}\}$$
$$\mathsf{false}^* : \{\mathsf{Tm}_m^*(\mathsf{Bool}^*) \mid z : \mathbf{syn} \mapsto \mathsf{false}\}$$
$$\mathsf{if}^* : (A : \mathsf{Tm}_m^*(\mathsf{Bool}^*) \to \mathsf{Ty}_m^*)$$
$$\to \mathsf{Tm}_m^*(A(\mathsf{true}^*))$$
$$\to \mathsf{Tm}_m^*(A(\mathsf{false}^*))$$
$$\to (b : \mathsf{Tm}_m^*(\mathsf{Bool}^*))$$
$$\to \{\mathsf{Tm}_m^*(A(b)) \mid z : \mathbf{syn} \mapsto \mathsf{if}(A,t,f,b)\}$$
$$\_ : (A : \mathsf{Tm}_m^*(\mathsf{Bool}^*) \to \mathsf{Ty}_m^*)$$
$$\to (t : \mathsf{Tm}_m^*(A(\mathsf{true}^*)))$$
$$\to (f : \mathsf{Tm}_m^*(A(\mathsf{false}^*)))$$
$$\to (\mathsf{if}^*(A,t,f,\mathsf{true}^*) = t) \times (\mathsf{if}^*(A,t,f,\mathsf{false}^*) = f)$$

First, we define Φ by realignment:

$$\mathbf{record}\ \Phi : \{\mathsf{U}_1 \mid z : \mathbf{syn} \mapsto \mathsf{Tm}_m(z,\mathsf{Bool})\}\ \mathbf{where}$$
$$\mathsf{tm} : \mathsf{Nf}_m(\mathsf{Bool})$$
$$\mathsf{prf} : \bullet\left(\begin{array}{c} \sum_{e:\mathsf{Ne}_m(\mathsf{Bool})} \mathsf{tm} = \mathbf{up}(e) \\ + \sum_{b:\mathbf{2}} \mathsf{tm} = \mathsf{rec}_\mathbf{2}(b;\mathsf{tt};\mathsf{ff})) \end{array} \right)$$

In the above, we have used $\mathsf{rec}_\mathbf{2}$ for the ordinary elimination principle for $\mathbf{2}$ in $\mathcal{G}$. We have opted for the names $\mathbf{2}$ and $\mathsf{rec}_\mathbf{2}$ in the hopes of avoiding ambiguity with Bool, $\mathbf{if}$, and if.

We may now define Bool^*:

$$\mathsf{Bool}^*.\mathsf{code} = \mathbf{Bool}$$
$$\mathsf{Bool}^*.\mathsf{pred} = \Phi$$
$$\mathsf{Bool}^*.\mathsf{reflect} = \lambda e.\langle \mathbf{up}(e), \eta(\iota_1(e,\star))\rangle$$
$$\mathsf{Bool}^*.\mathsf{reify} = \lambda b.\ b.\mathsf{tm}$$

It remains to define the introduction and elimination forms.

$$\mathsf{true}^* = \langle \mathbf{tt}, \eta(\iota_2(0,\star))\rangle$$
$$\mathsf{false}^* = \langle \mathbf{ff}, \eta(\iota_2(1,\star))\rangle$$

The elimination form is defined by constructing a map out of $\bullet X$, by taking advantage of its definition as a pushout (Diagram 4.1):

$$\mathsf{if}^*(A,t_0,t_1,b = \langle \mathsf{tm}, \mathsf{prf}\rangle) =$$

$$\begin{cases} \mathsf{if}(z, A, t_0, t_1, b) & \mathsf{prf} = \iota_1(z) \\ \downarrow_{A(b)}\mathbf{if}(\lambda v.\ A(\uparrow v).\mathsf{code}, \downarrow t_0, \downarrow t_1, e) & \mathsf{prf} = \iota_2(\iota_1(e, \_)) \\ \mathsf{rec_2}(b_0; t_0; t_1) & \mathsf{prf} = \iota_2(\iota_2(b_0, \_)) \end{cases}$$

In this definition, three different incarnations of the elimination rule for booleans are used. The first branch deals uses $\mathsf{if}(z, \dots)$ which is the elimination rule from the syntactic model, the second uses the neutral form $\mathbf{if}$ associated to if, and the third is the "ordinary" elimination principle for booleans available within the model. □

Lemma 5.10. $(\mathsf{Ty}^*_m, \mathsf{Tm}^*_m)$ *is closed under intensional identity types and the relevant constants lie over their counterparts in* $(\mathsf{Ty}_m, \mathsf{Tm}_m)$.

Proof. We must implement the following constants:

$$\begin{aligned} \mathsf{Id}^* : {}& (A : \mathsf{Ty}^*_m)(a_0, a_1 : \mathsf{Tm}^*_m(A)) \\ & \to \{\mathsf{Ty}^*_m \mid z : \mathbf{syn} \mapsto \mathsf{Id}(z, A, a_0, a_1)\} \\ \mathsf{refl}^* : {}& (A : \mathsf{Ty}^*_m)(a : \mathsf{Tm}^*_m(A)) \\ & \to \{\mathsf{Tm}^*_m(\mathsf{Id}^*(A, a, a)) \mid z : \mathbf{syn} \mapsto \mathsf{refl}(z, A, a)\} \\ \mathsf{J}^* : {}& (A : \mathsf{Ty}^*_m) \\ & \to (B : (a_0, a_1 : \mathsf{Tm}^*_m(A)) \to \mathsf{Tm}^*_m(\mathsf{Id}^*(A, a_0, a_1)) \to \mathsf{Ty}^*_m) \\ & \to (b : (a : \mathsf{Tm}^*_m(A)) \to \mathsf{Tm}^*_m(B(a, a, \mathsf{refl}(a)))) \\ & \to (a_0, a_1 : \mathsf{Tm}^*_m(A))(p : \mathsf{Tm}^*_m(\mathsf{Id}^*(A, a_0, a_1))) \\ & \to \{\mathsf{Tm}^*_m(B(a_0, a_1, p)) \mid z : \mathbf{syn} \mapsto \mathsf{J}(z, B, b, p)\} \\ {}_\_ : {}& (A : \mathsf{Ty}^*_m) \\ & \to (B : (a_0, a_1 : \mathsf{Tm}^*_m(A)) \to \mathsf{Tm}^*_m(\mathsf{Id}^*(A, a_0, a_1)) \to \mathsf{Ty}^*_m) \\ & \to (b : (a : \mathsf{Tm}^*_m(A)) \to \mathsf{Tm}^*_m(B(a, a, \mathsf{refl}(a)))) \\ & \to (a : \mathsf{Tm}^*_m(A)) \to \mathsf{J}^*(A, B, b, \mathsf{refl}^*(a)) = b(a) \end{aligned}$$

Fix $A : \mathsf{Ty}^*_m$ and $a_0, a_1 : \mathsf{Tm}^*_m(A)$. Just as with the normalization structure for booleans, we begin by defining Φ by realignment:

$$\begin{aligned} & \mathbf{record}\ \Phi : \{\mathsf{U}_1 \mid z : \mathbf{syn} \mapsto \mathsf{Tm}_m(z, \mathsf{Id}(A, a_0, a_1))\}\ \mathbf{where} \\ & \quad \mathsf{tm} : \mathsf{Nf}_m(\mathsf{Id}(A, a_0, a_1)) \\ & \quad \mathsf{prf} : \bullet\left(\begin{matrix} \sum_{e:\mathsf{Ne}_m(\mathsf{Id}(A, a_0, a_1))} \mathsf{tm} = \mathbf{up}(e) \\ + \sum_{a:A.\mathsf{pred}} a_0 = a_1 \times \mathsf{tm} = \mathsf{refl}(\downarrow_A a) \end{matrix} \right) \end{aligned}$$

We now define Id^*:

$$\begin{aligned} \mathsf{Id}^*(A, a_0, a_1).\mathsf{code} &= \mathsf{Id}_{\mathsf{code}A}(\downarrow_A a_0, \downarrow_A a_1) \\ \mathsf{Id}^*(A, a_0, a_1).\mathsf{pred} &= \Phi \\ \mathsf{Id}^*(A, a_0, a_1).\mathsf{reflect} &= \lambda e. \langle \mathbf{up}(e), \eta(\iota_1(e, \star)) \rangle \\ \mathsf{Id}^*(A, a_0, a_1).\mathsf{reify} &= \lambda p.\ p.\mathsf{tm} \end{aligned}$$

We define reflexivity by $\mathsf{refl}^* = \langle \mathsf{refl}, \eta(\iota_2(\star, \star, \star)) \rangle$. Finally, the elimination principle is defined using the induction principle for $\bullet X$.

$$\mathsf{J}^*(B, b, a_0, a_1, p = \langle \mathsf{tm}, \mathsf{prf} \rangle) =$$

$$\begin{cases} \mathsf{J}(z, B, b, a_0, a_1, p) & \mathsf{prf} = \iota_1(z) \\ \downarrow\mathbf{J}(\lambda l, r, p.B(\uparrow l, \uparrow r, \uparrow p).\mathsf{code}, \lambda a.\downarrow b(\uparrow a), e) & \mathsf{prf} = \iota_2(\iota_1(e, \_)) \\ b(a_0) & q = \iota_2(\iota_2(\_, \_, \_)) \end{cases} \qquad \square$$

Lemma 5.11. $(\mathsf{Ty}_m^*, \mathsf{Tm}_m^*)$ *is closed under a universe and the relevant constants lie over their counterparts in* $(\mathsf{Ty}_m, \mathsf{Tm}_m)$.

Proof. We begin by constructing the two constants for the universe and the decoding family:

$$\mathsf{Uni}^* : \{\mathsf{Ty}_m^* \mid z : \mathbf{syn} \mapsto \mathsf{Uni}\}$$

$$\mathsf{El}^* : (A : \mathsf{Tm}_m^*(\mathsf{Uni}^*)) \to \{\mathsf{Ty}_m^* \mid z : \mathbf{syn} \mapsto \mathsf{El}(A)\}$$

At this point we take advantage of the fact that pred is an element of U_1; in particular, we observe that U_0 is small enough to fit inside U_1.

We may then define Ψ by realigning the following element of U_1 along the evident isomorphism to $\mathsf{Tm}_m^*(z, \mathsf{Uni}(z))$:

$$\begin{aligned} &\mathbf{record}\ \Psi : \{\mathsf{U}_1 \mid z : \mathbf{syn} \mapsto \mathsf{Tm}_m^*(z, \mathsf{Uni})\}\ \mathbf{where} \\ &\quad \mathsf{code} : \mathsf{Nf}_m(\mathsf{Uni}) \\ &\quad \mathsf{pred} : \{\mathsf{U}_0 \mid z : \mathbf{syn} \mapsto \mathsf{Tm}_m(z, \mathsf{El}(\mathsf{code}))\} \\ &\quad \mathsf{reflect} : \{\mathsf{Ne}_m(\mathsf{El}(\mathsf{code})) \to \mathsf{pred} \mid z : \mathbf{syn} \mapsto \mathsf{id}\} \\ &\quad \mathsf{reify} : \{\mathsf{pred} \to \mathsf{Nf}_m(\mathsf{El}(\mathsf{code})) \mid z : \mathbf{syn} \mapsto \mathsf{id}\} \end{aligned}$$

With Ψ in hand, we may define Uni^*:

$$\mathsf{Uni}^*.\mathsf{code} = \mathbf{Uni}$$

$$\mathsf{Uni}^*.\mathsf{pred} = \Psi$$

$$\mathsf{Uni}^*.\mathsf{reflect} = \lambda e.\ \langle \mathbf{up}(e); \mathsf{Ne}_m(\mathsf{El}(e)); \mathsf{id}; \lambda e.\ \mathbf{up}(e)\rangle$$

$$\mathsf{Uni}^*.\mathsf{reify} = \lambda A.\ A.\mathsf{code}$$

The definition of El^* is essentially cumulativity:

$$\mathsf{El}^*(\langle\mathsf{code}; \mathsf{pred}; \mathsf{reify}; \mathsf{reflect}\rangle) = \langle\mathsf{El}(\mathsf{code}); \mathsf{pred}; \mathsf{reify}; \mathsf{reflect}\rangle$$

It remains to show that $(\mathsf{Uni}^*, \mathsf{El}^*)$ is closed under various type formers. We show a representative cases: modal types. This concretely entails implementing the following constants:

$$\widehat{\mathsf{Mod}}^* : (\mu \mid A : \mathsf{Tm}_n^*(\mathsf{Uni}^*)) \to \{\mathsf{Tm}_m^*(\mathsf{Uni}^*) \mid z : \mathbf{syn} \mapsto \widehat{\mathsf{Mod}}(z, A)\}$$

$$\mathsf{dec}_{\widehat{\mathsf{Mod}}}^* : (\mu \mid A : \mathsf{Tm}_n^*(\mathsf{Uni}^*))$$

$$\to \{\mathsf{Tm}_m^*(\mathsf{El}^*(\widehat{\mathsf{Mod}}^*(A))) \cong \mathsf{Tm}_m^*(\mathsf{Mod}_\mu^*(\mathsf{El}^*(A))) \mid z : \mathbf{syn} \mapsto \mathsf{dec}_{\widehat{\mathsf{Mod}}}(z, A)\}$$

Fix $(\mu \mid A : \mathsf{Tm}_n^*(\mathsf{Uni}^*))$. We realign $\mathsf{Tm}_m^*(\mathsf{Mod}_\mu^*(\mathsf{El}^*(A)))$ along the isomorphism $\mathsf{dec}_{\widehat{\mathsf{Mod}}}$ to obtain a type Ψ and an isomorphism:

$$\mathsf{dec}_{\mathsf{Mod}_\mu}^* : \{\Psi \cong \mathsf{Tm}_m^*(\mathsf{Mod}_\mu^*(\mathsf{El}^*(A))) \mid z : \mathbf{syn} \mapsto \mathsf{dec}_{\widehat{\mathsf{Mod}}}(z, A)\}$$

It remains only to define $\widehat{\mathsf{Mod}}^*(A)$ such that $\widehat{\mathsf{Mod}}^*(A).\mathsf{pred} = \Psi$:

$$\widehat{\mathsf{Mod}}^*(A).\mathsf{code} = \langle\mu \mid \widehat{A.\mathsf{code}}\rangle$$

$$\widehat{\mathsf{Mod}}^*(A).\mathsf{pred} = \Psi$$

$$\widehat{\mathsf{Mod}}^*(A).\mathsf{reflect} = \lambda e.\ (\mathsf{dec}_{\widehat{\mathsf{Mod}}}^*)^{-1}(\uparrow_{\mathsf{Mod}_\mu^*(\mathsf{El}^*(A))} \mathsf{dec}^{\triangleright}(e))$$

$$\widehat{\mathsf{Mod}}^*(A).\mathsf{reify} = \lambda m.\ \mathsf{dec}^{\lhd}(\downarrow_{\mathsf{Mod}^*_\mu(\mathsf{El}^*(A))}\mathsf{dec}^*_{\widehat{\mathsf{Mod}}}(m))$$

The checks that all constructions lie over their syntactic counterparts follow immediately from the conclusions of realignment. □

Theorem 5.12. $\mathcal{G}$ *supports an MTT cosmos built around* $(\mathsf{Ty}^*_m, \mathsf{Tm}^*_m)$ *and* $\pi_0 : \mathcal{G} \longrightarrow \mathcal{S}$ *is a map of MTT cosmoi.*

6. The normalization algorithm

After Theorem 5.12, it remains only to parlay the existence of the normalization cosmos into a normalization function.

6.1. **The normalization function.** At this point, it becomes necessary to shift from working purely internally to $\mathcal{G}$ to inspecting some constructions externally. Accordingly, we will have use for the *total* spaces of terms and normal forms e.g. $\mathsf{Tm}^*_m = \sum_{A:\mathsf{Ty}^*_m} \mathsf{Tm}^*_m(A)$. We write $\mathcal{T}_m$ and $\mathcal{T}^{\bullet}_m$ for the presheaves of types and terms in $\mathcal{S}(m)$ to disambiguate them from Ty^*_m and Tm^*_m.

Lemma 6.1. *There is a morphism* $\downarrow : \mathsf{Tm}^*_m \longrightarrow \mathsf{Nf}_m$ *which restricts to* id *under* **syn**.

Proof. Working internally, $\downarrow(A, M) = (A, \downarrow_A M)$. □

Fix a term $\Gamma \vdash M : A @ m$. Theorems 3.9 and 5.12 define a map $[\![M]\!] : [\![\Gamma]\!] \longrightarrow \mathsf{Tm}^*_m$ in $\mathcal{G}(m)$ along with an isomorphism $\alpha : \pi_0([\![\Gamma]\!]) \cong \mathbf{y}(\Gamma)$ such that $\pi_0([\![M]\!]) = \lfloor M \rfloor \circ \alpha$.

We would like to obtain a normal form for M from $[\![M]\!]$. To this end, we can unfold $[\![M]\!]$ along with $\downarrow$ from Lemma 6.1 to obtain a commuting diagram:

$$
\begin{array}{ccc}
\pi_1([\![\Gamma]\!]) & \longrightarrow \pi_1(\mathsf{Tm}^*_m) \longrightarrow & \pi_1(\mathsf{Nf}_m) \\
\mathbf{i}[m]^*(\alpha) \circ [\![\Gamma]\!] \downarrow & \downarrow & \swarrow \\
\mathbf{i}[m]^*(\mathbf{y}(\Gamma)) & \xrightarrow[\mathbf{i}[m]^*(\lfloor M \rfloor)]{} \mathbf{i}[m]^*(\mathcal{T}^{\bullet}_m) &
\end{array}
$$

To normalize M, it suffices to construct $\mathsf{atoms}_\Gamma : \pi_1([\![\Gamma]\!])_\Gamma$ such that $\alpha([\![\Gamma]\!](\mathsf{atoms}_\Gamma)) = \mathsf{id} : \mathbf{i}[m]^*(\mathbf{y}(\Gamma))_\Gamma$: pushing atoms_Γ along the top of the diagram would yield a normal form (an element of $\pi_1(\mathsf{Nf}_m)$) which decodes to M by Yoneda. Modulo technical details, atoms_Γ is produced by using $\uparrow$ to convert variables for each element of Γ into elements of $\pi_1([\![\Gamma]\!])$.

Lemma 6.2. *For any* Γ $\mathsf{cx} @ m$ *there exists* $\mathsf{atoms}_\Gamma : (\mathbf{y}(\Gamma), \mathbf{y}(\Gamma)) \longrightarrow [\![\Gamma]\!]$ *in* $\mathcal{G}$ *lying over* $\mathsf{id} : \mathbf{i}[m]^*(\mathbf{y}(\Gamma))$ *in* $\mathcal{S}$.

Proof. This proof proceeds by induction on Γ.

Case: $\Gamma = \mathbf{1}$

Here $[\![\Gamma]\!]$ is terminal, so $\mathsf{atoms}_\mathbf{1}$ is its unique element. The requirement that $\mathsf{atoms}_\mathbf{1}$ lie over id is then tautological.

Case: $\Gamma = \Delta.(\mu \mid A)$

In this case, we note that $[\![\Gamma]\!] = [\![\Delta]\!] \times_{\mathcal{G}(\mu)(\mathsf{Ty}_n^*)} \mathcal{G}(\mu)(\mathsf{Tm}_n^*)$ and, since pullback are computed pointwise, it suffices to construct element of $\pi_1([\![\Delta]\!]_\Gamma)$ and $\pi_1(\mathcal{G}(\mu)(\mathsf{Tm}_n^*)_\Gamma)$ separately which agree on $\pi_1(\mathcal{G}(\mu)(\mathsf{Ty}_n^*)_\Gamma)$.

First, we reindex atoms_Δ by $\Gamma \vdash \uparrow : \Delta @ m$ to obtain $\delta \in \pi_1([\![\Delta]\!]_\Gamma)$. Next, using the element $\mathbf{v}_0 \in \mathcal{G}(\mu)(\mathsf{Ne}_n(A))_\Gamma$. It is easily seen that these agree on $\pi_1(\mathcal{G}(\mu)(\mathsf{Ty}_n^*)_\Gamma)$. The check that this lies over id follows from the fact that (1) δ lies over $\uparrow$, (2) $\uparrow_A \mathbf{v}_0$ lies over $\mathbf{v}_0$ and (3) that $\uparrow.\mathbf{v}_0 = \mathsf{id}$.

Case: $\Gamma = \Delta.\{\mu\}$

We define $\mathsf{atoms}_\Gamma = \mathcal{G}(\mu)_!(\mathsf{atoms}_\Delta)$. The check that this lies over id amounts to the equation in syntax that $\mathsf{id}.\{\mu\} = \mathsf{id}$. $\qquad\square$

Remark 6.3. atoms_Γ is analogous to the initial environment used in classical NbE proofs to kick off normalization. Abel [Abe13], for instance, denotes the environment $\uparrow^\Gamma$.

Combining Lemma 6.2 with the argument above, we conclude that for term $\Gamma \vdash M : A @ m$, there exists $\Gamma \vdash^{\mathsf{nf}} u : A @ m$ such that $|u| = M$. Moreover, because we have consistently worked with equivalences class of terms, this function automatically respects definitional equality. Summarizing:

Theorem 6.4. *There is a function* $\mathbf{nf}_\Gamma(-, A)$ *sending terms of type* $\Gamma \vdash A @ m$ *to normal forms such that*

(1) *If* $\Gamma \vdash M : A @ m$ *then* $\Gamma \vdash |\mathbf{nf}_\Gamma(M, A)| = M : A @ m$.
(2) *If* $\Gamma \vdash M = N : A @ m$ *then* $\mathbf{nf}_\Gamma(M, A) = \mathbf{nf}_\Gamma(N, A)$.

We can repeat this process to normalize types instead of terms. Given $\Gamma \vdash A @ m$, we obtain $[\![A]\!] : [\![\Gamma]\!] \longrightarrow \mathsf{Ty}_m^*$ which unfolds to an analogous diagram with only a small change: rather than using $\uparrow$ to pass from $\pi_1(\mathsf{Tm}_m^*)$ to normal forms, we use code to shift from Ty_m^* to normal types:

$$\begin{array}{ccc}
\pi_1([\![\Gamma]\!]) & \longrightarrow \pi_1(\mathsf{Ty}_m^*) \longrightarrow \pi_1(\mathsf{NfTy}_m) \\
\alpha \circ [\![\Gamma]\!] \downarrow \qquad & \qquad \downarrow \qquad \swarrow \\
\mathbf{i}[m]^*(\mathbf{y}(\Gamma)) & \xrightarrow[\mathbf{i}[m]^*(\lfloor A \rfloor)]{} \mathbf{i}[m]^*(\mathcal{T}_m)
\end{array}$$

By again pushing atoms_Γ along the top of this diagram, we obtain a normalization function for types.

Theorem 6.5. *There is a function* $\mathbf{nfty}_\Gamma(-)$ *sending types to normal types such that*

(1) *If* $\Gamma \vdash A @ m$ *then* $\Gamma \vdash |\mathbf{nfty}_\Gamma(A)| = A @ m$.
(2) *If* $\Gamma \vdash A = B @ m$ *then* $\mathbf{nfty}_\Gamma(A) = \mathbf{nfty}_\Gamma(B)$.

6.2. **Corollaries of normalization.** A number of important theorems follow as corollaries of Theorems 6.4 and 6.5. For instance, we can reduce the decidability of conversion to the decidability of normal forms.

Corollary 6.6.

(1) $\Gamma \vdash M = N : A @ m$ *iff* $\mathbf{nf}_\Gamma(M, A) = \mathbf{nf}_\Gamma(N, A)$.
(2) $\Gamma \vdash A = B @ m$ *iff* $\mathbf{nfty}_\Gamma(A) = \mathbf{nfty}_\Gamma(B)$.

Proof. We show only the proof for this first claim. The 'only if' direction is established by the second point of Theorem 6.4. Suppose instead $\mathbf{nf}_\Gamma(M, A) = \mathbf{nf}_\Gamma(N, A)$, so $|\mathbf{nf}_\Gamma(M, A)| = |\mathbf{nf}_\Gamma(N, A)|$. By the first point of Theorem 6.4, $|\mathbf{nf}_\Gamma(M, A)| = M$ and $|\mathbf{nf}_\Gamma(M, A)| = N$, so the conclusion follows. ◻

A priori, however, a given term could have multiple normal forms which complicates further analysis. We therefore strengthen Theorem 6.4 with the following:

Theorem 6.7 (Tightness)**.**
(1) *If $\Gamma \vdash^{\mathsf{nf}} u : A @ m$, then $\mathbf{nf}_\Gamma(|u|, A) = u$.*
(2) *If $\Gamma \vdash^{\mathsf{nf}} \tau @ m$, then $\mathbf{nfty}_\Gamma(|\tau|) = \tau$.*

Proof. Recall that Theorems 3.9 and 5.12 induce a function $[\![-]\!]$ sending a piece of syntax to its interpretation in the normalization model. Furthermore, recall the Γ-element $\mathsf{atoms}_\Gamma : [\![\Gamma]\!]$ constructed in Lemma 6.2.

We begin by strengthening the statement to make it more amenable to induction:

(1) If $\Gamma \vdash^{\mathsf{ne}} e : A @ m$, then $[\![|M|]\!](\mathsf{atoms}_\Gamma) = \uparrow_{[\![A]\!](\mathsf{atoms}_\Gamma)} e$
(2) If $\Gamma \vdash^{\mathsf{nf}} u : A @ m$, then $\downarrow_{[\![A]\!](\mathsf{atoms}_\Gamma)} [\![|u|]\!](\mathsf{atoms}_\Gamma) = u$.
(3) If $\Gamma \vdash^{\mathsf{nf}} \tau @ m$, then $[\![|A|]\!].\mathsf{code}(\mathsf{atoms}_\Gamma) = \tau$.

Here we have identified a code u (resp. e) as an Γ-element of Nf_A (resp. Ne_A). All three follow straightforwardly from mutual induction and the relevant definitions. For instance, if we consider $\Gamma \vdash^{\mathsf{nf}} (\mu \mid \tau) \to \sigma @ m$, we calculate as follows:

$$[\![|(\mu \mid \tau) \to \sigma|]\!].\mathsf{code}(\mathsf{atoms}_\Gamma)$$
$$= [\![(\mu \mid |\tau|) \to |\sigma|]\!].\mathsf{code}(\mathsf{atoms}_\Gamma)$$
$$= (\mu \mid [\![|\tau|]\!].\mathsf{code}(\mathsf{atoms}_\Gamma)) \to [\![|\sigma|]\!].\mathsf{code}(\uparrow^* \mathsf{atoms}_\Gamma, \uparrow \mathbf{v}_0)$$
$$= (\mu \mid [\![|\tau|]\!].\mathsf{code}(\mathsf{atoms}_\Gamma)) \to [\![|\sigma|]\!].\mathsf{code}(\mathsf{atoms}_{\Gamma.(\mu|A)})$$
$$= (\mu \mid \tau) \to \sigma$$

In order to carry out this calculation, we took advantage of not only the definition of dependent products in the gluing model, but also the interpretation of HOAS and atoms. ◻

Corollary 6.8. *Normalization is an isomorphism between equivalence classes of terms (resp. types) and normal forms (resp. normal types).*

Proof. Corollary 6.6 already shows that normalization is injective and Theorem 6.7 provides a section. ◻

These results imply the injectivity of type constructors, an essential property for implementation.

Corollary 6.9. *If $\Gamma \vdash A_0 \to B_0 = A_1 \to B_1 @ m$ then $\Gamma \vdash A_0 = A_1 @ m$ and $\Gamma.(\mathsf{id} \mid A_0) \vdash B_0 = B_1 @ m$.*

Proof. Set $\tau_i = \mathbf{nfty}_\Gamma(A_i)$ and $\sigma_i = \mathbf{nfty}_{\Gamma.(\mathsf{id}|A_0)}(B_i)$. Unfolding definitions shows that $|(\mu \mid \tau_i) \to \sigma_i| = |\tau_i| \to |\sigma_i| = A_i \to B_i$. By Corollary 6.8, $\mathbf{nfty}_\Gamma(A_i \to B_i) = (\mu \mid \tau_i) \to \sigma_i$.

Next, we recall that $\Gamma \vdash A_0 \to B_0 = A_1 \to B_1 @ m$ by assumption, so $(\mu \mid \tau_0) \to \sigma_0 = (\mu \mid \tau_1) \to \sigma_1$. As an operation on normal forms, however, $(\mu \mid -) \to -$ is clearly injective, so $\tau_0 = \tau_1$ and $\sigma_0 = \sigma_1$. The result now follows from Corollary 6.6. ◻

In light of Corollary 6.6, to decide the equality of terms and types, it suffices to argue that one may decide the equality of neutral and normal forms along with normal types. For this purpose, we adapt the bidirectional algorithm given by Altenkirch and Kaposi [AK17]. This argument goes through essentially without alteration, except that since certain constructors are annotated with 1- and 2-cells from $\mathcal{M}$, we require a decision procedure for objects in the mode theory. Note that this procedure uses e.g., Corollary 6.9, which is why we have delayed its statement till now.

Corollary 6.10. *If $\mathcal{M}$ is decidable, type checking is decidable.*

Finally, Gratzer et al. [GKNB20a] show canonicity for MTT extended with the equality $\mathbf{1}.\{\mu\} = \mathbf{1}$. Normalization provides a (heavy-handed) proof of canonicity without this equation by scrutinizing the definition of normal forms:

Corollary 6.11. *If $\mathbf{1}.\{\mu\} \vdash M : \mathsf{bool} @ m$ then $M \in \{\mathsf{tt}, \mathsf{ff}\}$.*

7. Extending MTT with crisp identity induction

To demonstrate the flexibility of the normalization argument given in Sections 5 and 6, we now show how it may be extended to accommodate modal principles not included in MTT.

Recall that, intuitively, a modality in MTT corresponds to a right adjoint. This intuition is supported by the fact that MTT modalities commute with products. In an extensional version of MTT, modalities also commute with (extensional) equality. That is, the following canonical map is an equivalence:

$$(\mu \mid x, y : A) \to \mathsf{Id}_{\langle \mu \mid A \rangle}(\mathsf{mod}_\mu(x), \mathsf{mod}_\mu(y)) \to \langle \mu \mid \mathsf{Id}_A(x, y) \rangle \qquad (7.1)$$

Remark 7.1. Constructing this map is slightly intricate. We begin by generalizing:

$$(x, y : \langle \mu \mid A \rangle) \to \mathsf{Id}_{\langle \mu \mid A \rangle}(x, y) \to \mathsf{let}_\nu \; \mathsf{mod}_\mu(x') \leftarrow x \; \mathsf{in} \; \mathsf{let}_\nu \; \mathsf{mod}_\mu(y') \leftarrow y \; \mathsf{in} \; \langle \mu \mid \mathsf{Id}_A(x', y') \rangle$$

In this form, we may use ordinary identity induction followed by modal induction to reduce to $(x : \langle \mu \mid A \rangle) \to \mathsf{let}_\nu \; \mathsf{mod}_\mu(x') \leftarrow x \; \mathsf{in} \; \mathsf{let}_\nu \; \mathsf{mod}_\mu(y') \leftarrow x \; \mathsf{in} \; \langle \mu \mid \mathsf{Id}_A(x', y') \rangle$ and then $(\mu \mid x : A) \to \langle \mu \mid \mathsf{Id}_A(x, x) \rangle$ respectively.

In *intensional* MTT, the same principle is not derivable.

Theorem 7.2. *There exists a model of intensional MTT with one mode m and one modality $\mu : m \longrightarrow m$ in which Equation 7.1 is not invertible.*

Proof. Consider intensional MTT and define an interpretation of MTT into intensional MLTT which interprets both modes as MLTT and sends all non-modal types to their counterparts within MLTT and interprets modal connectives as follows:

$$[\![\Gamma.\{\mu\}]\!] = [\![\Gamma]\!].\mathsf{Nat}$$
$$[\![\Gamma.(\mu \mid A)]\!] = [\![\Gamma]\!].(\mathsf{Nat} \to [\![A]\!])$$
$$[\![\Gamma.\{\mathsf{id}\}]\!] = [\![\Gamma]\!]$$
$$[\![\Gamma.(\mathsf{id} \mid A)]\!] = [\![\Gamma]\!].[\![A]\!]$$
$$[\![\langle \mu \mid A \rangle]\!] = \mathsf{Nat} \to [\![A]\!]$$
$$[\![\langle \mathsf{id} \mid A \rangle]\!] = [\![A]\!]$$
$$[\![\mathsf{mod}_\mu(M)]\!] = \lambda([\![M]\!])$$

$$\llbracket \mathsf{mod}_{\mathsf{id}}(M) \rrbracket = \llbracket M \rrbracket$$

$$\llbracket \mathsf{let}_\chi \; \mathsf{mod}_\xi(\underline{\;\;}) \leftarrow M \text{ in } N \rrbracket = \llbracket N \rrbracket [\mathsf{id}.\llbracket M \rrbracket]$$

Unfolding the interpretation of Equation 7.1, we observe that an inverse to this map corresponds to function extensionality for functions $\mathsf{Nat} \to A$. As function extensionality is independent of MLTT, there must be no inverse to Equation 7.1 definable within MTT. $\square$

In light of Theorem 7.2, we refer to the existence of an inverse to Equation 7.1 as *modal extensionality*. Modal extensionality is useful in practice. In incarnations of guarded recursion within MTT, for instance, some version of modal extensionality is required to prove any equalities involving guarded types [GKNB21, GB22]. It is therefore worth investigating whether modal extensionality is compatible with both normalization and canonicity.[7]

In work by Shulman [Shu18] and Gratzer [GKNB21], *crisp* induction principles are a variation of the induction principles for types such as bool or $\mathsf{Id}_A(a_0, a_1)$ which allow the scrutinee of the induction to occur beneath a modality. Crisp induction principles are derivable in MTT if the modality has an internal right adjoint [GKNB21], but they are justified in other situations. In particular, crisp induction for identity types is validated if and only if modal extensionality holds. In contrast to modal extensionality, however, it is straightforward to directly adapt the proofs of normalization and canonicity to account for crisp identity induction principles:

$$\frac{\begin{array}{c} \Gamma.(\mu \mid A).(\mu \mid A[\uparrow]).(\mu \mid \mathsf{Id}_{A[\uparrow^2]}(\mathbf{v}_1, \mathbf{v}_0)) \vdash B @ m \\ \Gamma.(\mu \mid A) \vdash M : B[\uparrow.\mathbf{v}_0.\mathbf{v}_0.\mathsf{refl}(\mathbf{v}_0)] @ m \\ \Gamma.\{\mu\} \vdash N_0, N_1 : A @ n \qquad \Gamma.\{\mu\} \vdash P : \mathsf{Id}_A(N_0, N_1) @ n \end{array}}{\Gamma \vdash \mathsf{J}^\mu(B, M, P) : B[\mathsf{id}.N_0.N_1.P] @ m}$$

$$\mathsf{J}^\mu(B, M, \mathsf{refl}(N)) = M[\mathsf{id}.N]$$

The modularity of our proof of normalization ensures that only local changes to the construction of identity types in $\mathcal{G}$ are needed to adapt the entire proof to support crisp induction. Concretely, two changes to primitive constants added to MSTC by Section 5.1. One alteration to the definition of cosmoi and one to the definition of neutral forms:

$$\mathsf{J}_\mu : (\mu \mid A : \mathsf{Ty}_n)\,(B : (\mu \mid a_0, a_1 : \mathsf{Tm}_n(A))(\mu \mid p : \mathsf{Tm}_n(\mathsf{Id}(A, a_0, a_1))) \to \mathsf{Ty}_m)$$
$$\qquad \to ((\mu \mid a : \mathsf{Tm}_n(A)) \to \mathsf{Tm}_m(B(a, a, \mathsf{refl}(a))))$$
$$\qquad \to (\mu \mid a_0, a_1 : \mathsf{Tm}_n(A))(\mu \mid p : \mathsf{Tm}_n(\mathsf{Id}(A, a_0, a_1)))$$
$$\qquad \to \mathsf{Tm}_m(B(a_0, a_1, p))$$
$$\mathbf{J}_\mu : (\mu \mid A : \bigcirc\mathsf{Ty}_n)\,(B : (\mu \mid a_0, a_1 : \mathsf{V}_n(A))(\mu \mid p : \mathsf{V}_m(\mathsf{Id}(A, a_0, a_1))) \to \mathsf{NfTy}_m)$$
$$\qquad \to ((\mu \mid a : \mathsf{V}_n(A)) \to \mathsf{Nf}_m(B(a, a, \mathsf{refl}(a))))$$
$$\qquad \to (\mu \mid a_0, a_1 : \bigcirc_z\mathsf{Tm}_n(z, A(z)))(\mu \mid p : \mathsf{Ne}_n(\mathsf{Id}(A, a_0, a_1)))$$
$$\qquad \to \mathsf{Ne}_m(B(a_0, a_1, \eta(p)))$$

These changes simply reflect the change to the elimination principle of the identity type.

After having made this change, only one portion of Section 5.2 must change: Lemma 5.10 which shows that the gluing cosmos is closed under identity types. We must show that $(\mathsf{Ty}_m^*, \mathsf{Tm}_m^*)$ is closed under crisp induction.

---

[7]Like function extensionality, it is straightforward to maintain *either* normalization or canonicity in the presence of modal extensionality. Ensuring for both simultaneously is far more difficult.

Lemma 7.3. $(\mathsf{Ty}_m^*, \mathsf{Tm}_m^*)$ *supports crisp identity induction.*

Proof. This argument is similar to Lemma 5.7, as the induction principle for modal types is always 'crisp' in MTT. We must implement the following constant.

$$\mathsf{J}_\mu^* : (\mu \mid A : \mathsf{Ty}_n^*) \, (B : (\mu \mid a_0, a_1 : \mathsf{Tm}_n^*(A))(\mu \mid p : \mathsf{Tm}_n^*(\mathsf{Id}^*(A, a_0, a_1))) \to \mathsf{Ty}_m^*)$$
$$\to (b : (\mu \mid a : \mathsf{Tm}_n^*(A)) \to \mathsf{Tm}_m^*(B(a, a, \mathsf{refl}^*(a))))$$
$$\to (\mu \mid a_0, a_1 : \mathsf{Tm}_n^*(A))(\mu \mid p : \mathsf{Tm}_n^*(\mathsf{Id}(A, a_0, a_1))) \to$$
$$\to \{\mathsf{Tm}_m^*(B(a_0, a_1, p)) \mid z : \mathbf{syn} \mapsto \mathsf{J}_\mu(A, B, b, p)\}$$

Let us fix A, B, b, a_0, a_1, and p with the types described above. Recalling the definition of $\mathsf{Id}^*(A, a_0, a_1).\mathsf{pred}$ from Lemma 5.10, we can commute $\langle \mu \mid - \rangle$ past the dependent sum, closed modalities, equality types, and coproducts to decompose p into a pair of the following:

$$(\mu \mid \mathsf{tm} : \mathsf{Nf}_n(\mathsf{Id}(A, a_0, a_1)))$$

$$\mathsf{prf} : \bullet \begin{bmatrix} (\sum_{e:\langle \mu \mid \mathsf{Ne}_n(\mathsf{Id}(A,a_0,a_1))\rangle} \mathbf{up} \circledast e = \mathsf{mod}_\mu(\mathsf{tm})) \\ + \, (\mathsf{mod}_\mu(a_0) = \mathsf{mod}_\mu(a_1) \times \mathsf{mod}_\mu(\mathsf{tm}) = \mathsf{mod}_\mu(\mathsf{refl}(a_0))) \end{bmatrix}$$

We then define $\mathsf{J}_\mu^*(B, b, a_0, a_1, p)$ by analyzing prf:

$$\begin{cases} \mathsf{J}(z, B, b, a_0, a_1, p) & \mathsf{prf} = \iota_1(z) \\ {\downarrow}\mathbf{J}(\lambda a_0, a_1, p. \; B({\uparrow}a_0, {\uparrow}a_1, {\uparrow}p).\mathsf{code}, \lambda a. \; {\downarrow}b({\uparrow}a), e) & q = \iota_2(\iota_1(e, \_)) \\ b(a_0) & q = \iota_2(\iota_2(\_, \_)) \quad \square \end{cases}$$

Having made this alteration, the remainder of Sections 5 and 6 are unchanged. In particular, all the results of Section 6 continue to hold in the presence of crisp induction.

8. Related work

We have built on top of a long line of research systematically structuring logical relations as gluing models [MS93, AHS95, Str98, Fio02, Shu15, AK17, KHS19, Coq19, SA21, Ste21]. In particular, Altenkirch et al. [AHS95] and Fiore [Fio02] recast NbE into the construction of a gluing model in which types are triples $(A, \downarrow, \uparrow)$. Generalizing from this work to dependent type theory has proven a considerable challenge [AK16]. The final ingredient for Martin-Löf type theory was provided by Coquand [Coq19]: a construction of a universe in this gluing model similar to that of Shulman [Shu15].

Gluing for modal type theory. Gratzer et al. [GSB19a] gave a classical normalization-by-evaluation proof for a Fitch-style type theory. The complexity of this proof, however, makes it intractable to extend to a general modal type theory like MTT. Unfortunately, extending gluing techniques to modal type theories has proven challenging. In particular, Gratzer et al. [GKNB20a] used gluing to prove canonicity for MTT, but they were forced to add an additional equality to MTT ($\mathbf{1}.\{\mu\} = \mathbf{1}$) to tame the construction of the gluing model. The challenge lies in fitting the glued category of contexts into a CwF-style model of type theory; the natural definition of glued types and terms fails to admit modalities. While there have been some attempts to systematize the construction of glued CwFs [KHS19], they do not apply to MTT.

Recently, Hu and Pientka [HP22] gave a proof of normalization for a simply-typed Fitch-style type theory (Kripke-style in their parlance) with one modality. They give

two separate proofs of normalization; one through both an untyped PER model similar to Gratzer et al. [GSB19a] and one using a gluing model. Their gluing proof is closely related to the argument above. For instance, their theory of unified substitutions and modal transformations corresponds to a specialization of MTT's substitution calculus to one modality and, accordingly, their category of renamings offers a strict presentation of the category of renamings described above. Their proof, however, is done using external constructions on the gluing category which may make it difficult to scale to either multiple modalities or dependent types.

Synthetic Tait computability. The introduction of representable map categories [Uem19] and LCCCs [GS20] for modeling the syntax of (non-modal) type theory offered an alternative approach. Crucially, they show that syntax can be given a universal property among structured categories with better behavior than CwFs. Sterling and collaborators [SH21, SA21, Ste21] have built on this idea and introduced synthetic Tait computability to prove syntactic metatheorems via gluing together LCCCs rather than CwFs. Unlike other approaches to gluing, STC generalizes well to a multimodal setting and by extending STC to MSTC normalization for MTT becomes tractable.

MTT as a metalanguage. In a parallel line of work, Bocquet et al. [BKS21] have also used MTT as a metalanguage in the construction of models of type theory. They, however, do not work with a modal object type theory and instead use MTT to internalize a functor F rather than working internally to $\mathbf{Gl}(F)$. As a result, while both proofs use MTT modalities, the modalities used by op. cit. are encoded in our proof by fibered lex monads $(\bigcirc, \bullet)$ which prove easier to manipulate.

9. Conclusions and future work

We prove normalization for MTT (Theorem 6.4) and thereby reduce the decidability of conversion and type checking to the decidability of equality of the underlying mode theory (Corollaries 6.6 and 6.10). In addition, we deduce a number of corollaries from normalization itself, including the injectivity of type constructors and canonicity (Corollaries 6.9 and 6.11).

By working constructively, we have obtained an effective procedure for normalization. This, along with our results on type checking, open the door to a theoretically-sound implementation of MTT generic in the mode theory. In the future, we intend to develop a bidirectional syntax for MTT and implement it. Stassen et al. [SGB22] have made promising initial steps in this direction for *poset-enriched* mode theories.

References

[Abe13] Andreas Abel. *Normalization by Evaluation: Dependent Types and Impredicativity.* Habilitation, 2013.

[AGV72] Michael Artin, Alexander Grothendieck, and Jean-Louis Verdier. *Théorie des topos et cohomologie étale des schémas.* 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.

[AHS95] Thorsten Altenkirch, Martin Hofmann, and Thomas Streicher. Categorical reconstruction of a reduction free normalization proof. In David Pitt, David E. Rydeheard, and Peter Johnstone, editors, *Category Theory and Computer Science*, pages 182–199. Springer Berlin Heidelberg, 1995.

[AK16] Thorsten Altenkirch and Ambrus Kaposi. Normalisation by Evaluation for Dependent Types. In Delia Kesner and Brigitte Pientka, editors, *1st International Conference on Formal Structures for Computation and Deduction (FSCD 2016)*, volume 52 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 6:1–6:16, Dagstuhl, Germany, 2016. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. URL: `http://drops.dagstuhl.de/opus/volltexte/2016/5972`, `doi:10.4230/LIPIcs.FSCD.2016.6`.

[AK17] Thorsten Altenkirch and Ambrus Kaposi. Normalisation by evaluation for type theory, in type theory. *Logical Methods in Computer Science*, Volume 13, Issue 4, 10 2017. `doi:10.23638/LMCS-13(4:1)2017`.

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

[Awo18] Steve Awodey. Natural models of homotopy type theory. *Mathematical Structures in Computer Science*, 28(2):241–286, 2018. `arXiv:1406.3219`, `doi:10.1017/S0960129516000268`.

[BBC+19] Lars Birkedal, Aleš Bizjak, Ranald Clouston, Hans Bugge Grathwohl, Bas Spitters, and Andrea Vezzosi. Guarded cubical type theory. *Journal of Automated Reasoning*, (63):211–253, 2019.

[BCM+20] Lars Birkedal, Ranald Clouston, Bassel Mannaa, Rasmus Ejlers Møgelberg, Andrew M. Pitts, and Bas Spitters. Modal dependent type theory and dependent right adjoints. *Mathematical Structures in Computer Science*, 30(2):118–138, 2020. `arXiv:1804.05236`, `doi:10.1017/S0960129519000197`.

[BKS21] Rafaël Bocquet, Ambrus Kaposi, and Christian Sattler. Induction principles for type theories, internally to presheaf categories, 2021. `arXiv:2102.11649`.

[CJ95] Aurelio Carboni and Peter Johnstone. Connected limits, familial representability and Artin glueing. *Mathematical Structures in Computer Science*, 5(4):441–459, 1995. `doi:10.1017/S0960129500001183`.

[Clo18] Ranald Clouston. Fitch-Style Modal Lambda Calculi. In Christel Baier and Ugo Dal Lago, editors, *Foundations of Software Science and Computation Structures*, pages 258–275. Springer International Publishing, 2018.

[Coq19] Thierry Coquand. Canonicity and normalization for dependent type theory. *Theoretical Computer Science*, 777:184–191, 2019. `doi:10.1016/j.tcs.2019.01.015`.

[Fio02] Marcelo Fiore. Semantic analysis of normalisation by evaluation for typed lambda calculus. In *Proceedings of the 4th ACM SIGPLAN International Conference on Principles and Practice of Declarative Programming*, PPDP '02, pages 26–37. ACM, 2002. `doi:10.1145/571157.571161`.

[GB22] Daniel Gratzer and Lars Birkedal. A Stratified Approach to Löb Induction. In Amy P. Felty, editor, *7th International Conference on Formal Structures for Computation and Deduction (FSCD 2022)*, volume 228 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 23:1–23:22, Dagstuhl, Germany, 2022. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. URL: `https://drops.dagstuhl.de/opus/volltexte/2022/16304`, `doi:10.4230/LIPIcs.FSCD.2022.23`.

[GCK+22] Daniel Gratzer, Evan Cavallo, G. A. Kavvos, Adrien Guatto, and Lars Birkedal. Modalities and parametric adjoints. *ACM Trans. Comput. Logic*, 23(3), 04 2022. `doi:10.1145/3514241`.

[GKNB20a] Daniel Gratzer, G.A. Kavvos, Andreas Nuyts, and Lars Birkedal. Multimodal dependent type theory. In *Proceedings of the 35th Annual ACM/IEEE Symposium on Logic in Computer Science*, LICS '20. ACM, 2020. `doi:10.1145/3373718.3394736`.

[GKNB20b] Daniel Gratzer, G.A. Kavvos, Andreas Nuyts, and Lars Birkedal. Type theory à la mode, 2020. Technical Report for the LICS paper "Multimodal Dependent Type Theory". URL: `https://jozefg.github.io/papers/type-theory-a-la-mode.pdf`.

[GKNB21]   Daniel Gratzer, G. A. Kavvos, Andreas Nuyts, and Lars Birkedal. Multimodal Dependent Type Theory. *Logical Methods in Computer Science*, Volume 17, Issue 3, July 2021. URL: `https://lmcs.episciences.org/7713`, `doi:10.46298/lmcs-17(3:11)2021`.

[GS20]   Daniel Gratzer and Jonathan Sterling. Syntactic categories for dependent type theory: sketching and adequacy, 2020. `arXiv:2012.10783`.

[GSB19a]   Daniel Gratzer, Jonathan Sterling, and Lars Birkedal. Implementing a Modal Dependent Type Theory. *Proc. ACM Program. Lang.*, 3, 2019. `doi:10.1145/3341711`.

[GSB19b]   Daniel Gratzer, Jonathan Sterling, and Lars Birkedal. Normalization-by-evaluation for modal dependent type theory, 2019. Technical Report for the ICFP paper by the same name. URL: `https://jozefg.github.io/papers/2019-implementing-modal-dependent-type-theory-tech-report.pdf`.

[GSS22]   Daniel Gratzer, Michael Shulman, and Jonathan Sterling. Strict universes for Grothendieck topoi, 2022. URL: `https://arxiv.org/abs/2202.12012`.

[Hof97]   Martin Hofmann. Syntax and Semantics of Dependent Types. In Andrew M. Pitts and P. Dybjer, editors, *Semantics and Logics of Computation*, pages 79–130. Cambridge University Press, 1997. URL: `https://www.tcs.ifi.lmu.de/mitarbeiter/martin-hofmann/pdfs/syntaxandsemanticsof-dependenttypes.pdf`, `doi:10.1017/CBO9780511526619.004`.

[Hof99]   Martin Hofmann. Semantical analysis of higher-order abstract syntax. In *Proceedings of the 14th Annual IEEE Symposium on Logic in Computer Science*, LICS '99, pages 204–. IEEE Computer Society, 1999. URL: `http://dl.acm.org/citation.cfm?id=788021.788940`.

[HP22]   Jason Z.S. Hu and Brigitte Pientka. A categorical normalization proof for the modal lambda-calculus. volume Proceedings of the 38th International Conference on Mathematical Foundations of Programming Semantics (MFPS'22), 2022.

[HS97]   Martin Hofmann and Thomas Streicher. Lifting Grothendieck universes. Unpublished note, 1997. URL: `https://www2.mathematik.tu-darmstadt.de/~streicher/NOTES/lift.pdf`.

[JY20]   Niles Johnson and Donald Yau. 2-dimensional categories, 2020. `arXiv:2002.06055`.

[KHS19]   Ambrus Kaposi, Simon Huber, and Christian Sattler. Gluing for type theory. In Herman Geuvers, editor, *Proceedings of the 4th International Conference on Formal Structures for Computation and Deduction (FSCD 2019)*, volume 131, 2019.

[KKA19]   Ambrus Kaposi, András Kovács, and Thorsten Altenkirch. Constructing quotient inductive-inductive types. *Proc. ACM Program. Lang.*, 3(POPL):2:1–2:24, January 2019. `doi:10.1145/3290315`.

[KPT99]   Yoshiki Kinoshita, John Power, and Makoto Takeyama. Sketches. *Journal of Pure and Applied Algebra*, 143(1):275–291, 1999. `doi:10.1016/S0022-4049(98)00114-5`.

[LSR17]   Daniel R. Licata, Michael Shulman, and Mitchell Riley. A Fibrational Framework for Substructural and Modal Logics. In Dale Miller, editor, *2nd International Conference on Formal Structures for Computation and Deduction (FSCD 2017)*, volume 84 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 25:1–25:22. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2017. `doi:10.4230/LIPIcs.FSCD.2017.25`.

[ML92]   Per Martin-Löf. Substitution calculus, 1992. Notes from a lecture given in Göteborg.

[MP08]   Conor McBride and Ross Paterson. Applicative programming with effects. *Journal of Functional Programming*, 18(1), 2008. URL: `http://www.staff.city.ac.uk/~ross/papers/Applicative.pdf`, `doi:10.1017/S0956796807006326`.

[MS93]   John C. Mitchell and Andre Scedrov. Notes on sconing and relators. In E. Börger, G. Jäger, H. Kleine Büning, S. Martini, and M. M. Richter, editors, *Computer Science Logic*, pages 352–378. Springer Berlin Heidelberg, 1993. `doi:10.1007/3-540-56992-8_21`.

[OP18]   Ian Orton and Andrew M. Pitts. Axioms for Modelling Cubical Type Theory in a Topos. *Logical Methods in Computer Science*, 14(4), 2018. `arXiv:1712.04864`, `doi:10.23638/LMCS-14(4:23)2018`.

[Red20]   The RedPRL Development Team. `cooltt`, 2020. URL: `http://www.github.com/RedPRL/cooltt`.

[RSS20]   Egbert Rijke, Michael Shulman, and Bas Spitters. Modalities in homotopy type theory. *Logical Methods in Computer Science*, 16(1), 2020. `arXiv:1706.07526`.

[SA21]   Jonathan Sterling and Carlo Angiuli. Normalization for cubical type theory. In *Proceedings of the 36th Annual ACM/IEEE Symposium on Logic in Computer Science*, LICS '21, New York, NY, USA, 2021. ACM.

[SGB22]   Philipp Stassen, Daniel Gratzer, and Lars Birkedal. A flexible multimodal proof assistant. In *Workshop on the Implementation of Type Systems*, 2022.

[SH21]   Jonathan Sterling and Robert Harper. Logical relations as types: Proof-relevant parametricity for program modules. 68(6), 2021. `arXiv:2010.08599`, `doi:10.1145/3474834`.

[SH22]   Jonathan Sterling and Robert Harper. Sheaf semantics of termination-insensitive noninterference. In Amy P. Felty, editor, *7th International Conference on Formal Structures for Computation and Deduction (FSCD 2022)*, volume 228 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 5:1–5:19, Dagstuhl, Germany, August 2022. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. `arXiv:2204.09421`, `doi:10.4230/LIPIcs.FSCD.2022.5`.

[Shu15]   Michael Shulman. Univalence for inverse diagrams and homotopy canonicity. *Mathematical Structures in Computer Science*, 25(5):1203–1277, 2015. `arXiv:1203.3253`, `doi:10.1017/S0960129514000565`.

[Shu18]   Michael Shulman. Brouwer's fixed-point theorem in real-cohesive homotopy type theory. *Mathematical Structures in Computer Science*, 28(6):856–941, 2018. `doi:10.1017/S0960129517000147`.

[Ste21]   Jonathan Sterling. *First Steps in Synthetic Tait Computability: The Objective Metatheory of Cubical Type Theory*. PhD thesis, 2021. CMU technical report CMU-CS-21-142. `doi:10.5281/zenodo.5709838`.

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

[Tai67]   W. W. Tait. Intensional Interpretations of Functionals of Finite Type I. *Journal of Symbolic Logic*, 32(2):198–212, 1967. `doi:10.2307/2271658`.

[Uem19]   Taichi Uemura. A general framework for the semantics of type theory. 04 2019. URL: `https://arxiv.org/abs/1904.04097`, `arXiv:1904.04097`.

[WCPW04]   Kevin Watkins, Iliano Cervesato, Frank Pfenning, and David Walker. A concurrent logical framework: The propositional fragment. In Stefano Berardi, Mario Coppo, and Ferruccio Damiani, editors, *Types for Proofs and Programs*, pages 355–377, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg. `doi:10.1007/978-3-540-24849-1_23`.