Normalization for Multimodal Type Theory

Daniel Gratzer
gratzer@cs.au.dk
Aarhus University
Aarhus, Denmark

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 follows from 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.

CCS Concepts: • **Theory of computation** → **Type theory**; **Modal and temporal logics**; **Categorical semantics**.

Keywords: type theory, modal type theory, normalization, modalities, categorical semantics, Artin gluing

ACM Reference Format:
Daniel Gratzer. 2022. Normalization for Multimodal Type Theory. In *37th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS) (LICS '22), August 2–5, 2022, Haifa, Israel.* ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3531130.3532398

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 [15, 26]. These theories can be instantiated by a description of a modal situation to produce a system enjoying the theorems usually laboriously proved by experts.

1.1 Multimodal type theory

We focus on one such general modal type theory: MTT [15]. 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) [8] 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 [16, Theorem 10.4], for instance, exemplifies this and hinges on many such calculations. In fact, the metatheoretic results established by Gratzer et al. [15] (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 and only if conversion is decidable in the mode theory. As corollaries, we show that type constructors in MTT are always injective and that type checking is decidable when the mode theory is decidable.[1]

1.2 Normalization for type theories

A normalization algorithm must begin by defining *normal forms*. Their precise formulation varies 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 [1].

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)* [1]. Proving that an NbE algorithm works, however, is an extremely intricate affair involving a variety of complex constructions. Recent work by Gratzer et al. [18] 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 [19].

1.2.1 Normalization-by-gluing. These difficulties are not unique to modal type theories, and a long line of research focuses on taming the complexity of NbE through *gluing* [3, 4, 12, 13, 35, 38]. This line of work recasts normalization algorithms as the construction of models of type theory in categories defined by Artin gluing. Objects in this model are proof-relevant logical relations: proof-relevant predicates on syntax taken up to definitional equality.

To prove normalization, types in the gluing model are supplemented with *reify* and *reflect* maps reminiscent of those from classical NbE which e.g., send a witness of a predicate on some term to a normal form of that term. This structure, together with the initiality of syntax, allows us to extract a normalization algorithm from this model.

Normalization-by-gluing proofs force a number of design decisions which are not inherently categorical, but vital to simplifying the proof. Working with syntax up to definitional equality, for instance, may appear a minor point but it is essential: connectives in type theory only have universal properties when considered up to definitional equality. Only when working with equivalences classes therefore, can we use these universal properties and benefit from existing results. For instance, the local cartesian closure of the gluing category can be used to automatically close the model under dependent products. In a similar vein, proof-relevance is essential for a natural interpretation of the universe, where witnesses encode the data of a type in the model. In a proof-irrelevant setting, one must laboriously work around the inability to encode this data indirectly [2].

[1]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.

1.2.2 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 at may first appear. Models of type theory are subject to a variety of strict equations which often force external constructions, where naturality obligations can be prohibitive. Worse, the passage between between mathematics internal to the gluing category and external constructions is difficult, and the boundary frequently raises subtle mismatches.

We follow Sterling and Harper [37] and adopt a synthetic approach to gluing. By ensuring that the gluing category is a (presheaf) topos we obtain a rich internal language for manipulating logical relations synthetically. By further extending this language with a pair of lex monads and the strictification axiom stated by Orton and Pitts [29], we can construct the normalization model completely internally.

Sterling and collaborators have termed this approach *synthetic Tait computability (STC)* and shown that working internally simplifies constructions involved in the gluing model, making it practical to prove normalization for even extremely complex type theories like cubical type theory [35, 36].

1.2.3 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*.

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.

1.3 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. [15]. 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 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 3.8). 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 4.13). Finally, in Section 5 we construct the normalization cosmos (Theorem 5.7) and extract the normalization function in Section 6 (Theorem 6.3). Section 7 extends this proof to support crisp induction.

For reasons of space, we have deferred many details to the accompanying technical report [14].

Metatheory. We work in a constructive metatheory such as IZF equipped with a hierarchy of Grothendieck universes.

2 A primer on MTT

We collect the key ideas of MTT [16]. 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 maps between modalities. Henceforth, we will work with MTT over a fixed mode theory $\mathcal{M}$. Second, we formally view MTT as a particular generalized algebraic theory (GAT). Accordingly, binding is handled by de Bruijn indices and the theory uses explicit substitutions [27]. We refer the reader to Gratzer et al. [16, Section 4] for a discussion of the motivations and consequences behind this choice.

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., $\Gamma\ \mathsf{cx} @ m$, $\Gamma \vdash A @ m$, $\Gamma \vdash M : A @ m$. 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, dependent products, 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 [30].

2.2 Modalities in MTT

MTT draws inspiration from Fitch-style type theories [8, 11] with each modality defined 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 $\langle \mu \mid A \rangle$:

$$\frac{\Gamma\ \mathsf{cx} @ m}{\Gamma.\{\mu\}\ \mathsf{cx} @ n} \qquad \frac{\Gamma.\{\mu\} \vdash A @ n}{\Gamma \vdash \langle \mu \mid A \rangle @ m}$$

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

These context operations assemble into a 2-functor $m \mapsto \mathsf{Cx}_m$ from $\mathcal{M}^{\mathsf{coop}}$[2] to the category of contexts. 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 $\Gamma.\{\mu\}.\{\xi\} = \Gamma.\{\mu \circ \xi\}\ \mathsf{cx} @ o$.

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:

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

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 $\Delta.\{\mu\} \vdash M : A[\gamma'.\{\mu\}] @ n$.

Despite this similarity, 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 context extension rule and normal variable rules are special cases of this principle, obtained by setting $\mu = \mathsf{id}$.

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

Remark 1. From the view of Fitch-style type theories, where $-.\{\mu\}$ is a 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 $\langle \nu \mid A \rangle$ with annotation μ or directly hypothesize over A with annotation $\mu \circ \nu$. There is a substitution navigating in one direction, but not the other:

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

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

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

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

Modal types organize into a "pseudo-functor" with $\langle \mathsf{id} \mid A \rangle \simeq A$ and $\langle \mu \mid \langle \nu \mid A \rangle \rangle \simeq \langle \mu \circ \nu \mid A \rangle$. They also satisfy axiom K:

$$(\circledast) : \langle \mu \mid A \to B \rangle \to \langle \mu \mid A \rangle \to \langle \mu \mid B \rangle$$

Remark 2. In fact, dependent products in MTT are *modalized* so that $A \to B$ is replaced by $(\mu \mid A) \to B$:

$$\frac{\Gamma.(\mu \mid A) \vdash M : B \mathbin{@} m}{\Gamma \vdash \lambda(M) : (\mu \mid A) \to B \mathbin{@} m}$$

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

This feature will be a useful convenience when using MTT as a metalanguage, but for space reasons we will only prove normalization for non-modal dependent products.

2.3 Normal and neutral forms in MTT

As mentioned in Section 1.2, the starting point for normalizations 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 [41]. Accordingly, a normal form (resp. neutral, normal type) is typed by the judgment $\Gamma \Vdash^{\mathsf{nf}} u : A \mathbin{@} m$ (resp. $\Gamma \Vdash^{\mathsf{ne}} e : A \mathbin{@} m$, $\Gamma \Vdash^{\mathsf{nf}} \tau \mathbin{@} m$) 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:

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

For reasons of space, we omit a full description of neutral and normal forms. The main deviation from the familiar case of Martin-Löf type theory is the neutral form for variables: for MTT these are annotated with a 2-cell and index, decoding to $\mathbf{v}_0$ together with a combination of weakening and 2-cell substitutions $\uparrow$ and $\{\alpha\}$.

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 and only if equality of modalities and 2-cells is decidable.

Renamings. While normal and neutral forms are not stable under substitution, they are stable under the restricted class of *renamings*. While we omit the full definition, they are intuitively 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 2.1. *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. [16] 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 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. [16, Section 5] then repackages the definition of models in the language of natural models [6].

3.1 Natural models of MTT

We begin by recalling the presentation of a model of MTT given by Gratzer et al. [16]. Recall that a natural model of type theory [6] is a pair of a category 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 $f : X \longrightarrow Y : \mathbf{PSh}(C)$ 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 $\mathbf{y}(C) \longrightarrow Y$.

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 $F : \mathcal{M}^{\text{coop}} \longrightarrow \mathbf{Cat}$. 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 $F : \mathcal{M}^{\text{coop}} \longrightarrow \mathbf{Cat}$ 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 products:

$$\begin{tikzcd}
\sum_{A:\mathcal{T}_m} \sum_{B:\tau_m[A]\to\mathcal{T}_m} \prod_{a:\tau_m[A]} \tau_m[B(a)] \arrow[r] \arrow[d] \arrow[dr, phantom, "\lrcorner", very near start] & \mathcal{T}_m^\bullet \arrow[d] \\
\sum_{A:\mathcal{T}_m} \prod_{\_:\tau_m[A]} \mathcal{T}_m \arrow[r] & \mathcal{T}_m
\end{tikzcd} \qquad (1)$$

Diagram 1 takes advantage of the model of extensional MLTT in a presheaf topos [20]. 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 [22]. We refer to a family of presheaves as *small* if it is classified by a universe.

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

$$\begin{tikzcd}
F(\mu)^*\mathcal{T}_n^\bullet \arrow[r] \arrow[d] & \mathcal{T}_m^\bullet \arrow[d] \\
F(\mu)^*\mathcal{T}_n \arrow[r] & \mathcal{T}_m
\end{tikzcd} \qquad (2)$$

Unlike dependent 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. However, this principle is rather complex and, since we will soon be in a position to present a much simpler account of modal elimination, we refer the reader to Gratzer et al. [16, Section 5] for the full definition.

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. [16, Section 5] 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 3.3. *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 1 and 2.

Definition 3.4. 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 $F_!(\mu) \dashv F(\mu)$.

Example 3. 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 3.3.

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. For instance, we model the formation and introduction rules of modal type by requiring the following diagram:

$$\begin{tikzcd} F(\mu)(\mathcal{T}_n^\bullet) \arrow[r] \arrow[d] & \mathcal{T}_m^\bullet \arrow[d] \\ F(\mu)(\mathcal{T}_n) \arrow[r] & \mathcal{T}_m \end{tikzcd} \tag{3}$$

Definition 3.5. An MTT cosmos F is a cosmos equipped with the choice of morphism $\tau_m : \mathcal{T}_m^\bullet \longrightarrow \mathcal{T}_m : F(m)$ for each $m : \mathcal{M}$ along with choices of diagrams à la Diagram 3 to close each τ_m under dependent products, sums, booleans, identity types, a weak universe, and modal types.

Definition 3.6. A morphism of cosmoi $\alpha : F \longrightarrow G$ is a 2-natural transformation such that each naturality square satisfies Beck-Chevalley and each α_m is an LCC functor.

Definition 3.7. A morphism of MTT cosmoi $\alpha : F \longrightarrow G$ is a morphism of cosmoi such that α_m strictly preserves τ_m and all connectives.

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 proof 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.[3] Accordingly, we cannot easily obtain a section of a map into $\mathcal{S}$ and in fact sections rarely exist. Any such map, however, is surjective on definable terms and this 'quasi-projectivity' is sufficient:

Theorem 3.8. *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 $\alpha_\Gamma : \pi([\![\Gamma]\!]) \cong \mathbf{y}(\Gamma)$.*
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.

Remark 4. Both Theorem 3.3 and Theorem 3.8 are categorical abstractions of *rule induction*. Indeed, Theorem 3.3 is used to prove Theorem 3.8—via the construction of an appropriate displayed model [24]—and the latter takes the place of rule induction in the proof of normalization (see Theorem 6.3).

[3] 2-monad theory [17, 25] yields an initial cosmos $\mathcal{I}$ but we work with $\mathcal{S}$ because—unlike $\mathcal{I}$—it is known to adequately represent syntax.

$$\begin{aligned}
&\mathsf{Prod} : (A : \mathsf{Ty}_m)(B : \mathsf{Tm}_m(A) \to \mathsf{Ty}_m) \to \mathsf{Ty}_m \\
&\alpha_{\mathsf{Prod}} : (A : \mathsf{Ty}_m)(B : \mathsf{Tm}_m(A) \to \mathsf{Ty}_m) \\
&\quad \to \mathsf{Tm}_m(\mathsf{Prod}(A,B)) \cong \left[\textstyle\prod_{a:\mathsf{Tm}_m(A)} \mathsf{Tm}_m(B(a))\right] \\
&\mathsf{Mod}_\mu : (\mu \mid \mathsf{Ty}_n) \to \mathsf{Ty}_m \\
&\mathsf{m}_\mu : (\mu \mid A : \mathsf{Ty}_n)(\mu \mid \mathsf{Tm}_n(A)) \to \mathsf{Tm}_m(\mathsf{Mod}_\mu(A)) \\
&\mathsf{letmod}_{\mu;\nu} : (\nu \circ \mu \mid A : \mathsf{Ty}_n) \\
&\quad (B : (\mu \mid \mathsf{Tm}_n(\mathsf{Mod}_\mu(A))) \to \mathsf{Ty}_o) \\
&\quad (b : (\nu \circ \mu \mid x : \mathsf{Tm}_n(A)) \to \mathsf{Tm}_o(B(\mathsf{m}_\mu(A,x)))) \\
&\quad (\nu \mid a : \mathsf{Tm}_m(\mathsf{Mod}_\mu(A))) \\
&\quad \to \mathsf{Tm}_o(B(a)) \\
&\mathsf{Mod/beta}_{\mu;\nu} : (\cdots) \to \mathsf{letmod}_{\mu;\nu}(A,B,b,\mathsf{m}_\mu(A,a)) = b(a)
\end{aligned}$$

Figure 1. Dependent products and modal types, internally

3.3 Presheaf cosmoi

Example 3 shows that each model of MTT induces an MTT cosmos. In fact, such cosmoi are particularly well-behaved as they are comprised over presheaf topoi connected by adjoint triples. These cosmoi enjoy a privileged role in our proof and we observe some of their unique behavior.

Definition 3.9. A presheaf cosmos F is a cosmos where 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. [16] 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.

Some caution is required here, as a presheaf cosmos will frequently host more than one interpretation of MTT. A presheaf cosmos is always equipped with this modal metalanguage (extensional MTT) which can then be used to specify a model of (intensional) MTT. This is comparable to Diagram 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 small types $\vdash \mathsf{Ty}_m : \mathsf{U}_0$ @ m and $A : \mathsf{Ty}_m \vdash \mathsf{Tm}_m(A) : \mathsf{U}_0$ @ m. Translating Diagram 3 into this language yields the following constants:

$$\begin{aligned}
&\mathsf{Mod}_\mu : (\mu \mid \mathsf{Ty}_n) \to \mathsf{Ty}_m \\
&\quad \mathsf{m}_\mu : (\mu \mid A : \mathsf{Ty}_n)(\mu \mid \mathsf{Tm}_n(A)) \to \mathsf{Tm}_m(\mathsf{Mod}_\mu(A))
\end{aligned}$$

Figure 1 contains the complete specification of modal types—including modal elimination—and dependent products. The remaining connectives follow a similar pattern.

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.

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 $(E : \mathcal{E}, F : \mathcal{F}, F \longrightarrow \rho(E))$ while morphisms are commuting squares:

$$\begin{tikzcd}
F_0 \arrow[r, "f_0"] \arrow[d] & F_1 \arrow[d] \\
\rho(E_0) \arrow[r, "\rho(f_1)"'] & \rho(E_1)
\end{tikzcd}$$

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

Example 5. 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 [39], 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 4.2 (Artin et al. [5], Carboni and Johnstone [10])**.** *$\mathbf{Gl}(\rho)$ is a presheaf topos and π_0 is a LCC 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 [37]. 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 4.3. 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 $[\mathbf{syn} \times A, B]_{\mathbf{Gl}(\rho)} \cong [\pi_0(A), \pi(B)]_{\mathcal{E}}$. Internally, hypothesizing $\mathbf{syn}$ collapses the category to $\mathcal{E}$:

Lemma 4.4. *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 $\bigcirc A = \mathbf{syn} \to A$ [31]. This modality has a strongly disjoint lex idempotent modality, $\bullet A$ [31, Section 3.4]. While we could work with $\bullet$ entirely through this characterization, it is helpful to fix a definition:

$$\begin{tikzcd}
\mathbf{syn} \times A \arrow[r] \arrow[d] & A \arrow[d] \\
\mathbf{syn} \arrow[r] \arrow[ur, phantom, very near end, "\ulcorner"] & \bullet A
\end{tikzcd} \tag{4}$$

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* [7, 29, 37], stating that the following canonical map has an inverse re for any $B : \mathsf{U}$:

$$\left(\textstyle\sum_{A:\mathsf{U}} [A \cong B]\right) \to \left(\textstyle\sum_{A:\mathbf{syn}\to\mathsf{U}} \prod_{z:\mathbf{syn}} A(z) \cong B\right) \tag{5}$$

Unfolding these conditions yields the following:

Definition 4.5. 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:

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

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 3.6 where a priori two constants might agree only up to isomorphism.

Orton and Pitts [29, Theorem 8.4] show that a Hofmann–Streicher universe satisfies realignment for levelwise decidable propositions. Using the presentation of $\mathbf{Gl}(\rho)$ as a presheaf topos [10], $\mathbf{syn}$ is clearly levelwise decidable and so realignment at $\mathbf{syn}$ is constructively valid.

Definition 4.6. 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 **syn** : Ω such that each universe satisfies the realignment axiom for **syn**.

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

Theorem 4.7. *$\mathbf{Gl}(\rho)$ 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 [12] 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 continuous functors between them:

$$\begin{tikzcd}
\mathcal{E}_n \arrow[r, "\rho_n"] \arrow[d, "f"'] & \mathcal{F}_n \arrow[d, "g"] \\
\mathcal{E}_m \arrow[r, "\rho_m"'] & \mathcal{F}_m
\end{tikzcd} \tag{6}$$

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 4.2 and 4.7 both are presheaf topoi and models of STC. Artin gluing is functorial, and Diagram 6 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 4.8. *$\mathbf{Gl}(f, g) : \mathbf{Gl}(\rho_n) \longrightarrow \mathbf{Gl}(\rho_m)$ 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 $\beta \circ g_!(f) : g_!(F) \longrightarrow \rho_n(f_!(E))$. □

Lemma 4.9. *The adjunction $\mathbf{Gl}(f, g)_! \dashv \mathbf{Gl}(f, g)$ induces a dependent right adjoint.*

As a consequence of Lemma 4.9, 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 [8].

Lemma 4.10. *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 $\mathbf{Gl}(f, g)(\mathbf{syn}_n) \cong (\mathbf{1}, \mathbf{0}, !) = \mathbf{syn}_m$. □

In fact, in this model $\bigcirc\langle \mu \mid A \rangle \cong \langle \mu \mid \bigcirc A \rangle$ and $\bullet\langle \mu \mid A \rangle \cong \langle \mu \mid \bullet A \rangle$. These isomorphisms are not automatic consequences of Lemma 4.10 but both follow from similar calculations.

Remark 6. Technically, **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 4.11. 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 **syn**.
- Each universe satisfies the realignment axiom for **syn**.
- MTT modalities commute with **syn**, $\bigcirc$, and $\bullet$.

Summarizing the preceding discussion:

Theorem 4.12. *$\mathbf{Gl}(\rho_n)$, $\mathbf{Gl}(\rho_m)$, 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 4.13. *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 2.3 the 2-functor of categories of renamings Ren_-. By an identical construction to Example 3, 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 4.13 then yields the following:

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

As a further consequence of Theorem 4.13, 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 5.2 (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 5.3 (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 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 4.4 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 Fig. 1 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 5.4. 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 $\mathsf{Nf}_m(\lambda z.\ \mathsf{Prod}(z, A(z), B(z)))$.

The normals and neutrals themselves lift to constants of type $\mathsf{Nf}_m(A)$, $\mathsf{Ne}_m(A)$, and NfTy_m using a form of higher-order abstract syntax [21]. For example, the constants for dependent products and modal types are presented in Fig. 2. These operations collapse to the corresponding syntactic constants specified by Extension 1 under $z : \mathbf{syn}$—recall from Extension 2 that here e.g. $\mathsf{Nf}_m(A) = \mathsf{Tm}_m(z, A)$.

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 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 $z : \mathbf{syn}$.

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

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

$$\begin{aligned}\mathbf{lam} : {} & (\mu \mid A : \bigcirc\mathsf{Ty}_m)\,(B : (\mu \mid \bigcirc\mathsf{Tm}_m(A)) \to \bigcirc\mathsf{Ty}_m) \\ & \to ((a : \mathsf{V}_m(A)) \to \mathsf{Nf}_m(B(a))) \\ & \to \mathsf{Nf}_m(\mathsf{Prod}(A, B))\end{aligned}$$

$$\begin{aligned}\mathbf{app} : {} & (A : \bigcirc\mathsf{Ty}_m)\,(B : (\bigcirc\mathsf{Tm}_m(A)) \to \bigcirc\mathsf{Ty}_m) \\ & \to \mathsf{Ne}_m(\mathsf{Prod}(A, B)) \to (a : \mathsf{Nf}_m(A)) \to \mathsf{Ne}_m(B(a))\end{aligned}$$

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

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

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

Figure 2. Neutral and normal forms, internally

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 $m : \mathcal{M}$:

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

We start with the following putative definition of types:

$$\begin{aligned}&\mathbf{record}\ T : \mathsf{U}_2\ \mathbf{where} \\ &\quad \mathsf{code} : \mathsf{NfTy}_m \\ &\quad \mathsf{pred} : \{\mathsf{U}_1 \mid z : \mathbf{syn} \mapsto \mathsf{Tm}_m(z, \mathsf{code})\} \\ &\quad \mathsf{reflect} : \{\mathsf{Ne}_m(\mathsf{code}) \to \mathsf{pred} \mid \mathbf{syn} \mapsto \mathsf{id}\} \\ &\quad \mathsf{reify} : \{\mathsf{pred} \to \mathsf{Nf}_m(\mathsf{code}) \mid \mathbf{syn} \mapsto \mathsf{id}\}\end{aligned} \tag{7}$$

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

$$\alpha_\bigcirc(z, A) = A.\mathsf{code} : \textstyle\prod_{z:\mathbf{syn}} T \cong \mathsf{Ty}^*_m(z)$$

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 4.5 applies and we can define

$$(\mathsf{Ty}^*_m, \alpha) = \mathsf{re}(T, \mathsf{Ty}_m, \alpha_\bigcirc) \tag{8}$$

The equation $z : \mathbf{syn} \vdash \mathsf{Ty}^*_m = \mathsf{Ty}_m(z)$ follows immediately from the second half of Definition 4.5. 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 $\mathsf{Tm}^*_m(A)$:

$$\mathsf{Tm}^*_m(A) = A.\mathsf{pred} : \{\mathsf{U}_1 \mid z : \mathbf{syn} \mapsto \mathsf{Tm}^*_m(z, A)\} \tag{9}$$

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.code in Construction 7 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. For mode-local connectives, these constructions are identical to those given in Sterling [35]. Accordingly, we detail only dependent products as a representative mode-local connective along with modal types.

Lemma 5.5. *$(\mathsf{Ty}^*_m, \mathsf{Tm}^*_m)$ is closed under dependent products.*

Proof. We must define two constants:

$$\mathsf{Prod}^* : (A : \mathsf{Ty}^*_m)(B : \mathsf{Tm}^*_m(A) \to \mathsf{Ty}^*_m) \to \mathsf{Ty}^*_m$$
$$\alpha_{\mathsf{Prod}^*} : (A : \mathsf{Ty}^*_m)(B : \mathsf{Tm}^*_m(A) \to \mathsf{Ty}^*_m)$$
$$\to \mathsf{Tm}^*_m(\mathsf{Prod}^*(A, B)) \cong \left[\textstyle\prod_{a:\mathsf{Tm}^*_m(A)} \mathsf{Tm}^*_m(B(a))\right]$$

Additionally, we must show that if $z : \mathbf{syn}$ then $\mathsf{Prod}^* = \mathsf{Prod}(z)$ and $\alpha_{\mathsf{Prod}^*} = \alpha_{\mathsf{Prod}}(z)$.

We begin by fixing $A : \mathsf{Ty}^*_m$ and $B : \mathsf{Tm}^*_m(A) \to \mathsf{Ty}^*_m$. Define $\Phi = \prod_{a:\mathsf{Tm}^*_m(A)} \to \mathsf{Tm}^*_m(B(a))$ and observe under $z : \mathbf{syn}$ that $\Phi = \prod_{a:\mathsf{Tm}_m(z,A)} \to \mathsf{Tm}_m(B(z, a))$ and therefore

$$\alpha_{\mathsf{Prod}}(z) : \mathsf{Tm}_m(z, \mathsf{Prod}(z, A, B)) \cong \Phi$$

We now realign Φ along this isomorphism to obtain 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^*}:

$$\mathsf{Prod}^*(A, B).\mathsf{code} = \mathbf{Prod}(A.\mathsf{code}, \lambda v.\ B(\downarrow_A v).\mathsf{code})$$
$$\mathsf{Prod}^*(A, B).\mathsf{pred} = \Psi$$
$$\mathsf{Prod}^*(A, B).\mathsf{reflect} = \lambda e.\ \beta^{-1}(\lambda a.\ \mathbf{app}(e, \downarrow_A a))$$
$$\mathsf{Prod}^*(A, B).\mathsf{reify} = \lambda f.\ \mathbf{lam}(\lambda v.\ \uparrow_{B(\uparrow_A v)} \beta(f)(\uparrow_A v))$$
$$\alpha_{\mathsf{Prod}^*} = \beta$$

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 reflect and reify become the identity. These follow directly from assumptions about A, B, and the boundaries of various constructors. For instance

$$\begin{aligned}\mathsf{Prod}^*(A, B) &= \mathsf{Prod}^*(A, B).\mathsf{code}\\ &= \mathbf{Prod}(A.\mathsf{code}, \lambda v.\ B(\downarrow_A v).\mathsf{code})\\ &= \mathsf{Prod}(z, A.\mathsf{code}, \lambda v.\ B(\downarrow_A v).\mathsf{code})\\ &= \mathsf{Prod}(z, A, \lambda v.\ B(\downarrow_A v))\\ &= \mathsf{Prod}(z, A, B)\end{aligned}$$ □

Lemma 5.6. *$(\mathsf{Ty}^*_m, \mathsf{Tm}^*_m)$ is closed under modal types.*

Proof. Fix a modality $\mu : n \longrightarrow m$. In this case we define the four constants specified by Fig. 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:

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

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

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

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:

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

We must construct an element of $\mathsf{Tm}^*_o(B(a))$. We begin by inspecting m. As MTT modalities in extensional MTT commute with dependent sums, equality, $\bullet$, and—by Extension 4—with finite coproducts, m 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 that $\bullet X$ is a pushout of **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 : **syn**. We conclude by scrutinizing prf:

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

Given z : **syn**, both branches collapse to $\mathsf{letmod}_{\mu;\nu}(z, A, B, b, a)$ so this yields a well-defined map. We omit the routine computations of boundary conditions for space reasons. □

Theorem 5.7. *$\mathcal{G}$ 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 Normalization

After Theorem 5.7, 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 $\Gamma \vdash M : A \mathbin{@} m$. Theorems 3.8 and 5.7 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:

$$\begin{tikzcd}
\pi_1([\![\Gamma]\!]) \arrow[r] \arrow[d, "\alpha \circ [\![\Gamma]\!]"'] & \pi_1(\mathsf{Tm}^*_m) \arrow[r] \arrow[d] & \pi_1(\mathsf{Nf}_m) \arrow[dl] \\
\mathbf{i}[m]^*(\mathbf{y}(\Gamma)) \arrow[r, "\mathbf{i}[m]^*(\lfloor M \rfloor)"'] & \mathbf{i}[m]^*(\mathcal{T}^\bullet_m) &
\end{tikzcd}$$

To normalize M, it suffice 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.

Lemma 6.2. *For any Γ cx $@\, m$ there exists $\mathsf{atoms}_\Gamma : \pi_1([\![\Gamma]\!])_\Gamma$ lying over* $\mathsf{id} : \mathbf{i}[m]^*(\mathbf{y}(\Gamma))$.

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.

Case. $\Gamma = \Delta.(\mu \mid A)$
In this case $[\![\Gamma]\!] = [\![\Delta]\!] \times_{\mathcal{G}(\mu)(\mathsf{Ty}^*_n)} \mathcal{G}(\mu)(\mathsf{Tm}^*_n)$. First, we reindex atoms_Δ by $\Gamma \vdash \uparrow : \Delta \mathbin{@} m$ to obtain $\delta \in [\![\Delta]\!]_\Gamma$. Next, using the element $\mathbf{v}_0 \in \mathcal{G}(\mu)(\mathsf{Ne}_n(A))_\Gamma$ we define $\mathsf{atoms}_\Gamma = (\delta, \uparrow_A \mathbf{v}_0)$.

Case. $\Gamma = \Delta.\{\mu\}$
We define $\mathsf{atoms}_\Gamma = \mathcal{G}(\mu)_!(\mathsf{atoms}_\Delta)$ □

Remark 7. atoms_Γ is analogous to the initial environment used in classical NbE proofs to kick off normalization. Abel [1], for instance, denotes the environment $\uparrow^\Gamma$.

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

Theorem 6.3. *There is a function $\mathbf{nf}_\Gamma(-, A)$ sending terms of type $\Gamma \vdash A \mathbin{@} m$ to normal forms such that*

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

We can repeat this process to normalize types instead of terms. Given $\Gamma \vdash A \mathbin{@} m$, 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:

$$\begin{tikzcd}
\pi_1([\![\Gamma]\!]) \arrow[r] \arrow[d, "\alpha \circ [\![\Gamma]\!]"'] & \pi_1(\mathsf{Ty}^*_m) \arrow[r] \arrow[d] & \pi_1(\mathsf{NfTy}_m) \arrow[dl] \\
\mathbf{i}[m]^*(\mathbf{y}(\Gamma)) \arrow[r, "\mathbf{i}[m]^*(\lfloor A \rfloor)"'] & \mathbf{i}[m]^*(\mathcal{T}_m) &
\end{tikzcd}$$

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

Theorem 6.4. *There is a function $\mathbf{nfty}_\Gamma(-)$ sending types to normal types such that*

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

6.2 Corollaries of normalization

A number of important theorems follow as corollaries of Theorems 6.3 and 6.4. For instance, we can reduce the decidability of conversion to the decidability of the mode theory.

Corollary 6.5 (Decidability of conversion)**.**

1. *$\Gamma \vdash M = N : A \mathbin{@} m$ iff $\mathbf{nf}_\Gamma(M, A) = \mathbf{nf}_\Gamma(N, A)$.*
2. *$\Gamma \vdash A = B \mathbin{@} m$ 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 6.3. 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 6.3, $|\mathbf{nf}_\Gamma(M, A)| = M$ and $|\mathbf{nf}_\Gamma(M, A)| = N$, so the conclusion follows. □

Equality of normal forms and normal types is evidently decidable if equality in $\mathcal{M}$ is decidable, so this proves the promised sharp bound on the decidability of conversion in MTT. While we have not developed a bidirectional syntax for MTT, the fully annotated presentation of its syntax is decidable precisely when conversion is decidable:

Corollary 6.6. *If $\mathcal{M}$ is decidable, type checking is decidable.*

A priori, however, a given term could have multiple normal forms which complicates further analysis. We therefore strengthen Theorem 6.3 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 $\mathbf{nfty}_\Gamma(|\tau|) = \tau$.*

This result is proven by a straightforward inductive argument over neutrals, normal forms, and normal types and by inspecting the relevant portions of the normalization cosmos in each case.

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

Proof. Corollary 6.5 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 $|\tau_i \to \sigma_i| = |\tau_i| \to |\sigma_i| = A_i \to B_i$. By Corollary 6.8, $\mathbf{nfty}_\Gamma(A_i \to B_i) = \tau_i \to \sigma_i$.

Next, we recall that $\Gamma \vdash A_0 \to B_0 = A_1 \to B_1 @ m$ by assumption, so $\tau_0 \to \sigma_0 = \tau_1 \to \sigma_1$. As an operation on normal forms, however, $- \to -$ is clearly injective, so $\tau_0 = \tau_1$ and $\sigma_0 = \sigma_1$. The result now follows from Corollary 6.5. □

Finally, Gratzer et al. [15] show canonicity for MTT extended with the equality $\mathbf{1}.\{\mu\} = \mathbf{1}$. Normalization provides a (heavy-handed) proof of canonicity without this equation:

Corollary 6.10. *If $\mathbf{1}.\{\mu\} \vdash M : \mathsf{bool} @ m$ then $M \in \{\mathsf{tt}, \mathsf{ff}\}$.*

7 Crisp identity induction principles

In Shulman [33] and Gratzer et al. [16], *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 [16], but they are justified in other situations. In particular, crisp induction for identity types is validated if and only if the canonical map $\mathsf{Id}_{\langle\mu|A\rangle}(\mathsf{mod}_\mu(M_0), \mathsf{mod}_\mu(M_1)) \to \langle \mu \mid \mathsf{Id}_A(M_0, M_1)\rangle$ is an equivalence:

$$\mathsf{Id}_{\langle\mu|A\rangle}(\mathsf{mod}_\mu(M_0), \mathsf{mod}_\mu(M_1)) \simeq \langle \mu \mid \mathsf{Id}_A(M_0, M_1)\rangle \quad (10)$$

This is true if e.g., $\mathsf{Id}_A(a_0, a_1)$ supports equality reflection.

This equivalence is often indispensable when reasoning about modal operations in MTT; it plays essentially the same role as function extensionality. Accordingly, it is frequently useful to add the following strengthened version of the induction principle for identity types which implies Eq. (10):

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

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

Given, however, that Eq. (10) is analogous to function extensionality, it is unclear that MTT continues to enjoy canonicity and normalization under this extension.

The modularity of our proof of normalization ensures, however, that only local changes to the construction of identity types in $\mathcal{G}$ are needed to adapt the entire proof to support crisp induction. We conclude that all the results of Section 6 continue to hold after the addition of this induction scheme.

8 Related work

We have built on top of a long line of research systematically structuring logical relations as gluing models [3, 12, 13, 23, 28, 32, 35, 36, 38]. In particular, Altenkirch et al. [3] and Fiore [13] 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 [4]. The final ingredient for Martin-Löf type theory was provided by Coquand [12]: a construction of a universe in this gluing model similar to that of Shulman [32].

Gluing for modal type theory. Gratzer et al. [18] 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. [15] 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 [23], they do not apply to MTT.

Synthetic Tait computability. The introduction of representable map categories [40] and LCCCs [17] 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 [35–37] 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. [9] 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 6.3) and thereby reduce the decidability of conversion and type checking to the decidability of equality of the underlying mode theory (Corollaries 6.5 and 6.6). In addition, we deduce a number of corollaries from normalization itself, including the injectivity of type constructors and canonicity (Corollaries 6.9 and 6.10).

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. [34] have made promising initial steps in this direction for *poset-enriched* mode theories.

Acknowledgments

We are thankful for discussions with Carlo Angiuli, Martin Bidlingmaier, Lars Birkedal, Thierry Coquand, Alex Kavvos, Christian Sattler, and Jonathan Sterling. The author was supported in part by a Villum Investigator grant (no. 25804), Center for Basic Research in Program Verification (CPV), from the VILLUM Foundation.

References

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

[2] Stuart Frazier Allen. 1987. *A non-type-theoretic semantics for type-theoretic language.* Ph.D. Dissertation. Cornell University.

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

[4] Thorsten Altenkirch and Ambrus Kaposi. 2016. Normalisation by Evaluation for Dependent Types. In *1st International Conference on Formal Structures for Computation and Deduction (FSCD 2016) (Leibniz International Proceedings in Informatics (LIPIcs), Vol. 52)*, Delia Kesner and Brigitte Pientka (Eds.). Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany, 6:1–6:16. https://doi.org/10.4230/LIPIcs.FSCD.2016.6

[5] Michael Artin, Alexander Grothendieck, and Jean-Louis Verdier. 1972. *Théorie des topos et cohomologie étale des schémas.* Springer-Verlag. 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.

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

[7] Lars Birkedal, Aleš Bizjak, Ranald Clouston, Hans Bugge Grathwohl, Bas Spitters, and Andrea Vezzosi. 2019. Guarded Cubical Type Theory. *Journal of Automated Reasoning* 63 (2019), 211–253.

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

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

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

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

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

[13] Marcelo Fiore. 2002. 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* (Pittsburgh, PA, USA) *(PPDP '02)*. ACM, 26–37. https://doi.org/10.1145/571157.571161

[14] Daniel Gratzer. 2021. Normalization for multimodal type theory. arXiv:2106.01414 [cs.LO]

[15] Daniel Gratzer, G.A. Kavvos, Andreas Nuyts, and Lars Birkedal. 2020. Multimodal Dependent Type Theory. In *Proceedings of the 35th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS '20)*. ACM. https://doi.org/10.1145/3373718.3394736

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

[17] Daniel Gratzer and Jonathan Sterling. 2020. Syntactic categories for dependent type theory: sketching and adequacy. arXiv:2012.10783 [cs.LO]

[18] Daniel Gratzer, Jonathan Sterling, and Lars Birkedal. 2019. Implementing a Modal Dependent Type Theory. *Proc. ACM Program. Lang.* 3 (2019). Issue ICFP. https://doi.org/10.1145/3341711

[19] Daniel Gratzer, Jonathan Sterling, and Lars Birkedal. 2019. Normalization-by-Evaluation for Modal Dependent Type Theory. https://jozefg.github.io/papers/2019-implementing-modal-dependent-type-theory-tech-report.pdf Technical Report for the ICFP paper by the same name.

[20] Martin Hofmann. 1997. Syntax and Semantics of Dependent Types. In *Semantics and Logics of Computation*, Andrew M. Pitts and P. Dybjer (Eds.). Cambridge University Press, 79–130. https://doi.org/10.1017/CBO9780511526619.004

[21] Martin Hofmann. 1999. Semantical Analysis of Higher-Order Abstract Syntax. In *Proceedings of the 14th Annual IEEE Symposium on Logic in Computer Science* (Washington, DC, USA) *(LICS '99)*. IEEE Computer Society, 204–. http://dl.acm.org/citation.cfm?id=788021.788940

[22] Martin Hofmann and Thomas Streicher. 1997. Lifting Grothendieck Universes. (1997). https://www2.mathematik.tu-darmstadt.de/

~streicher/NOTES/lift.pdf Unpublished note.
[23] Ambrus Kaposi, Simon Huber, and Christian Sattler. 2019. Gluing for type theory. In *Proceedings of the 4th International Conference on Formal Structures for Computation and Deduction (FSCD 2019)*, Herman Geuvers (Ed.), Vol. 131.
[24] Ambrus Kaposi, András Kovács, and Thorsten Altenkirch. 2019. Constructing Quotient Inductive-inductive Types. *Proc. ACM Program. Lang.* 3, POPL, Article 2 (Jan. 2019), 24 pages. https://doi.org/10.1145/3290315
[25] Yoshiki Kinoshita, John Power, and Makoto Takeyama. 1999. Sketches. *Journal of Pure and Applied Algebra* 143, 1 (1999), 275–291. https://doi.org/10.1016/S0022-4049(98)00114-5
[26] Daniel R. Licata, Michael Shulman, and Mitchell Riley. 2017. A Fibrational Framework for Substructural and Modal Logics. In *2nd International Conference on Formal Structures for Computation and Deduction (FSCD 2017) (Leibniz International Proceedings in Informatics (LIPIcs), Vol. 84)*, Dale Miller (Ed.). Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 25:1–25:22. https://doi.org/10.4230/LIPIcs.FSCD.2017.25
[27] Per Martin-Löf. 1992. Substitution calculus. Notes from a lecture given in Göteborg.
[28] John C. Mitchell and Andre Scedrov. 1993. Notes on sconing and relators. In *Computer Science Logic*, E. Börger, G. Jäger, H. Kleine Büning, S. Martini, and M. M. Richter (Eds.). Springer Berlin Heidelberg, 352–378. https://doi.org/10.1007/3-540-56992-8_21
[29] Ian Orton and Andrew M. Pitts. 2018. Axioms for Modelling Cubical Type Theory in a Topos. *Logical Methods in Computer Science* 14, 4 (2018). https://doi.org/10.23638/LMCS-14(4:23)2018 arXiv:1712.04864
[30] The RedPRL Development Team. 2020. `cooltt`. http://www.github.com/RedPRL/cooltt
[31] Egbert Rijke, Michael Shulman, and Bas Spitters. 2020. Modalities in homotopy type theory. *Logical Methods in Computer Science* 16, 1 (2020). arXiv:1706.07526
[32] Michael Shulman. 2015. Univalence for inverse diagrams and homotopy canonicity. *Mathematical Structures in Computer Science* 25, 5 (2015), 1203–1277. https://doi.org/10.1017/S0960129514000565 arXiv:1203.3253
[33] Michael Shulman. 2018. Brouwer's fixed-point theorem in real-cohesive homotopy type theory. *Mathematical Structures in Computer Science* 28, 6 (2018), 856–941. https://doi.org/10.1017/S0960129517000147
[34] Philipp Stassen, Daniel Gratzer, and Lars Birkedal. 2022. A flexible multimodal proof assistant. In *Workshop on the Implementation of Type Systems*.
[35] Jonathan Sterling. 2021. *First Steps in Synthetic Tait Computability: The Objective Metatheory of Cubical Type Theory.* Ph.D. Dissertation. https://doi.org/10.5281/zenodo.5709838 CMU technical report CMU-CS-21-142.
[36] Jonathan Sterling and Carlo Angiuli. 2021. Normalization for Cubical Type Theory. In *Proceedings of the 36th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS '21)*. ACM, New York, NY, USA.
[37] Jonathan Sterling and Robert Harper. 2021. Logical Relations as Types: Proof-Relevant Parametricity for Program Modules. 68, 6 (2021). https://doi.org/10.1145/3474834 arXiv:2010.08599 [cs.PL]
[38] Thomas Streicher. 1998. Categorical intuitions underlying semantic normalisation proofs. In *Preliminary Proceedings of the APPSEM Workshop on Normalisation by Evaluation*, O. Danvy and P. Dybjer (Eds.). Department of Computer Science, Aarhus University.
[39] W. W. Tait. 1967. Intensional Interpretations of Functionals of Finite Type I. *Journal of Symbolic Logic* 32, 2 (1967), 198–212. https://doi.org/10.2307/2271658
[40] Taichi Uemura. 2019. A General Framework for the Semantics of Type Theory. (04 2019). arXiv:1904.04097 [math.CT] https://arxiv.org/abs/1904.04097
[41] Kevin Watkins, Iliano Cervesato, Frank Pfenning, and David Walker. 2004. A Concurrent Logical Framework: The Propositional Fragment. In *Types for Proofs and Programs*, Stefano Berardi, Mario Coppo, and Ferruccio Damiani (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 355–377. https://doi.org/10.1007/978-3-540-24849-1_23