Transpension: The Right Adjoint to the Pi-type

Andreas Nuyts
imec-DistriNet, KU Leuven, Belgium
andreas.nuyts@cs.kuleuven.be

Dominique Devriese
Vrije Universiteit Brussel, Belgium
dominique.devriese@vub.be

Abstract

Presheaf models of dependent type theory have been successfully applied to model HoTT, parametricity, and directed and guarded type theory. There has been considerable interest in internalizing aspects of these presheaf models, either to make the resulting language more expressive, or in order to carry out further reasoning internally, allowing greater abstraction and sometimes automated verification. While the constructions of presheaf models largely follow a common pattern, approaches towards internalization do not. Throughout the literature, various internal presheaf operators ($\surd$, Φ/extent, Ψ/Gel, Glue, Weld, mill and the strictness axiom) can be found and little is known about their relative expressivenes. Moreover, some of these require that variables whose type is a shape (representable presheaf) be used affinely.

We propose a novel type former, the transpension type, which is right adjoint to universal quantification over a shape. Its structure resembles a dependent version of the suspension type in HoTT. We give general typing rules and a presheaf semantics in terms of base category functors dubbed multipliers. Structural rules for shape variables and certain aspects of the transpension type depend on characteristics of the multiplier. We demonstrate how the transpension type and the strictness axiom can be combined to implement all and improve some of the aforementioned internalization operators.

2012 ACM Subject Classification Theory of computation → Type theory

Keywords and phrases

1 Introduction and Related Work

1.1 The Power of Presheaves

Presheaf semantics [18, 19] are an excellent tool for modelling relational preservation properties of (dependent) type theory. They have been applied to parametricity (which is about preservation of relations) [2, 4, 29, 32], univalent type theory (preservation of equivalences) [6, 13, 14, 20, 21, 33, 34], directed type theory (preservation of morphisms), guarded type theory (preservation of the stage of advancement of computation) [10] and even combinations thereof [7, 12, 37].[1] The presheaf models cited almost all follow a common pattern: First one chooses a suitable base category $\mathcal{W}$. The presheaf category over $\mathcal{W}$ is automatically a model of dependent type theory with important basic type formers [18] as well as a tower of universes [19]. Next, one identifies a suitable notion of fibrancy and replaces or supplements the existing type judgement $\Gamma \vdash T\,\mathsf{type}$ with one that classifies fibrant types:

HoTT For homotopy type theory (HoTT, [39]), one considers Kan fibrant types, i.e. presheaves in which edges can be composed and inverted as in an ∞-groupoid. The precise definition may differ in different treatments.

Parametricity For parametric type theory, one considers discrete types [2, 12, 29, 32]: essentially those that satisfy Reynolds' identity extension property [36] which states that

[1] We omit models that are not explicitly structured as presheaf models [1, 22, 25].

homogeneously related objects are equal.

Directed In directed type theory, one may want to consider Segal, covariant, discrete and Rezk types [37] and possibly also Conduché types [15, 27].

Guarded In guarded type theory, one considers clock-irrelevant types [10]: types A such that any non-dependent function $\circleddash \to A$ from the clock type, is constant.

To the extent possible, one subsequently proves that the relevant notions of fibrancy are closed under basic type formers, so that we can restrict to fibrant types and still carry out most of the familiar type-theoretic reasoning and programming. Special care is required for the universe: it is generally straightforward to adapt the standard Hofmann-Streicher universe to classify only fibrant types, but the universe of fibrant types is in general not automatically fibrant itself. In earlier work on parametricity with Vezzosi [32, 29], we made the universe discrete by modifying its presheaf structure and introduced a parametric modality in order to use that universe. In contrast, Atkey et al. [2] and Cavallo and Harper [12] simply accept that their universes of discrete types are not discrete. In guarded type theory, Bizjak et al. [9] let the universe depend on a collection of in-scope clock variables lest the clock-indexed later modality $\triangleright : \forall(\kappa : \circleddash).\mathsf{U}_\Delta \to \mathsf{U}_\Delta$ (where $\kappa \in \Delta$) be non-dependent and therefore constant (not clock-indexed) by clock-irrelevance of $\mathsf{U}_\Delta \to \mathsf{U}_\Delta$ [10].

1.2 Internalizing the Power of Presheaves

Purely metatheoretic results about type theory certainly have their value. Parametricity, for instance, has originated and proven its value as a metatheoretic technique for reasoning about programs. However, with dependent type theory being not only a programming language but also a logic, it is preferrable to formulate results about it within the type system, rather than outside it.

Enlarging the end user's toolbox One motivation for internalizing metatheorems is to enlarge the toolbox of the end user of the proof assistant. If this is the only goal, then we can prove the desired results in the model on pen and paper and then internalize them ad hoc with an axiom with or without computation rules.

HoTT Book HoTT [39] simply postulates the univalence axiom without computational behaviour, as justified e.g. by the model of Kan-fibrant simplicial sets [21].
CCHM cubical type theory [14] provides the Glue type, which comes with introduction, elimination, β- and η-rules and which turns the univalence axiom into a theorem with computational behaviour. It also contains CCHM-Kan-fibrancy of all types as an axiom, in the form of the CCHM-Kan composition operator, with decreed computational behaviour that is defined by induction on the type.

Parametricity Bernardy, Coquand and Moulin [4, 24] (henceforth: Moulin et al.) internalize their (unary, but generalizable to k-ary) cubical set model of parametricity using two combinators Φ and Ψ [24], a.k.a. extent and Gel [12]. Φ internalizes the presheaf structure of the function type, and Ψ that of the universe.
The combinator Φ and at first sight also Ψ require that the cubical set model lacks diagonals. Indeed, to construct a value over the primitive interval, Φ and Ψ each take one argument for every endpoint and one argument for the edge as a whole. Nested use of these combinators, e.g. to create a square, will take $(k+1)^2$ arguments for k^2 vertices, $2k$ sides and 1 square as a whole but none for specifying the diagonal. For this reason, Moulin et al.'s type system enforces a form of *affine* use of interval variables.
In earlier work with Vezzosi [32], we have internalized parametricity instead using the Glue type [14] and its dual Weld. Later on, we added a primitive mill [30] for swapping

Weld and $\Pi(i : \mathbb{I})$. These operations are sound in presheaves over any base category where we can multiply with $\mathbb{I}$, and therefore strictly less expressive than Φ which is not. Discreteness of all types was internalized as a non-computing *path degeneracy* axiom.

Guarded In guarded type theory [10], one axiomatizes Löb induction and clock-irrelevance.

Internalizing fibrancy proofs Another motivation to internalize aspects of presheaf categories, is for building parts of the model inside the type theory, thus abstracting away certain categorical details such as the very definition of presheaves, and for some type systems enabling automatic verification of proofs. Given the common pattern in models described in the previous section, it is particularly attractive to try and define fibrancy and prove results about it internally.

In the context of HoTT, Orton and Pitts [33, 34] study CCHM-Kan-fibrancy [14] in a type theory satisfying a set of axioms, of which all but one serve to characterize the interval and the notion of cofibration. One axiom, *strictness*, provides a type former Strict for strictifying partial isomorphisms, which exists in every presheaf category. In order to prove fibrancy of the universe, Licata et al. postulate an "amazing right adjoint" $\mathbb{I} \surd \sqcup$ to the non-dependent path functor $\mathbb{I} \to \sqcup$ [23, 33], which indeed exists in presheaves over cartesian base categories if $\mathbb{I}$ is representable. Since $\mathbb{I} \surd \sqcup$ and its related axioms are global operations (only applicable to closed terms, unless you want to open Pandora's box as we do in the current paper), they keep everything sound by introducing a judgemental comonadic *global* modality $\flat$.

Orton et al.'s formalization [23, 33, 34] is only what we call *meta-internal*: the argument is internalized to *some* type theory which still only serves as a metatheory of the type system of interest. Ideally, we would define and prove fibrancy of types *within* the type theory of interest, which we call *auto-internal*. Such treatments exist of discrete types in parametricity [12], and discrete, Segal and Rezk types in directed type theory [37], but not yet for covariant or CCHM-Kan-fibrant types due to the need to consider paths in the context $\mathbb{I} \to \Gamma$.

1.3 The Transpension Type

What is striking about the previous section is that, while most authors have been able to solve their own problems, a common approach is completely absent. We have encountered Φ and Ψ [24], the amazing right adjoint $\surd$ [23], Glue [14, 32], Weld [32], mill [30] and the strictness axiom [34]. We have also seen that Φ and Ψ presently require an affine base category, and that $\surd$ presently requires the global modality $\flat$.

The goal of the current paper is to develop a smaller collection of internal primitives that impose few restrictions on the choice of base category and allow the internal construction of the aforementioned operators when sound. To this end, we introduce the **transpension** type former $≬i : \mathrm{Ty}(\Gamma) \to \mathrm{Ty}(\Gamma, i : \mathbb{I})$ which in cartesian settings is right adjoint to $\Pi(i : \mathbb{I}) : \mathrm{Ty}(\Gamma, i : \mathbb{I}) \to \mathrm{Ty}(\Gamma)$ and is therefore not a quantifier binding i, but a coquantifier that *depends* on it. Using the transpension and Strict, we can construct Φ (when sound), Ψ, $\surd$ and Glue. Given a type former for certain pushouts, we can also construct Weld.

The transpension coquantifier $≬(u : \mathbb{U}) : \mathrm{Ty}(\Gamma) \to \mathrm{Ty}(\Gamma, u : \mathbb{U})$ is part of a sequence of adjoints $\Sigma\, u \dashv \Omega\, u \dashv \Pi\, u \dashv ≬\, u$, preceded by the Σ-type, weakening and the Π-type. Adjointness of the first three is provable from the structural rules of type theory. However, it is not immediately clear how to add typing rules for a further adjoint. Birkedal et al. [8] explain how to add a single modality that has a left adjoint in the semantics. If we want to have two or more adjoint modalities internally, then we can use a multimodal type system such as MTT [16, 17]. Each modality in MTT needs a semantic left adjoint, so we can only internalize $\Omega\, u$, $\Pi\, u$ and $\between\, u$. A drawback which we accept, is that $\Omega\, u$ and $\Pi\, u$ become modalities which are a bit more awkward to deal with than ordinary weakening and Π-types.

1.4 Contributions

Our central contribution is to reduce the plethora of interal presheaf operators in the literature to only a few operations.

- To this end, we introduce the **transpension type** $\between(u : \mathbb{U})$, right adjoint to $\Pi(u : \mathbb{U})$, with typing rules built on *extensional* MTT [16, 17]. We explain how it is reminiscent of the suspension type from HoTT [39].
- More generally, the transpension type can be right adjoint to any quantifier-like operation $\forall(u : \mathbb{U})$ which need neither respect the exchange rule, nor weakening or contraction. In this setting, we also introduce the **fresh weakening** coquantifier $\text{Ⅎ}(u : \mathbb{U})$, which is left adjoint to $\forall(u : \mathbb{U})$ and therefore coincides with weakening $\Omega(u : \mathbb{U})$ in cartesian settings.
- We provide a categorical semantics for $\between(u : \mathbb{U})$ in almost any presheaf category $\mathrm{Psh}(\mathcal{W})$ over base category $\mathcal{W}$, for almost any representable object $\mathbb{U} = \mathbf{y}U$, $U \in \mathcal{W}$. To accommodate non-cartesian variables, our system is not parametrized by a representable object $\mathbb{U} = \mathbf{y}U$, but by an arbitrary endofunctor $\sqcup \ltimes U$ on $\mathcal{W}$: the **multiplier**. We introduce **criteria** for characterizing the multiplier – viz. semi-cartesian, cartesian, cancellative, affine, connection-free and quantifiable – which we use as requirements for internal type theoretic features. We identify a complication dubbed **spookiness** in certain models (most notably in guarded type theory), and define dimensionally split morphisms (a generalization of split epimorphisms) in order to include spooky models. We exhibit relevant multipliers in base categories found in the literature (Example 4.4).
- We show that **all general presheaf internalization operators** that we are aware of – viz. Φ/extent (when sound), Ψ/Gel [24, 4], the amazing right adjoint $\surd$ [23], Glue [14, 32], Weld [32] and mill [30] – can be **recovered** from just the transpension type, the strictness axiom and pushouts along $\mathsf{snd} : \varphi \times A \to A$ where $\varphi : \mathsf{Prop}$. In the process, some of these operators can be **improved**: We generalize Ψ to arbitrary multipliers, including cartesian ones and we justify $\mathbb{U} \surd \sqcup$ without a global modality and get proper computation rules for it. Moreover, since our system provides an operation $\text{🔒}_{\between\, u}$ for quantifying over contexts, we take a step towards auto-internalizing Orton et al.'s work [23, 33, 34]. When Φ is not sound (e.g. in cartesian or non-connection-free settings), we introduce the internal notion of **transpensive** types to retain some of its power. Finally, a form of higher dimensional pattern matching is enabled by exposing $\forall(u : \mathbb{U})$ internally as a left adjoint.
- In a technical report [28], we investigate how the modalities introduced in this paper commute with each other, and with prior modalities (i.e. those already present before adding the transpension type). We also consider 2-cells arising from multiplier morphisms.

While MTT [16, 17] satisfies canonicity, decidable type-checking will at least require a computational understanding of the mode theory, and of some new typing rules that we add to MTT. For this reason, we build on extensional MTT [17], and defer decidability and canonicity to future work.

Overview of the paper In Section 2, we demonstrate in a simple setting how the transpension resembles the suspension from HoTT and how it allows for higher-dimensional pattern matching. In Section 3, we give a brief overview of the typing rules of MTT and introduce our notation using *ticks*. In Section 4, we present the transpension type and its typing rules by instantiating MTT on a specific mode theory with a transpension modality. In Section 5, we explain how to recover known internal presheaf operators. We conclude in Section 6.

$$\frac{\sigma : \Gamma \to \Gamma'}{(\sigma, u/v) : (\Gamma, u : \mathbb{U}) \to (\Gamma', v : \mathbb{U})} \qquad \frac{\sigma : \Gamma \to \Gamma'}{\sigma : (\Gamma, u : \mathbb{U}) \to \Gamma'} \qquad \frac{\Gamma, u : \mathbb{U} \vdash A\,\mathsf{type}}{\Gamma \vdash \forall\, u.A\,\mathsf{type}} \qquad \frac{\Gamma, u : \mathbb{U} \vdash a : A}{\Gamma \vdash \lambda u.a : \forall\, u.A}$$

$$\frac{\Gamma \vdash f : \forall\, u.A}{\Gamma, u : \mathbb{U}, \delta : \Delta \vdash f\, u : A}$$

$$\frac{\begin{array}{l}\Gamma, u : \mathbb{U} \vdash (\delta : \Delta)\,\mathsf{telescope} \\ \Gamma, \forall\, u.(\delta : \Delta) \vdash A\,\mathsf{type}\end{array}}{\Gamma, u : \mathbb{U}, \delta : \Delta \vdash \between(u : \mathbb{U}).A\,\mathsf{type}} \qquad \frac{\Gamma, \forall\, u.(\delta : \Delta) \vdash a : A}{\Gamma, u : \mathbb{U}, \delta : \Delta \vdash \mathsf{mer}\, u\, a : \between(u : \mathbb{U}).A}$$

$$\text{where } \mathsf{mer}\, u\, (\mathsf{unmer}(v.t[v/u, \delta|_{u=v}/\delta])) = t$$

$$\frac{\Gamma, u : \mathbb{U} \vdash t : \between\, u.(A[\theta/\theta|_{u=\sqcup}])}{\Gamma \vdash \mathsf{unmer}(u.t) : A}$$

$$\text{where } \mathsf{unmer}(u.\mathsf{mer}\, u\, a) = a$$

$$\begin{aligned} \forall\, u.() &= () \\ \forall\, u.(\delta : \Delta, x : A) &= (\forall\, u.(\delta : \Delta)), x|_{u=\sqcup} : \forall\, u.(A[\delta|_{u=u}/\delta]) \end{aligned}$$

Figure 1 Selection of typing rules for a naïve transpension type.

2 A Naïve Transpension Type

In this section, for purposes of demonstration, we present simplified typing rules for the transpension. Using these, we will already be able to exhibit the transpension as similar to a dependent version of the suspension in HoTT [39]. Moreover, in order to showcase how the transpension type allows us to internalize the presheaf structure of other types, we will demonstrate a technique which we call higher-dimensional pattern matching.

Typing rules To do this, we first present, in Figure 1, typing rules for the transpension type in a very simple setting: a type system with affine shape variables $u : \mathbb{U}$. Variables to the left of u are understood to be fresh for u; variables introduced after u may be substituted with terms depending on u. In particular, we have no contraction $(w/u, w/v) : (w : \mathbb{U}) \to (u, v : \mathbb{U})$, while exchange $(x : A, u : \mathbb{U}) \to (u : \mathbb{U}, x : A)$ only works in one direction. This is enforced by the special substitution rules for shape variables.

The system features an affine function type $\forall\, u.A$ over $\mathbb{U}$, with unsurprising formation and introduction rules. The elimination rule for $f\, u$ requires that the function f be fresh for u, i.e. that f depend only on variables to the left of u [4, 24].

Additionally, the system contains a transpension type $\between\, u.A$ over $\mathbb{U}$, with more unusual rules. When checking the type $\between(u : \mathbb{U}).A$ in context $(\Gamma, u : \mathbb{U}, y : B)$, the part A will be checked in a modified context [5, 31], were $y : B$ (which potentially depends on u) will change type, becoming a function $y|_{u=\sqcup} : \forall\, u.B$ that can be applied to $v : \mathbb{U}$ yielding $y|_{u=v} : B[v/u]$. In other words, we get hold of the dependency of y on u. The *meridian* constructor $\mathsf{mer}\, u\, a$ is checked in a similar way, and is modelled by transposition for the adjunction $\forall\, u \dashv \between\, u$. We remark that both $\between\, u.A$ and $\mathsf{mer}\, u\, a$ depend on u, whereas A and a do not, so in a way the transpension lifts data to a higher dimension, turning points into $\mathbb{U}$-cells. The elimination rule takes data again to a lower dimension: it turns a dependent $\mathbb{U}$-cell in the transpension into a point in A. This is the co-unit of the adjunction. The β- and η-rules internalize the adjunction laws.

These typing rules are sound in certain presheaf categories (those where $\mathbb{U}$ is cancellative and affine, see Definition 4.1), but are unsatisfactory in several respects. First, we have no story for substitutions which exist in cubical type systems such as $(0/i) : \Gamma \to (\Gamma, i : \mathbb{I})$ [4, 6, 14] or $(j \wedge k/i) : (\Gamma, j, k : \mathbb{I}) \to (\Gamma, i : \mathbb{I})$ [14], as there is no formation rule for $\between\, 0.A$ or $\between\, (j \wedge k).A$. Secondly, in non-affine generalizations, the transpension is not stable under substitution of the variables preceding u [28]. In order to obtain a better behaved type system, in the rest of the paper we will rely on MTT, which we briefly summarize in Section 3.

Poles We can still try to get a grasp on $\between 0.A$, however. In general we have $T[0/i] \cong (\forall\, i.(i = 0) \to T)$. For $T = \between\, i.A$, the latter type is inhabited by $\lambda i.\lambda e.\mathsf{mer}\, i\, (\lightning\, e|_{i=1})$, since $e|_{i=1}$ proves $1 = 0$. Moreover, using the η-rule, we can show that this is the only element.

Thus we see that the transpension type essentially consists of meridians $(i : \mathbb{I}) \to \between\, i.T$ for all $t : T$ which are all equal when $i = 0$ or $i = 1$. This makes the transpension type quite reminiscent of a dependent version of the suspension type from HoTT [39], although the quantification of the context is obviously a distinction.

Higher-dimensional pattern matching Given two types $A, B : \mathsf{U}$, higher-dimensional pattern matching allows us to construct a function $\Gamma \vdash f : (\forall\, u.A \uplus B) \to (\forall\, u.A) \uplus (\forall\, u.B)$. This function expresses that any $\mathbb{U}$-shaped cell in the coproduct type $A \uplus B$ must be either a cell in A or a cell in B. In that sense, it exposes the presheaf structure of the coproduct type $A \uplus B$. We can define f as follows (and generalization to $A, B : \forall\, u.\mathsf{U}$ is straightforward):

$$f\ \hat{c} = \mathsf{unmer}\big(u.\mathsf{case}\,\hat{c}\,u\,\mathsf{of}\big\{\ \mathsf{inl}\,a \mapsto \mathsf{mer}\,u\,(\mathsf{inl}\,(\lambda v.a|_{u=v}))\mid \mathsf{inr}\,b \mapsto \mathsf{mer}\,u\,(\mathsf{inr}\,(\lambda v.b|_{u=v}))\ \big\}\big)$$

The argument to $\mathsf{unmer}(u.\sqcup)$ should be of type $\between\, u.(\forall\, u.A) \uplus (\forall\, u.B)$. Interestingly, since we have u in scope, we can apply $\hat{c} : \forall\, u.\,(A \uplus B)$ to it, and pattern match to decide which case we are in. Both cases are analogous; in the first case, a variable $a : A$ is brought in scope, so we are in context $(\Gamma, \hat{c} : \forall\, u.A \uplus B, u : \mathbb{U}, a : A)$. We then use the constructor $\mathsf{mer}\,u\,\sqcup$, which again removes u from scope and turns $a : A$ into a function $a|_{u=\sqcup} : \forall\, u.A$. Then we trivially finish the proof by writing $\mathsf{inl}\,(\lambda v.a|_{u=v})$, where we have η-expanded $a|_{u=\sqcup}$ mainly for facilitating further narrative. In summary, between unmer and mer, we had temporary access to a variable $u : \mathbb{U}$ which allowed us to pattern-match on $\hat{c}\,u$. More conceptually, we can say that the transpension allows us to temporarily work with $\hat{c}$ as if it were a lower-dimensional value of type $A \uplus B$. We will see in Section 5 how similar ideas can be used to implement other internalization operators. Interestingly, this construction of f using the transpension also comes with suitable computational behavior. When we evaluate $f\,(\lambda u.\mathsf{inl}(\hat{a}\,u))$, then $\hat{a}\,u$ is substituted for a. Next, $(\hat{a}\,u)|_{u=v}$ simplifies to $\hat{a}\,v$, so we can η-contract $\lambda v.\hat{a}\,v$, and β-reduce unmer, which yields $\mathsf{inl}\,\hat{a}$ as expected.

3 Multimode Type Theory

As announced, we will rely on the extensional version of Gratzer et al.'s multimode and multimodal dependent type system MTT [16, 17] in order to frame the transpension and its left adjoints as modal operators. We refer to the original work for details, but in this section we highlight some important aspects of MTT and introduce our notation using *ticks*.

The mode theory MTT is parametrized by a *mode theory*, which is a strict 2-category whose objects, morphisms and 2-cells we will refer to as **modes**, **modalities** and, well, **2-cells** respectively. Semantically, every mode p will correspond to an entire model of dependent type theory $[\![p]\!]$. A modality $\mu : p \to q$ will consist of a functor $[\![\text{🔒}_\mu]\!] : [\![q]\!] \to [\![p]\!]$ and an operation $[\![\mu]\!]$ that is almost a dependent right adjoint (DRA [8]) to $[\![\text{🔒}_\mu]\!]$; for all our purposes it will be an actual DRA and even one arising from a weak CwF morphism [8, 26]. A 2-cell $\alpha : \mu \Rightarrow \nu$ is interpreted as a natural transformation $[\![\alpha\downarrow]\!] : [\![\text{🔒}_\nu]\!] \to [\![\text{🔒}_\mu]\!]$ and hence also gives rise to an appropriate transformation $[\![\alpha]\!] : [\![\mu]\!] \to [\![\nu]\!]$.

Judgement forms The judgement forms of MTT are listed in Figure 2. All forms are annotated with a mode p which specifies in what category they are to be interpreted. Every judgement form also has a corresponding equality judgement, which is respected by everything

$p \mid \Gamma\,\mathsf{ctx}$	Γ is a context at mode p,
$p \mid \sigma : \Gamma \to \Delta$	σ is a simultaneous substitution from Γ to Δ at mode p,
$p \mid \Gamma \vdash T\,\mathsf{type}$	T is a type in context Γ at mode p,
$p \mid \Gamma \vdash t : T$	t has type T in context Γ at mode p.

$$\frac{q\,\mathsf{mode}}{q \mid \cdot\,\mathsf{ctx}} \qquad \frac{q \mid \Gamma\,\mathsf{ctx} \quad \mu : p \to q}{\begin{array}{l} p \mid \Gamma, \text{🔒}^{\mathfrak{m}}_{\mu}\,\mathsf{ctx} \\ \text{where } (\Gamma, \text{🔒}^{\bullet}_{1}) = \Gamma, \quad (\Gamma, \text{🔒}^{\mathfrak{n}}_{\nu}, \text{🔒}^{\mathfrak{m}}_{\mu}) = (\Gamma, \text{🔒}^{\mathfrak{n}\,\mathfrak{m}}_{\nu \circ \mu}) \end{array}} \qquad \frac{\begin{array}{l} q \mid \Gamma\,\mathsf{ctx} \quad \mu : p \to q \\ p \mid \Gamma, \text{🔒}^{\mathfrak{m}}_{\mu} \vdash T\,\mathsf{type} \end{array}}{q \mid \Gamma, \mu \mid x :^{\mathfrak{m}} T\,\mathsf{ctx}}$$

$$\frac{q \mid \sigma : \Gamma \to \Delta \quad \mu : p \to q \quad p \mid \Gamma, \text{🔒}^{\mathfrak{m}}_{\mu} \vdash t : T[\sigma]}{q \mid (\sigma, t/x\!\downarrow_{\mathfrak{m}}) : \Gamma \to (\Delta, \mu \mid x :^{\mathfrak{m}} T)} \qquad \frac{q \mid \sigma : \Gamma \to \Delta \quad \mu : p \to q \quad p \mid \Gamma, \text{🔒}^{\mathfrak{m}}_{\mu} \vdash T\,\mathsf{type}}{q \mid \sigma : (\Gamma, \mu \mid x :^{\mathfrak{m}} T) \to \Delta}$$

$$\frac{q \mid \sigma : \Gamma \to \Delta \quad \mu, \nu : p \to q \quad \alpha : \mu \Rightarrow \nu}{\begin{array}{l} p \mid (\sigma, \alpha\!\downarrow^{\mathfrak{m}}_{\mathfrak{n}}) : (\Gamma, \text{🔒}^{\mathfrak{n}}_{\nu}) \to (\Delta, \text{🔒}^{\mathfrak{m}}_{\mu}) \\ \text{where } (\sigma, \alpha'\!\downarrow^{\mathfrak{m}'}_{\mathfrak{n}'}, \alpha\!\downarrow^{\mathfrak{m}}_{\mathfrak{n}}) = (\sigma, (\alpha' \star \alpha)\!\downarrow^{\mathfrak{m}'\mathfrak{m}}_{\mathfrak{n}'\mathfrak{n}}) \\ \qquad\quad (\sigma, \alpha\!\downarrow^{\mathfrak{m}}_{\mathfrak{n}}) \circ (\sigma, \beta\!\downarrow^{\mathfrak{n}}_{\mathfrak{o}}) = (\sigma, (\beta \circ \alpha)\!\downarrow^{\mathfrak{m}}_{\mathfrak{o}}) \end{array}} \qquad \frac{\mu : p \to q \quad \alpha : \mu \Rightarrow \mathsf{locks}(\Delta)}{\begin{array}{l} q \mid \Gamma, \mu \mid x :^{\mathfrak{m}} T, \Delta \vdash x\,\alpha\!\downarrow_{\mathsf{ticks}(\Delta)} : T[\alpha\!\downarrow^{\mathfrak{m}}_{\mathsf{ticks}(\Delta)}] \\ \text{where } x\,\alpha\!\downarrow_{\mathsf{ticks}(\Delta)}[\mathrm{id}_\Gamma, t/x\!\downarrow_{\mathfrak{m}}, \mathrm{id}_\Delta] = t[\alpha\!\downarrow^{\mathfrak{m}}_{\mathsf{ticks}(\Delta)}] \\ \qquad\quad x\,\alpha\!\downarrow_{\mathsf{ticks}(\Delta)}[\beta\!\downarrow^{\mathsf{ticks}(\Delta)}_{\mathsf{ticks}(\Theta)}] = x\,(\beta \circ \alpha)\!\downarrow_{\mathsf{ticks}(\Theta)} \end{array}}$$

$$\begin{array}{lll} \mathsf{locks}(\cdot) = 1 & \mathsf{locks}(\Delta, \text{🔒}^{\mathfrak{m}}_{\mu}) = \mathsf{locks}(\Delta) \circ \mu & \mathsf{locks}(\Delta, \mu \mid x :^{\mathfrak{m}} T) = \mathsf{locks}(\Delta) \\ \mathsf{ticks}(\cdot) = \bullet & \mathsf{ticks}(\Delta, \text{🔒}^{\mathfrak{m}}_{\mu}) = \mathsf{ticks}(\Delta)\mathfrak{m} & \mathsf{ticks}(\Delta, \mu \mid x :^{\mathfrak{m}} T) = \mathsf{ticks}(\Delta) \end{array}$$

$$\frac{\begin{array}{l} \mu : p \to q \\ p \mid \Gamma, \text{🔒}^{\mathfrak{m}}_{\mu} \vdash A\,\mathsf{type}_\ell \end{array}}{q \mid \Gamma \vdash \langle \mu \mid^{\mathfrak{m}} A \rangle\,\mathsf{type}_\ell} \qquad \frac{\begin{array}{l} \mu : p \to q \\ p \mid \Gamma, \text{🔒}^{\mathfrak{m}}_{\mu} \vdash a : A \end{array}}{q \mid \Gamma \vdash \mathsf{mod}^{\mathfrak{m}}_{\mu}\, a : \langle \mu \mid^{\mathfrak{m}} A \rangle} \qquad \frac{q \mid \Gamma\,\mathsf{ctx} \quad \ell \in \mathbb{N}}{q \mid \Gamma \vdash \mathsf{U}^q_\ell\,\mathsf{type}_{\ell+1}}$$

$$\frac{\begin{array}{l} \mu : p \to q \quad \nu : q \to r \\ q \mid \Gamma, \text{🔒}^{\mathfrak{n}}_{\nu} \vdash \hat{a} : \langle \mu \mid^{\mathfrak{m}} A \rangle \\ r \mid \Gamma, \nu \mid \hat{x} :^{\mathfrak{n}} \langle \mu \mid^{\mathfrak{m}} A \rangle \vdash C\,\mathsf{type} \\ r \mid \Gamma, \nu \circ \mu \mid x :^{\mathfrak{o}} A \vdash c : C[\mathsf{mod}^{\mathfrak{m}}_{\mu}\, x\!\downarrow_{\mathfrak{n}\,\mathfrak{m}} / \hat{x}\!\downarrow_{\mathfrak{n}}] \end{array}}{\begin{array}{l} r \mid \Gamma \vdash \mathsf{let}^{\mathfrak{n}}_{\nu}\,(\mathsf{mod}^{\mathfrak{m}}_{\mu}\, x\!\downarrow_{\mathfrak{n}\,\mathfrak{m}} = \hat{a})\,\mathsf{in}\,c : C[\hat{a}/\hat{x}\!\downarrow_{\mathfrak{n}}] \\ \text{where } \mathsf{let}^{\mathfrak{n}}_{\nu}\,(\mathsf{mod}^{\mathfrak{m}}_{\mu}\, x\!\downarrow_{\mathfrak{n}\,\mathfrak{m}} = \mathsf{mod}^{\mathfrak{m}}_{\mu}\, a)\,\mathsf{in}\,c = c[a/x\!\downarrow_{\mathfrak{n}\,\mathfrak{m}}] \end{array}} \qquad \frac{q \mid \Gamma \vdash t : \mathsf{U}^q_\ell}{\begin{array}{l} q \mid \Gamma \vdash \mathsf{El}(t)\,\mathsf{type}_\ell \\ \text{where } \mathsf{El}(\ulcorner T \urcorner) = T \end{array}} \qquad \frac{q \mid \Gamma \vdash T\,\mathsf{type}_\ell}{\begin{array}{l} q \mid \Gamma \vdash \ulcorner T \urcorner : \mathsf{U}^q_\ell \\ \text{where } \ulcorner \mathsf{El}(t) \urcorner = t \end{array}}$$

$$\frac{\begin{array}{l} p \mid \Gamma, \text{🔒}^{\mathfrak{m}}_{\mu} \vdash A\,\mathsf{type}_\ell \quad \mu : p \to q \\ q \mid \Gamma, \mu \mid x :^{\mathfrak{m}} A \vdash B\,\mathsf{type}_\ell \end{array}}{q \mid \Gamma \vdash (\mu \mid x :^{\mathfrak{m}} A) \to B\,\mathsf{type}_\ell} \qquad \frac{\begin{array}{l} q \mid \Gamma \vdash f : (\mu \mid x :^{\mathfrak{m}} A) \to B \\ p \mid \Gamma, \text{🔒}^{\mathfrak{m}}_{\mu} \vdash a : A \quad \mu : p \to q \end{array}}{\begin{array}{l} q \mid \Gamma \vdash f \cdot^{\mathfrak{m}} a : B[a/x] \\ \text{where } (\lambda(\mu \mid x).b) \cdot^{\mathfrak{m}} a = b[a/x\!\downarrow_{\mathfrak{m}}] \end{array}}$$

$$\frac{q \mid \Gamma, \mu \mid x :^{\mathfrak{m}} A \vdash b : B \quad \mu : p \to q}{\begin{array}{l} q \mid \Gamma \vdash \lambda(\mu \mid x).b : (\mu \mid x :^{\mathfrak{m}} A) \to B \\ \text{where } \lambda(\mu \mid x).f \cdot^{\mathfrak{m}} x\!\downarrow_{\mathfrak{m}} = f \end{array}}$$

Figure 2 Judgement forms and selection of typing rules from MTT [16, 17].

as the typing rules are to be read as a specification of a generalized algebraic theory (GAT [11]). The statements $p\,\mathsf{mode}$, $\mu : p \to q$ and $\alpha : \mu \Rightarrow \nu$ are simply requirements about the mode theory. This means we give no syntax or equality rules for modalities and 2-cells: these are fixed by the choice of mode theory.

Typing rules Figure 2 lists some selected typing rules. We will discuss these in turn.

The formation and introduction rules of **modal types** $\langle \mu \mid A \rangle$ (which should help to understand the more basic rules) work by transposition: we apply the left adjoint (in the form of a lock) to the context of their argument. In order to disambiguate in certain situations, we choose to enrich the original MTT syntax by annotating locks with a name, which we call **ticks** after Bahr et al. [3]. So $\langle \mu \mid^{\mathfrak{m}} A \rangle$ and $\mathsf{mod}^{\mathfrak{m}}_{\mu}\, a$ will both bind a tick $\mathfrak{m}$.

Context formation starts with the empty context which exists at any mode, and proceeds by adding locks and variables. Adding locks is strictly functorial, and we take the liberty to use strings like $\mathfrak{n}\mathfrak{m}$ and $\bullet$ as ticks. A modal variable $\mu \mid x :^{\mathfrak{m}} T$ is essentially the same as a non-modal variable $\hat{x} : \langle \mu \mid^{\mathfrak{m}} T \rangle$ (which in turn is shorthand for $1 \mid \hat{x} :^{\bullet} \langle \mu \mid^{\mathfrak{m}} T \rangle$), but the judgemental modal annotation allows direct access to a term of type A through the variable rule. Hence, the type T is checked the same way as it would be in $\langle \mu \mid^{\mathfrak{m}} T\rangle$.

Terms **substituted** for a modal variable x are also checked in the locked context, and therefore term substitution binds a tick. Context extension with a lock is bifunctorial: we can combine a substitution and a 2-cell to a substitution between locked contexts. If the 2-cell is the identity and we are just renaming locks, then we write $\downarrow^{\mathfrak{m}}_{\mathfrak{m}'}$ for $1_\mu\downarrow^{\mathfrak{m}}_{\mathfrak{m}'}$. The basic **variable** rule would be $\Gamma, \mu \shortmid x :^{\mathfrak{m}} T, \text{🔒}^{\mathfrak{m}}_{\mu} \vdash x\downarrow_{\mathfrak{m}} : T$. A 2-cell $\alpha : \mu \Rightarrow \nu$ takes the variable to a context with $\text{🔒}^{\mathfrak{n}}_{\nu}$ instead of $\text{🔒}^{\mathfrak{m}}_{\mu}$. By functoriality of locks, $\text{🔒}^{\mathfrak{n}}_{\nu}$ may be split into multiple locks, and by weakening we may insert types in between. Building in these substitutions, we get the general variable rule which features an arbitrary telescope Δ with $\alpha : \mu \Rightarrow \mathsf{locks}(\Delta)$.

Modal elimination uses a let-syntax to turn the modal type into a judgemental annotation on a variable. Modal **function type** formation and introduction are by simple abstraction. Modal function application binds a tick. We have a **universe** à la Coquand with mutually inverse encoding and decoding operations (which we will henceforth suppress). We ignore cumulativity-related hassle, and refer to Gratzer et al. for details.

Results We highlight some results about MTT that are relevant in the current paper.

▶ **Proposition 3.1.** *We have* $\langle 1 \mid^{\bullet} A\rangle \cong A$ *and* $\langle \nu \circ \mu \mid^{\mathfrak{n}\mathfrak{m}} A\rangle \cong \langle \nu \mid^{\mathfrak{n}} \langle \mu \mid^{\mathfrak{m}} A\rangle\rangle$.

▶ **Proposition 3.2.** *For any 2-cell* $\alpha : \mu \Rightarrow \nu$*, we have* $\langle \mu \mid^{\mathfrak{m}} A\rangle \to \langle \nu \mid^{\mathfrak{n}} A[\alpha\downarrow^{\mathfrak{m}}_{\mathfrak{n}}]\rangle$.

▶ **Proposition 3.3.** *If* $\kappa \dashv \mu$ *internal to the mode theory, there is a function* $\mathsf{prmod}_{\mu} : (\kappa \shortmid^{\mathfrak{t}} \langle \mu \mid^{\mathfrak{m}} A\rangle) \to A[\varepsilon\downarrow^{\mathfrak{t}\mathfrak{m}}_{\bullet}]$*, satisfying a* β*- and (thanks to extensionality) an* η*-law. Combined with these rules,* prmod_{μ} *is equally expressive as the* let*-eliminator for* $\langle \mu \mid \sqcup\rangle$.

▶ **Proposition 3.4.** *If* $\kappa \dashv \mu$ *internal to the mode theory, there is an isomorphism of contexts* $\sigma = (x\,\eta\downarrow_{\mathfrak{m}\mathfrak{t}}/y\downarrow_{\mathfrak{t}}) : (\Gamma, x : A, \text{🔒}^{\mathfrak{m}}_{\mu}) \cong (\Gamma, \text{🔒}^{\mathfrak{m}}_{\mu}, \kappa \shortmid y :^{\mathfrak{t}} A[\eta\downarrow^{\bullet}_{\mathfrak{m}\mathfrak{t}}])$ *with inverse* $\sigma^{-1} = (\mathrm{id}_{\Gamma}, \eta\downarrow^{\bullet}_{\mathfrak{m}\mathfrak{t}}, y\downarrow_{\mathfrak{t}}/x\downarrow_{\bullet}, \downarrow^{\mathfrak{m}}_{\mathfrak{m}'}) \circ (\mathrm{id}_{(\Gamma, \text{🔒}^{\mathfrak{m}}_{\mu}, y)}, \varepsilon\downarrow^{\mathfrak{t}\mathfrak{m}'}_{\bullet})$*. Correspondingly, given* B *in the latter context, there is an isomorphism of types* $\big((x : A) \to \langle \mu \mid^{\mathfrak{m}} B[\sigma^{-1}]\rangle\big) \cong \langle \mu \mid^{\mathfrak{m}} (\kappa \shortmid y :^{\mathfrak{t}} A[\eta\downarrow^{\bullet}_{\mathfrak{m}\mathfrak{t}}]) \to B\rangle$.

4 A Type System for Shape (Co)quantification

In Section 4.1, we instantiate MTT with a mode theory that contains a transpension modality. In Section 4.2, we add a few additional typing rules to MTT. In Section 4.3, we investigate the structure of the transpension type.

4.1 A Mode Theory for Shape (Co)quantification

For space reasons, we assume that there are no prior modalities, i.e. that the type system to which we wish to add a transpension type, is non-modal in the sense that it has a single mode and only the identity modality. Prior modalities are considered in the technical report [28]. We assume that this single prior mode is modelled by the presheaf category $\mathrm{Psh}(\mathcal{W})$.

Shape contexts A first complication is that the modalities $\Omega\, u \dashv \Pi\, u \dashv between\, u$ all bind or depend on a variable, a phenomenon which is not supported by MTT. However, the following trick solves this problem. Assume we have in the prior system a context Ξ modelled by a presheaf over $\mathcal{W}$. Then the presheaves $\mathrm{Psh}(\mathcal{W}/\Xi)$ over the category of elements of the presheaf Ξ are also a model of dependent type theory. Denoting the judgements of the latter system with a prefix $\Xi \mid$, it happens to be the case that judgements $\Xi \mid \Gamma \vdash J$ have precisely the same meaning as judgements $\Xi.\Gamma \vdash J$ (for a suitable but straightforward translation of J).

Thus, we will group together all shape variables (variables for which we want a transpension type) in a **shape context** Ξ in front of the typing context. Our judgements will then take the form $\Xi \mid \Gamma \vdash J$. This allows us to frame Ξ as the *mode* of the judgement.

Mode Theory For simplicity, we take a highly general mode theory and will then only be able to say interesting things about specific modes, modalities and 2-cells. In practice, and especially in implementations, one will want to select a subtheory right away.

As **modes**, we take the set of all small presheaves over $\mathcal{W}$, which we think of as **shape contexts**. The mode Ξ is modelled in $\mathrm{Psh}(\mathcal{W}/\Xi)$. As **modalities** $\mu : \Xi_1 \to \Xi_2$, we take all functors $[\![🔒_\mu]\!] : \mathrm{Psh}(\mathcal{W}/\Xi_2) \to \mathrm{Psh}(\mathcal{W}/\Xi_1)$ which have a right adjoint μ that is then automatically a weak CwF morphism [28] and gives rise to a DRA [8, 26]. As **2-cells** $\alpha : \mu \Rightarrow \nu$, we take all natural transformations.

Shapes and Multipliers We now proceed by highlighting some interesting modes, modalities, and 2-cells. To begin with, we fix a collection of shapes. We associate to each shape $\mathbb{U}$ a functor $\sqcup \ltimes U : \mathcal{W} \to \mathcal{W}$ which extends to a functor $\sqcup \ltimes \mathbf{y}U : \mathrm{Psh}(\mathcal{W}) \to \mathrm{Psh}(\mathcal{W})$.[2] Internally, we will use shape variables to increase human readability and to reduce the heaviness of notation for shape substitutions, writing $(\Xi, u : \mathbb{U})$ for the shape context $\Xi \ltimes \mathbf{y}U$. However, in a fully elaborate syntax these variables would be redundant.

▶ **Definition 4.1.** *Assume $\mathcal{W}$ has a terminal object $\top$. A **multiplier** for an object U is a functor $\sqcup \ltimes U : \mathcal{W} \to \mathcal{W}$ such that $\top \ltimes U \cong U$.*[3] *This gives us a natural second projection $\pi_2 : (\sqcup \ltimes U) \to U$. We define the **fresh weakening functor** to the slice category as $\exists_U : \mathcal{W} \to \mathcal{W}/U : W \mapsto (W \ltimes U, \pi_2)$. We say that a multiplier (as well as its shape) is:*

- ***Semicartesian** if it is copointed, i.e. has a first projection $\pi_1 : (\sqcup \ltimes U) \to \mathrm{Id}$,*
- ***Cartesian** if it is naturally isomorphic to the cartesian product with U,*
- ***Cancellative** if $\exists_U$ is faithful,*
- ***Affine** if $\exists_U$ is full,*
- ***Connection-free** if $\exists_U$ is essentially surjective on objects (V, ψ) such that ψ is dimensionally split (Definition 4.3),*
- ***Quantifiable** if $\exists_U$ has a left adjoint $\exists_U : \mathcal{W}/U \to \mathcal{W}$.*

Variables of shape $\mathbb{U}$ admit weakening if and only if the multiplier is semicartesian, and exchange and contraction if (but not only if) it is cartesian. Weakening, exchange and contraction are all shape substitutions, which will be internalized as modalities.

▶ **Proposition 4.2.** *If a multiplier is affine and cartesian, then it is the identity functor. [28]*

▶ **Definition 4.3.** *A morphism $\psi : V \to U$ is called **dimensionally split** (w.r.t. $\sqcup \ltimes U$) if there is some W such that $\pi_2 : W \ltimes U \to U$ factors over ψ. We define the **boundary** ∂U as the subpresheaf of the Yoneda-embedding $\mathbf{y}U$ consisting of those morphisms that are* not *dimensionally split, and we define $\sqcup \ltimes \partial U$ by pullback. We also write $(\Xi, u : \partial\mathbb{U})$ for $\Xi \ltimes \partial U$.*

In most popular base categories, all morphisms to $\top$ are split epi. Being dimensionally split is then equivalent to being split epi. We call a category **spooky** if some morphism to $\top$ is not split epi. The notion of dimensionally split morphisms lets us consider the boundary and connection-freedom (a requirement for modelling Φ) also in spooky base categories.

[2] Both $\sqcup \ltimes U$ and $\sqcup \ltimes \mathbf{y}U$ are to be regarded as single-character symbols, i.e. $\ltimes$ in itself is meaningless.
[3] In the technical report [28], we generalize beyond endofunctors.

Base category	$\mathcal{W}$	$\mathcal{W}$	$\square^k$	$\boxslash^k$	CCHM	🕒	BPCube	$\bowtie$	Srp
Multiplier	Id	$\times W$	$*\mathbb{I}$	$\times\mathbb{I}$	$\times\mathbb{I}$	$\times$🕒$_k$	$\times\mathbb{B}, \times\mathbb{P}$	tw-prism	$\times\bot$
Spooky	?	?	if $k = 0$		✗	✓	✗	✗	✓
Wkn. (semicart.)	✓	✓	✓	✓	✓	✓	✓	✗	✓
Exchange	✓	✓	✓	✓	✓	✓	✓	✗	✓
Contraction	✓	✓	✗	✓	✓	✓	✓	✗	✓
Cartesian	✓	✓	✗	✓	✓	✓	✓	✗	✓
Cancellative	✓	?	✓	✓	✓	✓	✓	✓	✓
Affine	✓	✗	✓	✗	✗	✗	✗	✓	✗
Connection-free	✓	?	✓	✓	✗	✓	✓	✓	✓
Quantifiable	✓	✓	✓	✓	✓	✓	✓	✓	✓

Figure 3 Some interesting multipliers and their properties (see Example 4.4 for a legend).

▶ **Example 4.4.** Figure 3 lists a number of multipliers and their properties. From left to right: the identity on an arbitrary category $\mathcal{W}$ (spooky or not); cartesian product with any $W \neq \top$ in $\mathcal{W}$; product with a line $\mathbb{I}$ in the categories of k-ary affine cubes, cartesian cubes[4] and (binary) CCHM cubes [14]; product with a clock 🕒$_k$ of longevity k in the category of clocks 🕒 [10]; product with a bridge $\mathbb{B}$ or path $\mathbb{P}$ in the category BPCube of bridge/path-cubical sets [26, 32] (identical results would hold for depth n cubes [26, 29]); the twisted prism functor on twisted cubes $\bowtie$ [35]; product with $\bot$ (constant functor on $\bot$) in the base category of the Sierpiński topos $\mathsf{Srp} = \{\bot \to \top\}$. Connection-freedom in CCHM is violated by $\vee, \wedge : \mathbb{I}^2 \to \mathbb{I}$.

Modalities for substitution A substitution $\sigma : \Xi_1 \to \Xi_2$ gives rise to a functor $\Sigma^{/\sigma} : \mathcal{W}/\Xi_1 \to \mathcal{W}/\Xi_2$ and hence [38] to a triple of adjoint functors $\Sigma\,\sigma \dashv \Omega\,\sigma \dashv \Pi\,\sigma$ between the presheaf categories, where $\Omega\,\sigma$ has the exact same semantics as ordinary substitution. The two functors $\Omega\,\sigma$ and $\Pi\,\sigma$ give rise to DRAs and can be internalized as MTT modalities. We denote these as[5] $\Omega\,\sigma$ or $\Omega(\xi_1 : \Xi_1, \xi_2 = \xi_2[\sigma]) : (\xi_2 : \Xi_2) \to (\xi_1 : \Xi_1)$ and as $\Pi\,\sigma$ or $\Pi(\xi_1 : \Xi_1, \xi_2 = \xi_2[\sigma]) : (\xi_1 : \Xi_1) \to (\xi_2 : \Xi_2)$. For example, in cubical type theory, we get $\Omega(i = 0) : (\Xi, i : \mathbb{I}) \to \Xi$ with right adjoint $\Pi(i = 0)$, and for semicartesian $\mathbb{U}$, we get $\Omega(u : \mathbb{U}) : \Xi \to (\Xi, u : \mathbb{U})$ with right adjoint $\Pi(u : \mathbb{U})$. The Π-modality is strictly functorial, whereas the Ω-modality is pseudofunctorial: we need explicit 2-cells witnessing $\Omega\,\tau \circ \Omega\,\sigma \cong \Omega(\tau \circ \sigma)$.[6] These modalities are adjoint internally by virtue of the 2-cells for the unit $\mathsf{const}_\sigma : 1 \Rightarrow \Pi\,\sigma \circ \Omega\,\sigma$ and co-unit $\mathsf{app}_\sigma : \Omega\,\sigma \circ \Pi\,\sigma \Rightarrow 1$. If σ introduces variables, then the codomain of the co-unit may be a variable renaming that is semantically the identity, e.g. $\mathsf{app}_{(v/u:\mathbb{U})} : \Omega(v : \mathbb{U}) \circ \Pi(u : \mathbb{U}) \Rightarrow 1(v : \mathbb{U}, u = v)$.

▶ **Example 4.5.** If $\mathbb{U}$ is cartesian, then there is a diagonal substitution $(w/u, w/v) : (\Xi, w : \mathbb{U}) \to (\Xi, u, v : \mathbb{U})$. Writing $\alpha = 1_{\Pi\,u} \star 1_{\Pi\,v} \star \mathsf{const}_{(w,u=w,v=w)}$, this allows us to implement the naïvely typed function $\lambda f.\lambda w.f\,w\,w : (\Pi\,u.\Pi\,v.A) \to \Pi\,w.A[w/u, w/v]$ as

$$\langle \Pi(u : \mathbb{U}) \mid^{\mathfrak{p}_u} \langle \Pi(v : \mathbb{U}) \mid^{\mathfrak{p}_v} A \rangle\rangle \to \langle \Pi(w : \mathbb{U}) \mid^{\mathfrak{p}_w} \langle \Omega(w : \mathbb{U}, u = w, v = w) \mid^{\mathfrak{o}} A[\alpha\downarrow^{\mathfrak{p}_u\mathfrak{p}_v}_{\mathfrak{p}_w\mathfrak{o}}]\rangle\rangle.$$

[4] Let RG^k be the base category of k-ary reflexive graphs: it has objects $\top$ and $\mathbb{I}$, and morphisms freely generated by $0, \ldots, k-1 : \top \to \mathbb{I}$ and terminality of $\top$. The category of k-ary affine (cartesian) cubes $\square^k$ ($\boxslash^k$) is the free (cartesian) monoidal category with same terminal object over RG^k. Moulin et al. [4, 24] use $\square^1$, Bezem et al. [6] use $\square^2$.

[5] By the second notation, we mean that we declare new variables (with their type if there is space) and write $u = t$ when we substitute t for u.

[6] This is because we defined the modality μ via $[\![$🔒$_\mu]\!]$ and only Ω is strictly functorial in the model. If we are willing to rely on a strictification conjecture by Gratzer et al. [17], then the internal modality Ω can be made strictly functorial too.

▶ Remark 4.6. The reframing of shape substitutions as a modality, has the annoying consequence that substitution no longer reduces. However, both $\langle \Omega\,\sigma \mid \sqcup \rangle$ and $\mathsf{mod}_{\Omega\,\sigma}$ are semantically an ordinary substitution.[7] Thus, we could add computation rules such as:

$$\begin{array}{rclrcl} \langle \Omega\,\sigma \mid^{\mathfrak{o}} A \times B \rangle & = & \langle \Omega\,\sigma \mid^{\mathfrak{o}} A \rangle \times \langle \Omega\,\sigma \mid^{\mathfrak{o}} B \rangle, & \langle \Omega\,\sigma \mid^{\mathfrak{o}} \mathsf{U} \rangle & = & \mathsf{U}, \\ \mathsf{mod}^{\mathfrak{o}}_{\Omega\,\sigma}\,(a, b) & = & (\mathsf{mod}^{\mathfrak{o}}_{\Omega\,\sigma}\,a, \mathsf{mod}^{\mathfrak{o}}_{\Omega\,\sigma}\,b), & \mathsf{mod}^{\mathfrak{o}}_{\Omega\,\sigma}\,A & = & \langle \Omega\,\sigma \mid^{\mathfrak{o}} A \rangle. \end{array}$$

This is fine in an extensional type system, but would not play well with the β-rule for modal types in an intensional system.

Modalities for (co)quantification The fresh weakening functor $\Finv_U : \mathcal{W} \to \mathcal{W}/U$ generalizes to a functor $\Finv_U^{/\Xi} : \mathcal{W}/\Xi \to \mathcal{W}/(\Xi \ltimes \mathbf{y}U)$ between categories of elements. Assuming that the multiplier is quantifiable,[8] there is a left adjoint $\exists_U^{/\Xi} \dashv \Finv_U^{/\Xi}$ [28]. These two give rise to four adjoint functors $\exists(u : \mathbb{U}) \dashv \Finv(u : \mathbb{U}) \dashv \forall(u : \mathbb{U}) \dashv \between(u : \mathbb{U})$ between the presheaf categories. The latter three give rise to DRAs and can be internalized as MTT modalities. We denote the units and co-units as

$$\begin{array}{ll} \mathsf{const}_{(u:\mathbb{U})} : 1 \Rightarrow \forall(u : \mathbb{U}) \circ \Finv(u : \mathbb{U}) & \mathsf{app}_{(u:\mathbb{U})} : \Finv(u : \mathbb{U}) \circ \forall(u : \mathbb{U}) \Rightarrow 1 \\ \mathsf{reidx}_{(u:\mathbb{U})} : 1 \Rightarrow \between(u : \mathbb{U}) \circ \forall(u : \mathbb{U}) & \mathsf{unmer}_{(u:\mathbb{U})} : \forall(u : \mathbb{U}) \circ \between(u : \mathbb{U}) \Rightarrow 1 \end{array}$$

Again, we also write $\mathsf{app}_{(v/u:\mathbb{U})} : \Finv(v : \mathbb{U}) \circ \forall(u : \mathbb{U}) \Rightarrow 1(v : \mathbb{U}, u = v)$ and $\mathsf{reidx}_{(v/u:\mathbb{U})} : 1(v : \mathbb{U}, u = v) \Rightarrow \between(v : \mathbb{U}) \circ \forall(u : \mathbb{U})$ to handle shape variable renamings that are semantically the identity.

▶ **Theorem 4.7** (Quantification). *[28] If the multiplier is*

- *cancellative and affine, then $\mathsf{const}_{(u:\mathbb{U})}$ and $\mathsf{unmer}_{(u:\mathbb{U})}$ are natural isomorphisms,*
- *semi-cartesian (so that $\Omega(u : \mathbb{U})$ and $\Pi(u : \mathbb{U})$ exist), then we have $\mathsf{spoil}_{(u:\mathbb{U})} : \Finv(u : \mathbb{U}) \Rightarrow \Omega(u : \mathbb{U})$ and $\mathsf{cospoil}_{(u:\mathbb{U})} : \Pi(u : \mathbb{U}) \Rightarrow \forall(u : \mathbb{U})$,*
- *cartesian, then we can soundly identify $\Finv(u : \mathbb{U}) = \Omega(u : \mathbb{U})$ and $\forall(u : \mathbb{U}) = \Pi(u : \mathbb{U})$.*

To understand the difference between $\Finv(u : \mathbb{U})$ and $\Omega(u : \mathbb{U})$, we can compare to the naïve system (Section 2) or Moulin et al.'s system [4, 24]. There, shape variables $u : \mathbb{U}$ are part of the context, and variables to the left of u are fresh for u. Here, shape variables are all in the shape context, but we use $\Finv\,u$ and $\Omega\,u$ (as well as the semantically identical $\text{🔒}_{\forall\,u}$ and $\text{🔒}_{\Pi\,u}$) to keep track of whether or not other variables should be fresh for u. Thus, in terms of the naïve system, $\forall(u : \mathbb{U})$ introduces u at the end of Γ (marking all variables fresh with $\text{🔒}_{\forall\,u}$), and $\Pi(u : \mathbb{U})$ introduces u in front of Γ (marking them non-fresh with $\text{🔒}_{\Pi\,u}$). The naïve system allows exchanging shape and other variables in one direction only, which is represented here by the generally non-invertible 2-cells spoil and cospoil. We emphasize that the word 'fresh' needs to be taken with a grain of salt: only for cancellative and affine $\mathbb{U}$ does invertibility of const_u guarantee that something fresh for u does not depend on u.

▶ **Example 4.8.** As an instance of Proposition 3.3, we obtain $\mathsf{prmod}_{\forall(v/u:\mathbb{U})} : (\Finv(v : \mathbb{U}) \mid^{\mathfrak{f}} \langle \forall(u : \mathbb{U}) \mid^{\mathfrak{a}} A \rangle) \to \left\langle 1(v : \mathbb{U}, u = v) \mid^{\bullet} A[\mathsf{app}_{v/u}{\downarrow}^{\mathfrak{fa}}_{\bullet}] \right\rangle$, which says that a non-cartesian function g with naïve type $g : \forall\,u.A$ can be applied to $v : \mathbb{U}$ provided that g is fresh for v.

▶ **Example 4.9** (Higher-dimensional pattern matching). As in Section 2, we create a function $f : \langle \forall(u : \mathbb{U}) \mid^{\mathfrak{a}} A \uplus B \rangle \to \langle \forall(u : \mathbb{U}) \mid^{\mathfrak{a}} A \rangle \uplus \langle \forall(u : \mathbb{U}) \mid^{\mathfrak{a}} B \rangle$, namely:

$$f\,\hat{c} = \mathsf{prmod}_{\between\,u} \cdot^{\mathfrak{a}} \mathsf{case}\,(\mathsf{prmod}_{\forall\,u} \cdot^{\mathfrak{o}} (\hat{c}\,\mathsf{const}_u{\downarrow}_{\mathfrak{ao}}))\,\mathsf{of} \left\{ \begin{array}{l} \mathsf{inl}\,a \mapsto \mathsf{mod}^{\mathfrak{t}}_{\between\,u}\,(\mathsf{inl}\,(\mathsf{mod}^{\mathfrak{a}'}_{\forall\,u}\,(a\,\mathsf{reidx}_u{\downarrow}_{\mathfrak{ta}'}))) \\ \mathsf{inr}\,b \mapsto \mathsf{mod}^{\mathfrak{t}}_{\between\,u}\,(\mathsf{inr}\,(\mathsf{mod}^{\mathfrak{a}'}_{\forall\,u}\,(b\,\mathsf{reidx}_u{\downarrow}_{\mathfrak{ta}'}))) \end{array} \right\}.$$

[7] Not along $\sigma : \Xi_1 \to \Xi_2$, but along $\sigma.\eta_{\Sigma\,\sigma \dashv \Omega\,\sigma} : \Xi_1.\Gamma \cong \Xi_2.\Sigma\,\sigma\,\Gamma$, which happens to be an isomorphism.
[8] We have not encountered any interesting examples where this is not the case.

We see that $\mathsf{mod}_{\forall u}$ and $\mathsf{prmod}_{\forall u}$ correspond to shape abstraction and application in the naïve system, $\mathsf{prmod}_{\between u}$ to unmer, and $\mathsf{mod}_{\between u}$ to mer. The annotation const_u is an explicit weakening over $u : \mathbb{U}$, whereas reidx_u replaces an explicit reindexing $|_{u=u}$.

4.2 Additional Typing Rules

In this section, we add a few extensions to MTT in order to reason about boundaries (Definition 4.3) in the *type* theory, rather than in the shape theory, and in order to recover all known presheaf operators in Section 5.

Subobject classifier We add a universe of propositions (semantically the subobject classifier) $\mathsf{Prop} : \mathsf{U}_0$, with implicit encoding and decoding operations à la Coquand. This is necessary to talk about Ψ and Φ. We identify all proofs of the same proposition.

Boundary predicate We add a predicate $\Xi, u : \mathbb{U} \mid \cdot \vdash (u \in \partial\mathbb{U}) : \mathsf{Prop}$ corresponding in the model to the subobject $(\Xi, u : \partial\mathbb{U}) \subseteq (\Xi, u : \mathbb{U})$. A naïve introduction rule would be $\Xi, u : \partial\mathbb{U} \mid \cdot \vdash \_ : \langle \Omega(u \in \partial\mathbb{U}) \mid^{\circ} (u \in \partial\mathbb{U})\rangle$. As all our modalities are proper DRAs [8] as opposed to the weaker concepts required by the general model of MTT, the modal introduction rule is invertible in the model, so we may as well take $\Xi, u : \mathbb{U} \mid \cdot, \text{🔒}^{\circ}_{\Omega(u\in\partial\mathbb{U})} \vdash \mathsf{on}\partial \downarrow_{\circ} : (u \in \partial\mathbb{U})$ as an introduction rule. If we absorb a substitution into these rules, we get

$$\frac{\Xi, u : \mathbb{U} \mid \Gamma\ \mathsf{ctx}}{\Xi, u : \mathbb{U} \mid \Gamma \vdash (u \in \partial\mathbb{U})\ \mathsf{prop}}, \qquad \frac{\Xi, u : \mathbb{U} \mid \Gamma, \Delta\ \mathsf{ctx} \qquad \alpha : \Omega(u \in \partial\mathbb{U}) \Rightarrow \mathsf{locks}(\Delta)}{\Xi, u : \mathbb{U} \mid \Gamma, \Delta \vdash \mathsf{on}\partial\, \alpha\!\downarrow_{\mathsf{ticks}(\Delta)} : (u \in \partial\mathbb{U})}.$$

An elimination rule could build a function $(\_ : u \in \partial\mathbb{U}) \to A\_$ from $\Pi(u \in \partial\mathbb{U}) \circ \Omega(u \in \partial\mathbb{U})$ applied to A, but it is not clear how to formulate the β-rule. Instead, we will eliminate to the transpension type (Theorem 4.10).

Strictness axiom The strictness axiom [34] allows to extend a partial type T to a total type if T is isomorphic to a total type A, effectively strictifying the isomorphism:

$$\frac{\Xi \mid \Gamma \vdash \varphi : \mathsf{Prop} \qquad \Xi \mid \Gamma \vdash A : \mathsf{U}_\ell \qquad \Xi \mid \Gamma, \_ : \varphi \vdash T : \mathsf{U}_\ell \qquad \Xi \mid \Gamma, \_ : \varphi \vdash i : A \cong T}{\begin{array}{l}\Xi \mid \Gamma \vdash \mathsf{Strict}\{A \cong (\varphi \mathrel{?} T \mathrel{;} i)\} : \mathsf{U}_\ell \qquad \Xi \mid \Gamma \vdash \mathsf{strict}\{\varphi \mathrel{?} i\} : A \cong \mathsf{Strict}\{A \cong (\varphi \mathrel{?} T \mathrel{;} i)\} \\ \text{where } \Gamma, \_ : \varphi \vdash \mathsf{Strict}\{A \cong (\varphi \mathrel{?} T \mathrel{;} i)\} = T : \mathsf{U}_\ell \qquad \Gamma, \_ : \varphi \vdash \mathsf{strict}\{\varphi \mathrel{?} i\} = i : A \cong T\end{array}}$$

4.3 Investigating the Transpension Type

In this section, we investigate the structure of the transpension type.

Poles Our first observation is that on the boundary, the transpension type is trivial. Let $\top : \Xi_1 \to \Xi_2$ be the modality, between any two shape contexts, which maps any presheaf to the terminal presheaf. We clearly have $\top \circ \mu = \top$ for any μ, but also $\mu \circ \top \cong \top$ because all internal modalities are right adjoints and therefore preserve the terminal object.

▶ **Theorem 4.10.** *We have $\Omega(u \in \partial\mathbb{U}) \circ \between(u : \mathbb{U}) \cong \top$. We can thus postulate a term $(u \in \partial\mathbb{U}) \vdash \mathsf{pole} : \langle \between(u : \mathbb{U}) \mid^{\mathsf{t}} T\rangle$ for any T, with an η-rule $(u \in \partial\mathbb{U}) \vdash t = \mathsf{pole} : \langle \between(u : \mathbb{U}) \mid^{\mathsf{t}} T\rangle$.*

Sketch of proof. The left adjoints $\forall(u : \mathbb{U}) \circ \Sigma(u \in \partial\mathbb{U})$ and $\bot$ are isomorphic because $\forall(u : \mathbb{U}).(u \in \partial\mathbb{U})$ is false. We give a full proof in the technical report [28]. ◀

Definition 4.3 of the boundary relied on the notion of dimensional splitness. The following result shows that it was a good one: the transpension is *only* trivial on the boundary:

▶ **Theorem 4.11.** *In the model, we have $\cdot, u : \mathbb{U} \mid \Gamma \vdash (u \in \partial\mathbb{U}) \cong \langle \between(u : \mathbb{U}) \mid^{\mathsf{t}} \mathsf{Empty}\rangle$. [28]*

Meridians As all our modalities are proper DRAs [8], the modal introduction rule is invertible in the model. This immediately shows that sections[9] of the transpension type $\Xi \mid \Gamma \vdash f : \langle \forall(u : \mathbb{U}) \mid^{\mathfrak{a}} \langle \between(u : \mathbb{U}) \mid^{\mathfrak{t}} T \rangle\rangle$ (which we call meridians) are in 1-1 correspondence with terms $\Xi \mid \Gamma, 🔒^{\mathfrak{a}}_{\forall(u:\mathbb{U})}, 🔒^{\mathfrak{t}}_{\between(u:\mathbb{U})} \vdash t : T$. If it were not for the locking of the context, this characterization in terms of poles and meridians would make the transpension type look quite similar to a dependent version of the suspension type in HoTT [39], whence our choice of name. If $\mathbb{U}$ is cancellative and affine, then the locks can actually be ignored (Theorem 4.7). In any case, Proposition 3.3 tells us that the let-rule for $\between(u : \mathbb{U})$ has the same power as $\mathsf{prmod}_{\between(u:\mathbb{U})} : (\forall(u : \mathbb{U}) \shortmid^{\mathfrak{a}} \langle \between(u : \mathbb{U}) \mid^{\mathfrak{t}} T \rangle) \to T[\mathsf{unmer}_u \downarrow^{\mathfrak{at}}_{\bullet}]$ (unmer in Section 2) which extracts meridians. If $\mathbb{U}$ is cancellative and affine, then unmer_u is invertible (Theorem 4.7) and we can also straightforwardly *create* meridians from elements of $T[\mathsf{unmer}_u \downarrow^{\mathfrak{at}}_{\bullet}]$.

Pattern matching The eliminator $\mathsf{prmod}_{\between(u:\mathbb{U})}$ is only capable of eliminating *sections* of the transpension type. Sometimes we can eliminate locally by pattern matching:

▶ **Theorem 4.12.** *If $\mathbb{U}$ is cancellative, affine, connection-free and quantifiable, then the following rule is sound [28]:*

$$\frac{\begin{array}{l}\Xi, u : \mathbb{U} \mid \Gamma\ \mathsf{ctx} \qquad \Xi \mid \Gamma, 🔒^{\mathfrak{t}}_{\between_u} \vdash A\ \mathsf{type} \qquad \Xi, u : \mathbb{U} \mid \Gamma, r : \left\langle \between u \mid^{\mathfrak{t}} A \right\rangle \vdash C\ \mathsf{type} \\ \Xi, u : \mathbb{U} \mid \Gamma, \_ : (u \in \partial\mathbb{U}) \vdash c_\partial : C[\mathsf{pole}/r] \qquad \Xi, u : \mathbb{U} \mid \Gamma \vdash t : \left\langle \between_u \mid^{\mathfrak{t}} A \right\rangle \\ \Xi, u : \mathbb{U} \mid \Gamma, 🔒^{\mathfrak{t}}_{\between u}, x : A, 🔒^{\mathfrak{a}}_{\forall u} \vdash c : C[\mathsf{reidx}_u \downarrow^{\bullet}_{\mathfrak{ta}}]\left[\mathsf{mod}^{\mathfrak{t}'}_{\between u}\,(x\ \mathsf{unmer}^{-1}_u \downarrow_{\mathfrak{at}'})/r\right] \\ \Xi, u : \mathbb{U} \mid \Gamma, 🔒^{\mathfrak{t}}_{\between u}, x : A, 🔒^{\mathfrak{a}}_{\forall u}, \_ : (u \in \partial\mathbb{U}) \vdash c = c_\partial[\mathsf{reidx}_u \downarrow^{\bullet}_{\mathfrak{ta}}] : C[\mathsf{reidx}_u \downarrow^{\bullet}_{\mathfrak{ta}}, \mathsf{pole}/r]\end{array}}{\begin{array}{l}\Xi, u : \mathbb{U} \mid \Gamma \vdash c := \mathsf{case}\ t\ \mathsf{of}\left\{\ \mathsf{pole} \mapsto c_\partial \mid \mathsf{merid}(\mathfrak{t}, x, \mathfrak{a}) \mapsto c\ \right\} : C[t/r] \\ \text{where } c[\mathsf{pole}/t] = c_\partial \\ \qquad\qquad \Delta, 🔒^{\mathfrak{a}'}_{\forall u} \vdash c = c[\mathsf{prmod}_{\between u} \cdot^{\mathfrak{a}} (t[\mathsf{reidx}_u \downarrow^{\bullet}_{\mathfrak{ta}}])/x, \downarrow^{\mathfrak{a}}_{\mathfrak{a}}][(\mathsf{unmer}_u \star 1_{\forall u}) \downarrow^{\mathfrak{a}'}_{\mathfrak{a}'\mathfrak{ta}}]\end{array}}$$

We get a context Γ depending on $u : \mathbb{U}$, a type A depending on sections of Γ (recall that $🔒_{\between u}$ is semantically $\forall u$), a type C depending on u and $\langle \between u \mid A \rangle$, and an argument of type $\langle \between u \mid A \rangle$. To obtain a value of type C, we need to give an action c_∂ on pole, i.e. when we are on the boundary, and compatibly map meridians (i.e. elements of A) to sections of C (the quantifier has been brought to the left as $🔒_{\forall u}$). The computation rule for meridians is in a non-general context and needs to be forcibly closed under substitution.

5 Recovering Known Operators

In this section, we explain how to recover the amazing right adjoint $\surd$ [23], Moulin et al.'s Φ and Ψ combinators [4, 24] and Glue [14, 32], Weld [32] and mill [30] from the transpension, the strictness axiom [34] and certain pushouts.

The amazing right adjoint $\surd$ Licata et al. [23] use presheaves over a cartesian base category of cubes and introduce $\surd$ as the right adjoint to the non-dependent exponential $\mathbb{I} \to \sqcup$. We generalize to semicartesian systems and look for a right adjoint to $\mathbb{U} \multimap \sqcup$, which decomposes as substructural quantification after cartesian weakening $\forall(u : \mathbb{U}) \circ \Omega(u : \mathbb{U})$. Then the right adjoint is obviously $\surd_{\mathbb{U}} := \Pi(u : \mathbb{U}) \circ \between(u : \mathbb{U})$. The type constructor has type $\langle \surd_{\mathbb{U}} \mid \sqcup \rangle : (\surd_{\mathbb{U}} \shortmid^{\mathfrak{r}} \mathsf{U}_\ell) \to \mathsf{U}_\ell$ and the transposition rule is as in Proposition 3.4. This is an improvement in two ways: First, we have computation rules, so that we do not need to postulate functoriality of $\surd_{\mathbb{U}}$ and invertibility of transposition. Secondly, we have no need

[9] By a section of a dependent type, we mean a dependent function with the same domain as the type.

$$\frac{\begin{array}{l}\Xi, u : \mathbb{U} \mid \Gamma \vdash B \,\mathsf{type} \qquad \Xi, u : \mathbb{U} \mid \Gamma, y : B \vdash C \,\mathsf{type} \qquad \Xi, u : \mathbb{U} \mid \Gamma \vdash b : B \\ \Xi, u : \mathbb{U} \mid \Gamma, y : B, \_ : (u \in \partial\mathbb{U}) \vdash c_\partial : C \\ \Xi, u : \mathbb{U} \mid \Gamma, y : B, \text{🔒}^{\mathfrak{t}}_{\between u}, \text{🔒}^{\mathfrak{a}}_{\forall u} \vdash c : C[\mathsf{reidx}_u {\downarrow}^{\bullet}_{\mathfrak{t}\,\mathfrak{a}}] \\ \Xi, u : \mathbb{U} \mid \Gamma, y : B, \text{🔒}^{\mathfrak{t}}_{\between u}, \text{🔒}^{\mathfrak{a}}_{\forall u}, \_ : (u \in \partial\mathbb{U}) \vdash c = c_\partial[\mathsf{reidx}_u {\downarrow}^{\bullet}_{\mathfrak{t}\,\mathfrak{a}}] : C[\mathsf{reidx}_u {\downarrow}^{\bullet}_{\mathfrak{t}\,\mathfrak{a}}]\end{array}}{\begin{array}{l}\Xi, u : \mathbb{U} \mid \Gamma \vdash \Phi_u\,(y.c_\partial)\,(y.\mathfrak{t}.\mathfrak{a}.c)\,b : C[b/y] \\ \text{where } \Gamma, \_ : (u \in \partial\mathbb{U}) \vdash \Phi_u\,(y.c_\partial)\,(y.\mathfrak{t}.\mathfrak{a}.c)\,b = c_\partial[b/y] \\ \qquad \Delta, \text{🔒}^{\mathfrak{a}'}_{\forall u} \vdash (\Phi_u\,(y.c_\partial)\,(y.\mathfrak{t}.\mathfrak{a}.c)\,b) = c[b/y, {\downarrow}^{\mathfrak{t}\,\mathfrak{a}}_{\mathfrak{t}\,\mathfrak{a}}][(\mathsf{unmer}_u \star 1_{\forall u}){\downarrow}^{\mathfrak{a}'\,\mathfrak{t}\,\mathfrak{a}}_{\mathfrak{a}'}]\end{array}}$$

Figure 4 The Φ-rule for $(y : B) \to C$ (sound if $\mathbb{U}$ is cancellative, affine and connection-free).

$$\frac{\begin{array}{l}\Xi, u : \mathbb{U} \mid \Gamma, \_ : (u \in \partial\mathbb{U}) \vdash A \,\mathsf{type} \\ \Xi \mid \Gamma, \alpha : (\_ : u \in \partial\mathbb{U}) \to A, \text{🔒}^{\mathfrak{t}}_{\between(u:\mathbb{U})} \vdash R \,\mathsf{type}\end{array}}{\begin{array}{l}\Xi, u : \mathbb{U} \mid \Gamma \vdash \Psi_u\,A\,(\alpha.\mathfrak{t}.R) \,\mathsf{type} \\ \text{where } \_ : (u \in \partial\mathbb{U}) \vdash \Psi_u\,A\,(\alpha.\mathfrak{t}.R) = A\end{array}} \qquad \frac{\begin{array}{l}\Xi, u : \mathbb{U} \mid \Gamma, \_ : (u \in \partial\mathbb{U}) \vdash a : A \\ \Xi \mid \Gamma, \text{🔒}^{\mathfrak{t}}_{\between(u:\mathbb{U})} \vdash r : R[\lambda\_.a/\alpha]\end{array}}{\begin{array}{l}\Xi, u : \mathbb{U} \mid \Gamma \vdash \mathsf{in}\Psi_u\,a\,(\mathfrak{t}.r) : \Psi_u\,A\,(\alpha.\mathfrak{t}.R) \\ \text{where } \_ : (u \in \partial\mathbb{U}) \vdash \mathsf{in}\Psi_u\,a\,(\mathfrak{t}.r) = a\end{array}}$$

$$\begin{array}{l}\Xi \mid \Delta \vdash \mathsf{out}\Psi : (\forall(u : \mathbb{U}) \shortmid x :^{\mathfrak{a}} \Psi_u\,A\,(\alpha.\mathfrak{t}.R)) \to R[\lambda\_.x{\downarrow}_{\mathfrak{a}}/\alpha, {\downarrow}^{\mathfrak{t}}_{\mathfrak{t}}][\mathsf{unmer}_u {\downarrow}^{\mathfrak{a}\,\mathfrak{t}}_{\bullet}] \\ \text{where } \mathsf{out}\Psi \cdot^{\mathfrak{a}} \mathsf{in}\Psi_u\,a\,(\mathfrak{t}.r) = r[\mathsf{unmer}_u {\downarrow}^{\mathfrak{a}\,\mathfrak{t}}_{\bullet}] \qquad \mathsf{in}\Psi_u\,t\,(\mathfrak{t}.\mathsf{out}\Psi \cdot^{\mathfrak{a}} t[\mathsf{reidx}_u {\downarrow}^{\bullet}_{\mathfrak{t}\,\mathfrak{a}}]) = t\end{array}$$

Figure 5 Typing rules for the Ψ-type

for a global sections modality $\flat$. Our very general mode theory does contain $\flat : \top \to \top$, and Licata et al.'s formation and functoriality axioms can be proven because $\flat \Rightarrow \sqrt{}_{\mathbb{U}}$. Their transposition rule is provable by applying $\langle \flat \mid \sqcup \rangle$ to both sides of the isomorphism in Proposition 3.4 and using that $\flat \circ \sqrt{}_{\mathbb{U}} \cong \flat$, which follows from an isomorphism between their left adjoints $(\mathbb{I} \multimap \sqcup) \circ \int \cong \int$, where $\int \dashv \flat$ is the connected components functor.

The Φ-combinator In Figure 4, we *state* Moulin et al.'s Φ-rule [4, 24] adapted to the current system. For types B and C (depending on $u : \mathbb{U}$), the combinator allows us to define functions of naïve type $\forall u.(y : B\,u) \to C\,u\,y$ from an action c_∂ on the boundary (in Moulin et al.'s work: on every endpoint of the interval) and a compatible action c on sections $\forall u.B\,u$. Given a section of B (recall that $\text{🔒}_{\between u}$ is semantically $\forall u$), c provides a section of C (but the quantification has been moved to the left as $\text{🔒}_{\forall u}$), compatible with c_∂ on the boundary. Again, the computation rule for sections is in a non-general context and needs to be forcibly closed under substitution.

The main difference with Moulin et al.'s formulation is that they require that $i : \mathbb{I}$ be the last variable before y, i.e. it comes after Γ. Here, this would mean that Γ is fresh for $u : \mathbb{U}$, i.e. $\Gamma = (\Delta, \text{🔒}_{\forall u})$. In that case, by Proposition 3.4 and Theorem 4.7 and the fact that Moulin et al.'s system is cancellative and affine, the context of c is essentially $\Delta, \forall u \shortmid y :^{\mathfrak{a}} B[\ldots], \text{🔒}^{\mathfrak{a}}_{\forall u}$, which corresponds more closely to Moulin et al.'s rules. The greater generality of our Φ-rule means that there is in fact no reason not to absorb B into Γ.

We remark that if C is a transpension and $\mathbb{U}$ is cancellative and affine, then the Φ-rule holds by Theorems 4.7 and 4.10.

▶ **Theorem 5.1.** *If $\mathbb{U}$ is cancellative, affine, connection-free and quantifiable, then the Φ-rule (Figure 4) is derivable for all B and C. [28]*

Proof. Absorb B into Γ and use the case-eliminator for $\langle \between u \mid^{\mathfrak{t}} \mathsf{Unit} \rangle$ (Theorem 4.12). ◀

The Ψ-combinator Moulin et al.'s Ψ-combinator constructs an edge in the universe with endpoints A_ϵ from a relation $R : \bigtimes_\epsilon A_\epsilon \to \mathsf{U}$. In Figure 5, we adapt the typing rules, replacing the concept of endpoints with the boundary. The formation rule takes a type A

that exists on the boundary, and a type R depending on sections of A (with 🔒$_{\between u}$ being $\forall u$). It yields a Ψ-type which equals A on the boundary. The introduction rule is only necessary when we are not on the boundary (otherwise we can just coerce elements of A). It requires an element of A on the boundary and, for every section of Γ (which need not exist), a proof that the resulting boundary section satisfies R. The elimination rule sends sections of the Ψ-type to proofs that the boundary part satisfies R. We have β- and η-rules. In cancellative, affine and connection-free systems, the Φ-rule yields a pattern-matching eliminator. Again, the main difference with Moulin et al. is that Γ need not be fresh for u.

The Ψ-type can be implemented by strictifying the following using Strict:

$$\Psi_u\, A\, (\alpha.\mathfrak{t}.R) :\cong (\alpha : (\_ : u \in \partial U) \to A) \times \langle \between u \mid^{\mathfrak{t}} R \rangle,$$

The fact that it is isomorphic to A on the boundary follows from Theorem 4.10.

Transpensivity The Φ-rule is extremely powerful but not available in all systems. However, when the codomain C is a Ψ-type, then the $\mathsf{in}\Psi$-rule is actually quite similar to the Φ-rule. As such, we take an interest in types that are very Ψ-like. We have a monad (idempotent if $\mathbb{U}$ is cancellative and affine) $\bar{\Psi}_u A := \Psi_u\, A\, (\alpha.\mathfrak{t}.\langle \forall u \mid^{\mathfrak{a}} A[u \in \partial\mathbb{U}\ ?\ \alpha\,\_][\mathsf{reidx}_u{\downarrow}^{\bullet}_{\mathfrak{ta}}] \rangle)$, where $A[\varphi\ ?\ a]$ is the type of elements of A that are equal to a when φ holds.

▶ **Definition 5.2.** *A type is* ***transpensive over*** u *if it is a monad-algebra for* $\bar{\Psi}_u$.

For cancellative, affine, connection-free, quantifiable multipliers, Φ entails that all types are transpensive. For other systems, many interesting types will still be transpensive.

Glue, Weld, mill $\mathsf{Glue}\{A \leftarrow (\varphi\ ?\ T\ ;\ f)\}$ and $\mathsf{Weld}\{A \to (\varphi\ ?\ T\ ;\ g)\}$ are similar to Strict but extend unidirectional functions. Orton and Pitts [34] already show that Glue [14, 32] can be implemented by strictifying a pullback along $A \to (\varphi \to A)$ [30] which is definable internally using a Σ-type. Dually, Weld [32] can be implemented if there is a type former for pushouts along $\varphi \times A \to A$ where $\varphi : \mathsf{Prop}$ [30], which is sound in all presheaf categories. Finally, mill [30] states that $\forall (u : \mathbb{U})$ preserves Weld and is provable by higher-dimensional pattern matching.

6 Conclusion

To summarize, the transpension type can be defined in a broad class of presheaf models and generalizes previous internalization operators. For now, we only present an extensional type system without an algorithmic typing judgement. The major hurdles towards producing an intensional version with decidable type-checking, are the following:

- We need to decide equality of 2-cells. Solutions may exist in the literature on higher-dimensional rewriting.
- If we want the substitution modality to reduce (Remark 4.6), we need to solve the following problem: when $\hat{a} = \mathsf{mod}^{\mathfrak{o}}_{\Omega\,\sigma}\, a$ definitionally, then we need to infer a up to definitional equality from $\hat{a}$ in order to β-reduce the let-rule for $\langle \Omega\,\sigma \mid A \rangle$.
- We need a syntax-directed way to close the section computation rules of Φ (Figure 4) and transpension elimination (Section 4.3) under substitution.
- We need to decide whether a proposition is true. This problem has been dealt with in special cases, e.g. in implementations of cubical type theory [40].

References

1 Carlo Angiuli, Kuen-Bang Hou (Favonia), and Robert Harper. Cartesian Cubical Computational Type Theory: Constructive Reasoning with Paths and Equalities. In Dan Ghica and Achim Jung, editors, *27th EACSL Annual Conference on Computer Science Logic (CSL 2018)*, volume 119 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 6:1–6:17, Dagstuhl, Germany, 2018. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. URL: `http://drops.dagstuhl.de/opus/volltexte/2018/9673`, `doi:10.4230/LIPIcs.CSL.2018.6`.

2 Robert Atkey, Neil Ghani, and Patricia Johann. A relationally parametric model of dependent type theory. In *Principles of Programming Languages*, 2014. `doi:10.1145/2535838.2535852`.

3 Patrick Bahr, Hans Bugge Grathwohl, and Rasmus Ejlers Møgelberg. The clocks are ticking: No more delays! In *32nd Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2017, Reykjavik, Iceland, June 20-23, 2017*, pages 1–12, 2017. URL: `https://doi.org/10.1109/LICS.2017.8005097`, `doi:10.1109/LICS.2017.8005097`.

4 Jean-Philippe Bernardy, Thierry Coquand, and Guilhem Moulin. A presheaf model of parametric type theory. *Electron. Notes in Theor. Comput. Sci.*, 319:67 – 82, 2015. `doi:http://dx.doi.org/10.1016/j.entcs.2015.12.006`.

5 Jean-Philippe Bernardy and Andrea Vezzosi. Parametric application. Private communication, 2017.

6 Marc Bezem, Thierry Coquand, and Simon Huber. A Model of Type Theory in Cubical Sets. In *19th International Conference on Types for Proofs and Programs (TYPES 2013)*, volume 26, pages 107–128, Dagstuhl, Germany, 2014. URL: `http://drops.dagstuhl.de/opus/volltexte/2014/4628`, `doi:10.4230/LIPIcs.TYPES.2013.107`.

7 Lars Birkedal, Aleš Bizjak, Ranald Clouston, Hans Bugge Grathwohl, Bas Spitters, and Andrea Vezzosi. Guarded cubical type theory. *Journal of Automated Reasoning*, 63(2):211–253, Aug 2019. URL: `https://doi.org/10.1007/s10817-018-9471-7`, `doi:10.1007/s10817-018-9471-7`.

8 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*, page 1–21. `doi:10.1017/S0960129519000197`.

9 Aleš Bizjak, Hans Bugge Grathwohl, Ranald Clouston, Rasmus Ejlers Møgelberg, and Lars Birkedal. Guarded dependent type theory with coinductive types. In *FOSSACS '16*, 2016. `doi:10.1007/978-3-662-49630-5\_2`.

10 Aleš Bizjak and Rasmus Ejlers Møgelberg. Denotational semantics for guarded dependent type theory. *CoRR*, abs/1802.03744, 2018. URL: `http://arxiv.org/abs/1802.03744`, `arXiv:1802.03744`.

11 John Cartmell. *Generalised Algebraic Theories and Contextual Categories.* PhD thesis, 1978.

12 Evan Cavallo and Robert Harper. Internal parametricity for cubical type theory. In *28th EACSL Annual Conference on Computer Science Logic, CSL 2020, January 13-16, 2020, Barcelona, Spain*, pages 13:1–13:17, 2020. URL: `https://doi.org/10.4230/LIPIcs.CSL.2020.13`, `doi:10.4230/LIPIcs.CSL.2020.13`.

13 Evan Cavallo, Anders Mörtberg, and Andrew W Swan. Unifying Cubical Models of Univalent Type Theory. In Maribel Fernández and Anca Muscholl, editors, *28th EACSL Annual Conference on Computer Science Logic (CSL 2020)*, volume 152 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 14:1–14:17, Dagstuhl, Germany, 2020. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. URL: `https://drops.dagstuhl.de/opus/volltexte/2020/11657`, `doi:10.4230/LIPIcs.CSL.2020.14`.

14 Cyril Cohen, Thierry Coquand, Simon Huber, and Anders Mörtberg. Cubical type theory: A constructive interpretation of the univalence axiom. *FLAP*, 4(10):3127–3170, 2017. URL: `http://collegepublications.co.uk/ifcolog/?00019`.

15 Jean Giraud. Méthode de la descente. *Bull. Soc. Math. Fr., Suppl., Mém.*, 2:115, 1964. `doi:10.24033/msmf.2`.

16 Daniel Gratzer, Alex Kavvos, Andreas Nuyts, and Lars Birkedal. Multimodal dependent type theory. 2020. Submitted to LICS'20. URL: `https://people.cs.kuleuven.be/~andreas.nuyts/mtt-paper.pdf`.

17 Daniel Gratzer, Alex Kavvos, Andreas Nuyts, and Lars Birkedal. Type theory à la mode. Unpublished, 2020. URL: `https://people.cs.kuleuven.be/~andreas.nuyts/mtt-techreport.pdf`.

18 Martin Hofmann. *Syntax and Semantics of Dependent Types*, chapter 4, pages 79–130. Cambridge University Press, 1997.

19 Martin Hofmann and Thomas Streicher. Lifting grothendieck universes. Unpublished note, 1997.

20 Simon Huber. A model of type theory in cubical sets. Licentiate's thesis, University of Gothenburg, Sweden, 2015.

21 Chris Kapulkin, Peter LeFanu Lumsdaine, and Vladimir Voevodsky. The simplicial model of univalent foundations. 2012. Preprint, `http://arxiv.org/abs/1211.2851`.

22 Daniel R. Licata and Robert Harper. 2-dimensional directed type theory. *Electr. Notes Theor. Comput. Sci.*, 276:263–289, 2011. URL: `https://doi.org/10.1016/j.entcs.2011.09.026`, `doi:10.1016/j.entcs.2011.09.026`.

23 Daniel R. Licata, Ian Orton, Andrew M. Pitts, and Bas Spitters. Internal universes in models of homotopy type theory. In *3rd International Conference on Formal Structures for Computation and Deduction, FSCD 2018, July 9-12, 2018, Oxford, UK*, pages 22:1–22:17, 2018. URL: `https://doi.org/10.4230/LIPIcs.FSCD.2018.22`, `doi:10.4230/LIPIcs.FSCD.2018.22`.

24 Guilhem Moulin. *Internalizing Parametricity.* PhD thesis, Chalmers University of Technology, Sweden, 2016.

25 Paige Randall North. Towards a directed homotopy type theory. *Proceedings of the Thirty-Fifth Conference on the Mathematical Foundations of Programming Semantics, MFPS 2019, London, UK, June 4-7, 2019*, pages 223–239, 2019. URL: `https://doi.org/10.1016/j.entcs.2019.09.012`, `doi:10.1016/j.entcs.2019.09.012`.

26 Andreas Nuyts. Presheaf models of relational modalities in dependent type theory. *CoRR*, abs/1805.08684, 2018. `arXiv:1805.08684`.

27 Andreas Nuyts. Robust notions of contextual fibrancy. In *Workshop on Homotopy Type Theory / Univalent Foundations*, 2018. URL: `https://hott-uf.github.io/2018/abstracts/HoTTUF18_paper_2.pdf`.

28 Andreas Nuyts. The transpension type: Technical report. Unpublished, 2020. URL: `https://people.cs.kuleuven.be/~andreas.nuyts/transpension-techreport.pdf`.

29 Andreas Nuyts and Dominique Devriese. Degrees of relatedness: A unified framework for parametricity, irrelevance, ad hoc polymorphism, intersections, unions and algebra in dependent type theory. In *Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2018, Oxford, UK, July 09-12, 2018*, pages 779–788, 2018. URL: `https://doi.org/10.1145/3209108.3209119`, `doi:10.1145/3209108.3209119`.

30 Andreas Nuyts and Dominique Devriese. Internalizing Presheaf Semantics: Charting the Design Space. In *Workshop on Homotopy Type Theory / Univalent Foundations*, 2018. URL: `https://hott-uf.github.io/2018/abstracts/HoTTUF18_paper_1.pdf`.

31 Andreas Nuyts and Dominique Devriese. Dependable atomicity in type theory. In *TYPES*, 2019.

32 Andreas Nuyts, Andrea Vezzosi, and Dominique Devriese. Parametric quantifiers for dependent type theory. *PACMPL*, 1(ICFP):32:1–32:29, 2017. URL: `http://doi.acm.org/10.1145/3110276`, `doi:10.1145/3110276`.

33 Ian Orton. *Cubical Models of Homotopy Type Theory - An Internal Approach.* PhD thesis, University of Cambridge, 2018.

34 Ian Orton and Andrew M. Pitts. Axioms for modelling cubical type theory in a topos. *Logical Methods in Computer Science*, 14(4), 2018. URL: `https://doi.org/10.23638/LMCS-14(4:23)2018`, `doi:10.23638/LMCS-14(4:23)2018`.

35 Gun Pinyo and Nicolai Kraus. From cubes to twisted cubes via graph morphisms in type theory. *CoRR*, abs/1902.10820, 2019. URL: `http://arxiv.org/abs/1902.10820`, `arXiv:1902.10820`.

36 John C. Reynolds. Types, abstraction and parametric polymorphism. In *IFIP Congress*, pages 513–523, 1983.

37 E. Riehl and M. Shulman. A type theory for synthetic ∞-categories. *ArXiv e-prints*, May 2017. `arXiv:1705.07442`.

38 The Stacks Project Authors. Stacks project. `http://stacks.math.columbia.edu`, 2019. Tags 00VC and 00XF.

39 The Univalent Foundations Program. *Homotopy Type Theory: Univalent Foundations of Mathematics*. `http://homotopytypetheory.org/book`, IAS, 2013.

40 Andrea Vezzosi, Anders Mörtberg, and Andreas Abel. Cubical agda: a dependently typed programming language with univalence and higher inductive types. *PACMPL*, 3(ICFP):87:1–87:29, 2019. URL: `https://doi.org/10.1145/3341691`, `doi:10.1145/3341691`.