

Internal Parametricity, without an Interval

THORSTEN ALTENKIRCH, University of Nottingham, UK

YORG CHAMOUN, École Polytechnique, France

AMBRUS KAPSI, Eötvös Loránd University, Hungary

MICHAEL SHULMAN, University of San Diego, USA

Parametricity is a property of the syntax of type theory implying, e.g., that there is only one function having the type of the polymorphic identity function. Parametricity is usually proven externally, and does not hold internally. Internalising it is difficult because once there is a term witnessing parametricity, it also has to be parametric itself and this results in the appearance of higher dimensional cubes. In previous theories with internal parametricity, either an explicit syntax for higher cubes is present or the theory is extended with a new sort for the interval. In this paper we present a type theory with internal parametricity which is a simple extension of Martin-Löf type theory: there are a few new type formers, term formers and equations. Geometry is not explicit in this syntax, but emergent: the new operations and equations only refer to objects up to dimension 3. We show that this theory is modelled by presheaves over the BCH cube category. Fibrancy conditions are not needed because we use span-based rather than relational parametricity. We define a gluing model for this theory implying that external parametricity and canonicity hold. The theory can be seen as a special case of a new kind of modal type theory, and it is the simplest setting in which the computational properties of higher observational type theory can be demonstrated.

CCS Concepts: • Theory of computation → Type theory.

Additional Key Words and Phrases: homotopy type theory, parametricity, logical relations, gluing

ACM Reference Format:

Thorsten Altenkirch, Yorgo Chamoun, Ambrus Kapsi, and Michael Shulman. 2024. Internal Parametricity, without an Interval. *Proc. ACM Program. Lang.* 8, POPL, Article 78 (January 2024), 30 pages. <https://doi.org/10.1145/3632920>

1 INTRODUCTION

Parametricity was introduced by Reynolds [Reynolds 1983] as a theory of representation-independence for the polymorphic lambda calculus. The idea is that a polymorphic function has to work uniformly on all types, i.e., it cannot inspect its type arguments, and thus for example there are zero, one and two terms of types $\forall a.a$, $\forall a.a \rightarrow a$ and $\forall a.a \rightarrow a \rightarrow a$, respectively. This intuition is formalised by relation-preservation: each type is equipped with a relation (called logical relation), and one can prove by induction on the syntax that every term respects the relation corresponding to its type (called the fundamental lemma). Dependent types are expressive enough that they can formulate their own parametricity relations. This was used by [Bernardy et al. 2010] to define a parametricity translation for type theory. We describe this translation below. We expect that the reader is familiar with the syntax of type theory.

Authors' addresses: Thorsten Altenkirch, University of Nottingham, , UK, thorsten.altenkirch@nottingham.ac.uk; Yorgo Chamoun, École Polytechnique, , France, yorgo.chamoun@polytechnique.edu; Ambrus Kapsi, Eötvös Loránd University, , Hungary, akapsi@inf.elte.hu; Michael Shulman, University of San Diego, , USA, shulman@sandiego.edu.

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).

© 2024 Copyright held by the owner/author(s).

ACM 2475-1421/2024/1-ART78

<https://doi.org/10.1145/3632920>

The external parametricity translation. By mutual induction on syntactic contexts, substitutions, types and terms, we define the following $-^P$ operations. On contexts, we further define operations 0 and 1 and we write k when we mean either. On substitutions we mutually prove an equation.

$$\frac{\Gamma : \text{Con}}{\Gamma^P : \text{Con}} \quad \frac{\sigma : \text{Sub } \Delta \Gamma}{\sigma^P : \text{Sub } \Delta^P \Gamma^P} \quad \frac{A : \text{Ty } \Gamma}{A^P : \text{Ty}(\Gamma^P, A[0_\Gamma], A[1_\Gamma])} \quad \frac{t : \text{Tm } \Gamma A}{t^P : \text{Tm } \Gamma^P(A^P[t[0_\Gamma], t[1_\Gamma]])}$$

$$k_\Gamma : \text{Sub } \Gamma^P \Gamma \quad \sigma \circ k_\Delta = k_\Gamma \circ \sigma^P$$

The main point of this translation is to compute the logical relation A^P from a type A (the third operation). For a closed type $A : \text{Ty } \diamond$, we obtain a homogeneous binary relation $A^P : \text{Ty}(\diamond, A, A)$, that is, a type depending on two variables of type A . For an A in a non-empty context, A^P is a heterogeneous relation which depends on the operations for contexts. Because types can include terms, we need to define $-^P$ mutually on terms and substitutions as well. We explain how these operations are defined for each sort.

- Γ^P is a context that contains two copies of each type in Γ together with witnesses of their relatedness. The empty context \diamond stays empty. For a context ending with A , we obtain two copies of A which are substituted by 0 and 1, respectively; finally we have a witness of relatedness. The projections 0 and 1 return the x_0 and x_1 components, respectively.

$$\diamond^P := \diamond \quad (\Gamma, x : A)^P := \Gamma^P, x_0 : A[0_\Gamma], x_1 : A[1_\Gamma], x_2 : A^P[x_0, x_1]$$

$$k_\diamond := \epsilon \quad := (k_\Gamma, x \mapsto x_k)$$

- A substitution is a list of terms, each variable x in the codomain context is mapped to some term t which we denote $x \mapsto t$. On the empty substitution $-^P$ is the identity, on a substitution into an extended context $(\sigma, x \mapsto t) : \text{Sub } \Delta(\Gamma, x : A)$, it is defined pointwise.

$$\epsilon^P := \epsilon \quad (\sigma, x \mapsto t)^P := (\sigma^P, x_0 \mapsto t[0_\Delta], x_1 \mapsto t[1_\Delta], x_2 \mapsto t^P)$$

- A^P is a heterogeneous relation between two different copies of A , the dependencies of which are given by 0 and 1 components, respectively. For example,

$$(\diamond, x : A, y : B)^P = \diamond, x_0 : A, x_1 : A, x_2 : A^P[x_0, x_1], y_0 : B[x_0], y_1 : B[x_1], y_2 : B^P[y_0, y_1].$$

It is defined separately for each type A . On the universe, $-^P$ returns the relation space. On El , it returns the type of witnesses of the relation using function application $- \$ -$. Both U and $\text{El } a$ are types, a is a term of type U .

$$\text{U}^P[a_0, a_1] := \text{El } a_0 \Rightarrow \text{El } a_1 \Rightarrow \text{U} \quad (\text{El } a)^P[x_0, x_1] := \text{El}(a^P \$ x_0 \$ x_1)$$

- The term t^P says that t respects logical relations: if the relations hold for every dependency in the context, the relation A^P also holds for the two copies of t , depending on the respective copies of Γ . Sometimes t^P is called the fundamental lemma for the term t . Note that when we say “relation” we always mean proof-relevant relation (correspondence, or family of types).

The “hello world” example of parametricity. A term in $\text{Tm}(\diamond, x : \text{U}, y : \text{El } x)(\text{El } x)$ can only contain two free variables, x and y . Using the translation $-^P$, we show that for any such term t and closed terms b and u , the substituted term $t[b, u]$ is equal to u . This is one way to formalise that $\forall a. a \rightarrow a$ has only one element. In fact, the unary version of the translation is enough, so for this example we restrict ourselves to the $k = 0$ case and omit the 1 components (alternatively, we could fill the 1 components using dummy \top arguments). Now t^P says that if there is a code for a type x_0 , a predicate on elements of this type, and an element y_0 for which the predicate holds, then the predicate will also hold for $t[x_0, y_0]$.

$$t^P : \text{Tm}(\diamond, x_0 : \text{U}, x_2 : \text{El } x_0 \Rightarrow \text{U}, y_0 : \text{El } x_0, y_2 : \text{El}(x_2 \$ y_0))(\text{El}(x_2 \$ (t[x_0, y_0])))$$

Given a closed type $b : \text{Tm} \diamond U$, a term $u : \text{Tm} \diamond (\text{El } b)$, we define a predicate $Q \$ y := \text{Eq}_{\text{El } b} u y$ expressing equality to u . As Q holds for u by reflexivity, we obtain the following substituted term with the desired type.

$$t^P[b, Q, u, \text{refl}] : \text{Tm} \diamond (\text{Eq}_{\text{El } b} u (t[b, u]))$$

The operation $-^P$ can be defined for most well-behaved type theories as a syntactic translation or model construction. However, it only gives parametricity in the empty context. In the above example if we replace the empty context \diamond by an arbitrary Γ , we only obtain

$$\text{Tm } \Gamma^P (\text{Eq}_{\text{El } b} u (t[b, u])),$$

which expresses the same equality, but it is only valid in a different (larger) context Γ^P which includes the information that the elements in Γ are themselves parametric. Operations that are internal to type theory (such as λ or application $\$$) do not act on the full context, as can be seen by looking at their inference rules. This is in contrast with $-^P$ which takes a term into a completely different context. This is why $-^P$ is called an *external* parametricity translation.

Internalising parametricity. Internal parametricity can be obtained by postulating a substitution $R_\Gamma : \text{Sub } \Gamma \Gamma^P$ for every context Γ [Altenkirch and Kaposi 2015]. Now given a $t : \text{Tm } (\Gamma, x : U, y : \text{El } x) (\text{El } x)$, and b, u, Q as above we have

$$t^P[R_\Gamma, b, Q, u, \text{refl}] : \text{Tm } \Gamma (\text{Eq}_{\text{El } b} u (t[b, u])),$$

thus we obtain the desired equality in the same context as t .

However there is no hope of being able to define the substitution R_Γ by induction on the syntax. Type theory has non-parametric models in which the above equality does not hold for any t : e.g., models with excluded middle [Booij et al. 2016] or a type-case operator [Boulier et al. 2017].

So it is not a surprise that when trying to define R_Γ by induction on the context Γ , we need to extend the syntax by a new operator which we call *rel*:

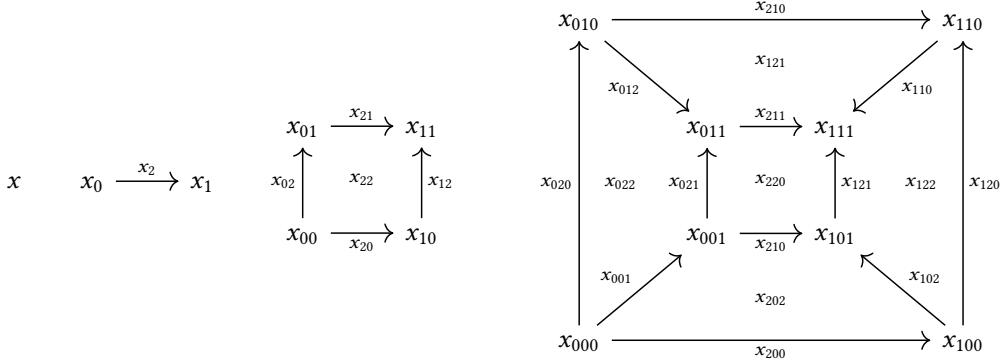
$$R_\diamond := \epsilon \quad R_{\Gamma, x:A} := (R_\Gamma, x_0 \mapsto x, x_1 \mapsto x, x_2 \mapsto \text{rel } x) \quad \frac{a : \text{Tm } \Gamma A}{\text{rel } a : \text{Tm } \Gamma (A^P[R_\Gamma, a, a])}$$

Once we introduce new terms (such as *rel*), we have to say how $-^P$ acts on them, and it is not clear how to do this. A solution is to turn the $-^P$ operations into operators of the syntax and their definitions into conversion rules. Such a $-^P$ relation behaves like an identity type that is reflexive and a congruence, but has no transport (it is sometimes called a Bridge type). Before defining our syntax with a Bridge type, we take a detour to understand how iterated usages of $-^P$ behave.

Higher cubes in external parametricity. Γ^P can be seen as a context of lines, $(\Gamma^P)^P$ as a context of squares, $((\Gamma^P)^P)^P$ as a context of three-dimensional cubes, and so on. We illustrate this by computing the contents of $(\diamond, x : A)$ after applying $-^P$ to it twice.

$$\begin{aligned} (\diamond, x : A)^{P^P} &= (\diamond, x_0 : A, x_1 : A, x_2 : A^P[x_0, x_1])^P = \\ &\quad (\diamond, x_{00} : A, \quad x_{01} : A, \quad x_{02} : A^P[x_{00}, x_{01}], \\ &\quad x_{10} : A, \quad x_{11} : A, \quad x_{12} : A^P[x_{10}, x_{11}], \\ &\quad x_{20} : A^P[x_{00}, x_{10}], x_{21} : A^P[x_{01}, x_{11}], x_{22} : A^{P^P}[x_{00}, x_{01}, x_{02}, x_{10}, x_{11}, x_{12}, x_{20}, x_{21}]) \end{aligned}$$

The contexts $(\diamond, x : A)$, $(\diamond, x : A)^P$, $(\diamond, x : A)^{P^P}$, $(\diamond, x : A)^{P^{P^P}}$ can be depicted as follows.



In the last diagram, the filler for the biggest square is \$x_{221}\$ and the filler for the cube is \$x_{222}\$.

Degenerating a line. Using our newly developed geometric intuition, we explain how two substitutions of type \$\text{Sub } \Gamma^P (\Gamma^P)^P\$, namely \$R_{\Gamma^P}\$ and \$(R_\Gamma)^P\$, differ. They correspond to the two different ways of turning a line into a square: given \$(\diamond, x : A)^P\$, \$R_{(\diamond, x : A)^P}\$ produces the square on the left, \$(R_{\diamond, x : A})^P\$ produces the square on the right.

$$\begin{array}{c} x_0 \xrightarrow{x_2} x_1 \\ \text{rel } x_0 \parallel \text{rel } x_2 \parallel \text{rel } x_1 \\ x_0 \xrightarrow{x_2} x_1 \end{array} \quad \begin{array}{c} x_1 \xrightarrow{\text{rel } x_1} x_1 \\ x_2 \uparrow \quad \uparrow \text{rel } x_1 \\ x_0 \xrightarrow{\text{rel } x_0} x_0 \end{array}$$

Assuming \$R_{\Gamma^P} = (R_\Gamma)^P\$ is incompatible with injectivity of \$\Pi\$ (an analogous observation was made by [Bernardy and Moulin 2012, p. 138]). To explain this, we need to know how \$-^P\$ acts on \$\Pi\$ types:

$$\begin{aligned} (\Pi(x : A).B)^P[\gamma_P, f_0, f_1] &:= \\ \Pi(x_0 : A[0_\Gamma \circ \gamma_P], x_1 : A[1_\Gamma \circ \gamma_P], x_2 : A^P[\gamma_P, x_0, x_1]).B^P[\gamma_P, x_0, x_1, x_2, f_0 \$ x_0, f_1 \$ x_1] \end{aligned}$$

Now we can see that \$U^{P^P}[a_{00}, \dots, a_{21}]\$ is the type of two-dimensional relations, parameterised by four codes \$a_{00}, a_{01}, a_{10}, a_{11}\$ in \$U\$ and four relations \$a_{02}, a_{12}, a_{20}, a_{21}\$ between them. We compute as follows.

$$\begin{aligned} U^{P^P}[a_{00}, \dots, a_{21}] &= \\ U^P[a_0, a_1]^P[a_{00}, \dots, a_{21}] &= \\ (\text{El } a_0 \Rightarrow \text{El } a_1 \Rightarrow U^P[a_{00}, \dots, a_{21}]) &= \\ \Pi(x_{00} : \text{El } a_{00}, x_{01} : \text{El } a_{01}, x_{02} : \text{El } (a_{02} \$ x_{00} \$ x_{01}), \\ x_{10} : \text{El } a_{10}, x_{11} : \text{El } a_{11}, x_{12} : \text{El } (a_{12} \$ x_{10} \$ x_{11})) \cdot U^P[a_{20} \$ x_{00} \$ x_{10}, a_{21} \$ x_{01} \$ x_{11}] &= \\ \Pi(x_{00} : \text{El } a_{00}, x_{01} : \text{El } a_{01}, x_{02} : \text{El } (a_{02} \$ x_{00} \$ x_{01}), \\ x_{10} : \text{El } a_{10}, x_{11} : \text{El } a_{11}, x_{12} : \text{El } (a_{12} \$ x_{10} \$ x_{11})) \cdot \text{El } (a_{20} \$ x_{00} \$ x_{10}) \Rightarrow \text{El } (a_{21} \$ x_{01} \$ x_{11}) \Rightarrow U \end{aligned}$$

Assuming \$R_{(\diamond, a : U)^P} = (R_{\diamond, a : U})^P\$, we also have \$U^{P^P}[R_{(\diamond, a : U)^P}] = U^{P^P}[(R_{\diamond, a : U})^P]\$, but the first one is a type of the form \$\Pi(x_{00} : \text{El } a_0, x_{01} : \text{El } a_0 \dots) \dots\$, the second one is a type of the form \$\Pi(x_{00} :

$\text{El } a_0, x_{01} : \text{El } a_1 \dots$ If Π has injectivity (which follows from normalisation), then for any a_0, a_1 in U , $\text{El } a_0 = \text{El } a_1$.

Symmetry and emergent geometry. In a syntax for internal parametricity, we either need to postulate the existence of an infinite hierarchy of rels from which the substitutions $R_\Gamma, (R_\Gamma)^P, (R_\Gamma)^{PP}$, . . . can be obtained, or we need to provide another way to relate R_{Γ^P} and $(R_\Gamma)^P$. We choose the latter:¹ we introduce a new substitution S_Γ (called symmetry) which satisfies $S_\Gamma \circ R_{\Gamma^P} = (R_\Gamma)^P$. Intuitively, symmetry maps x_{ij} to x_{ji} .

$$\begin{array}{ccc} \begin{array}{ccc} x_{01} & \xrightarrow{x_{21}} & x_{11} \\ x_{02} \uparrow & x_{22} & \uparrow x_{12} \\ x_{00} & \xrightarrow{x_{20}} & x_{10} \end{array} & \xrightarrow{S} & \begin{array}{ccc} x_{10} & \xrightarrow{x_{12}} & x_{11} \\ x_{20} \uparrow & \text{sym } x_{22} & \uparrow x_{21} \\ x_{00} & \xrightarrow{x_{02}} & x_{01} \end{array} \end{array}$$

It turns out that this operation is enough, and there is no need to introduce higher dimensional versions of S_Γ (as in [Bernardy and Moulin 2012]) or an extra sort of intervals (as in [Bernardy et al. 2015; Cavallo 2021]). In this paper we define a theory with internal parametricity which does not have explicit geometry in the syntax. Compared to previous theories with internal parametricity, geometry is *emergent* rather than explicitly built-in. We do have ways to talk about higher dimensional cubes (as we saw when iterating $-^P$ on contexts) but this is nothing special: Martin-Löf type theory also has all higher dimensional cubes simply because the identity type can be iterated. E.g. the type $\text{Id}_{(\text{Id}_A a_{01} a_{11})} (\text{transport}_{(\text{Id}_A a_{01} -)} a_{12} (\text{transport}_{(\text{Id}_A - a_{10})} a_{02} a_{20})) a_{21}$ expresses the type of fillers of the following two-dimensional square.

$$\begin{array}{ccc} a_{01} & \xrightarrow{a_{21}} & a_{11} \\ a_{02} \uparrow & & \uparrow a_{12} \\ a_{00} & \xrightarrow{a_{20}} & a_{10} \end{array}$$

We have one two-dimensional operation (S_Γ) and one equation about S_Γ that involves three dimensional cubes, but we never mention anything higher than that.

Obtaining a theory from a model. Even if higher cubes are not explicitly built into our syntax, our type theory is informed by an analysis of the cubical set model built on Bezem-Coquand-Huber (BCH) cubes [Bezem et al. 2013].

The BCH cube category can be presented using a finite number of basic operators and equations between them. It is given by the free 2-category generated by the diagram on the left in Figure 1 and five equations relating the 2-cells. This 2-category has one 0-cell $*$, and can be seen as a 1-category where objects are given by 1-cells from $*$ to itself, and morphisms are given by the 2-cells. In this presentation we have numbered dimensions instead of named dimensions (see [Buchholtz and Morehouse 2017] for a comparison of different presentations). The objects of the cube category are natural numbers given by $0 = \text{id}$, $1 = \text{suc} \circ \text{id}$, $2 = \text{suc} \circ \text{suc} \circ \text{id}$, and so on. Degeneracies are generated by R , e.g. there is one map R from 1 to 0, there are two maps from 2 to 1, namely $\text{id}_{\text{suc}} \bullet R$ and $R \bullet \text{id}_{\text{suc}}$ (where $- \bullet -$ denotes horizontal composition). Face maps are similarly generated by 0 and 1, and there is a symmetry operation S .

The category of presheaves over the BCH cube category supports the exact same structure with maps in the other direction (diagram on the right in Figure 1). Here V is precomposition by suc , and the natural transformations are named after their generating 2-cells. This picture can

¹In fact, it is not clear whether the former is even possible. The naive presheaf model of such a theory does not satisfy the needed computation rule for $-^P$ on Π , and the syntax has stuck terms that it is not clear how to compute with.

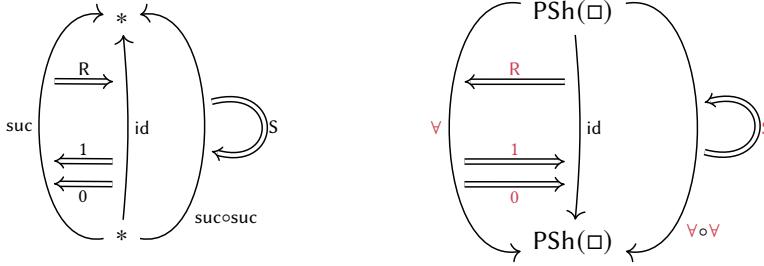


Fig. 1. The generating 1-cells and 2-cells of the BCH cube category \square as a 2-category (left). Structure on presheaves over this category (right). The two occurrences of $*$ represent the same object, depicted twice solely for presentation, and the same holds for $\text{PSh}(\square)$.

be reified into new operations of the type theory: we add a strict morphism from the syntax to itself corresponding to V , natural transformations R , 0 , 1 and S , and the five extra equations. The new equations and the other operators that we add are those which are justified by this presheaf model: the morphism V strictly respects the substitution calculus (the category with families, CwF), \top , Σ , strict identity Eq, Bool and K. Our model does not justify an equation such as $\text{V}(\Pi A B) = \Sigma(\Pi (\text{V}A) (\text{V}B)) \dots$, but only the analogous isomorphism, so for Π we add new operators and equations expressing this isomorphism. Similarly, VU is described by a section-retraction pair. We call the collection of these new operations the *global theory*, as it involves operations on contexts and substitutions. We directly obtained the global theory from the presheaf model over BCH cubes, thus it is immediately justified by this model.

Local theory. Multi-modal type theory [Gratzer et al. 2021] gives a generic way to construct a type theory from a CwF morphism such as suc . It uses that precomposition $\text{V} = \text{suc}^*$ has a left adjoint $\text{suc}_!$ (the left Kan extension), and this has a dependent right adjoint. However this theory is still non-local as it introduces extra (un)lock operations on contexts. A presentation of internal parametricity using this method is [Cavallo 2021], where the lock operation of multi-modal type theory becomes context extension with an interval variable.

In our case we can define a version of our theory with only local operations, that is, operations that do not change the context. This is what we call the *local theory*. It is specified by the exact same data as the global theory, but now V is not a morphism from the syntax to itself, but a morphism from the standard model to itself, internal to presheaves over the syntax. We explain this in detail.

Any category of presheaves has a type-theoretic internal language. For example, when we write $A : \text{Set}$ in this internal language, externally this means that A is a presheaf. Similarly, the internal $B : A \rightarrow \text{Set}$ means a dependent presheaf over A externally. If the base category C of the presheaf model is not only a category, but a CwF, then internally we have $\text{Ty} : \text{Set}$ and $\text{Tm} : \text{Ty} \rightarrow \text{Set}$ which externally are defined by the presheaf of types and the dependent presheaf of terms in C . This is the main idea of two-level type theory [Altenkirch et al. 2016; Annenkov et al. 2017].

If C has Σ -types, then internally we have $\Sigma : (A : \text{Ty}) \rightarrow (\text{Tm}A \rightarrow \text{Ty}) \rightarrow \text{Ty}$ together with an isomorphism $(a : \text{Tm}A) \times \text{Tm}(B a) \cong \text{Tm}(\Sigma A B)$. In this case (still internally), Ty and Tm form a universe closed under Σ types. [Bocquet et al. 2023] call this a higher-order model of type theory with Σ types. The situation is analogous for other type formers, e.g., if C has Π -types, then internally we have $\Pi : (A : \text{Ty}) \rightarrow (\text{Tm}A \rightarrow \text{Ty}) \rightarrow \text{Ty}$ together with an isomorphism $((a : \text{Tm}A) \rightarrow \text{Tm}(B a)) \cong \text{Tm}(\Pi A B)$.

Now we define a CwF internal to presheaves over C . We call this the internal standard model. Contexts in this model are given by Ty , a type in a context Ω is a function $\text{Tm } \Omega \rightarrow \text{Ty}$, a term in context Ω of type A is a dependent function $(\omega : \text{Tm } \Omega) \rightarrow \text{Tm}(A \omega)$. Context extension is given by Σ , hence we need that Ty is closed under Σ , which in turn needs that C has Σ types. This standard model is a generalisation of the set model (or type model, or metacircular model) [Altenkirch and Kaposi 2016] and is a variant of the telescopic contextualisation of [Bocquet et al. 2023].

The \forall of the local theory is specified by a CwF-morphism from this standard model to itself. Internally written, it maps contexts to contexts ($\forall : \text{Ty} \rightarrow \text{Ty}$), types to types ($\forall d : (\text{Tm } A \rightarrow \text{Ty}) \rightarrow \text{Tm } (\forall A) \rightarrow \text{Ty}$), and so on. We add a “d” suffix to the operation on types to distinguish from the one on contexts. Externally, these operations are natural transformations described as follows.

$$\frac{A : \text{Ty } \Gamma}{\forall A : \text{Ty } \Gamma} \quad \frac{(\forall A)[\sigma] = \forall(A[\sigma])}{(\forall d)[\sigma] = \forall d(B[\sigma \uparrow])} \quad \frac{B : \text{Ty } (\Gamma, x : A)}{\forall d B : \text{Ty } (\Gamma, x : \forall A)} \quad \frac{}{(\forall d B)[\sigma] = \forall d(B[\sigma \uparrow])}$$

We obtain all of the local theory this way: we start with a strictly democratic CwF C with $\top, \Sigma, \text{Eq}, \Pi, \text{U}$ and Bool (we call this the core theory). Internally to presheaves over C , we have the standard model of this core theory. Now $\forall, \forall d$ and the other new operations and equations providing internal parametricity are specified by a core theory morphism from this standard model to itself. Just as in the case of the global theory, this morphism respects the CwF structure, $\top, \Sigma, \text{Eq}, \text{K}$ and Bool strictly, Π up to an isomorphism, and U up to section-retraction. Note that this only gives a specification of the local theory, and does not directly provide a model of it. We justify the local theory by deriving its syntax from the syntax of the global theory which we localise using \mathbf{R} .

The local theory is truly local: it does not mention contexts and it can be described as a second-order generalised algebraic theory (SOGAT) [Uemura 2019]. From this SOGAT we obtain a first-order GAT in a way that makes sure that all operations are stable under substitution. As far as we know, our local theory is the first non-substructural type theory describing presheaves over BCH cubes. We distinguish the corresponding operations of the local and global theories by writing those of the global theory in **brick red colour**.

Span-based parametricity. In our global theory, $\forall \Gamma$ exactly corresponds to the Γ^P of the external parametricity translation. For types however we don't compute parametricity relations, but parametricity spans: $\forall A : \text{Ty } (\forall \Gamma)$ together with maps $k_A : \text{Tm } (\forall \Gamma, \forall A) (A[k_\Gamma])$. Similarly, in the local theory, $\forall A$ is a type with the structure of a span $A \xleftarrow{0_A} \forall A \xrightarrow{1_A} A$. We can recover the relational version (Bridge type) using the strict identity type Eq :

$$A^P a_0 a_1 := \Sigma(a : \forall A). \text{Eq}_A (0_A a) a_0 \times \text{Eq}_A (1_A a) a_1.$$

Just as $-^P$, the operation \forall computes definitionally on \top, Σ, Eq and Bool . For example, \forall of a Σ is equal to a Σ of \forall s. On function types we have the span-preservation variant of usual relation-preservation $(A \Rightarrow B)^P f_0 f_1 = \Pi(x_0 : A[0], x_1 : A[1], x_2 : A^P x_0 x_1). B^P (f_0 \$ x_0) (f_1 \$ x_1)$ saying that related inputs are mapped to related outputs. An element of $\forall(A \Rightarrow B)$ corresponds to a function t from $\forall A$ to $\forall B$, and functions t_k from $A[k]$ to $B[k]$ such that the following diagram commutes.

$$\begin{array}{ccc} A[0] & \xleftarrow{0_A} & \forall A & \xrightarrow{1_A} & A[1] \\ t_0 \downarrow & & \downarrow t & & \downarrow t_1 \\ B[0] & \xleftarrow{0_B} & \forall B & \xrightarrow{1_B} & B[1] \end{array}$$

This correspondence holds only up to isomorphism in the model and thus in our theory.

In the external parametricity translation, we have $\mathbf{U}^P a_0 a_0 = (\mathbf{El} a_0 \Rightarrow \mathbf{El} a_1 \Rightarrow \mathbf{U})$. The main reason that we use span-based instead of relation-based parametricity is that in our model this is not an equality, only a logical equivalence.² We have maps in both directions, but the composite map

$$(\mathbf{El} a_0 \Rightarrow \mathbf{El} a_1 \Rightarrow \mathbf{U}) \rightarrow \mathbf{U}^P a_0 a_1 \rightarrow (\mathbf{El} a_0 \Rightarrow \mathbf{El} a_1 \Rightarrow \mathbf{U})$$

is not the identity (and neither is the other roundtrip). However, if we replace relations by spans, we do have that the analogous composite map for spans

$$\Sigma(a, a_0, a_1 : \mathbf{U}).(\mathbf{El} a \Rightarrow \mathbf{El} a_0) \times (\mathbf{El} a \Rightarrow \mathbf{El} a_1) \rightarrow \forall \mathbf{U} \rightarrow \Sigma(a, a_0, a_1 : \mathbf{U}).(\mathbf{El} a \Rightarrow \mathbf{El} a_0) \times (\mathbf{El} a \Rightarrow \mathbf{El} a_1)$$

is the identity. The intuition for why this works for spans and not for relations is that presheaves are span-based by nature. Given a presheaf Γ over BCH-cubes \square , we denote the action on objects $\Gamma : \square \rightarrow \mathbf{Set}$ and the action on morphisms $-[-]_\Gamma : \Gamma I \rightarrow \square(J, I) \rightarrow \Gamma J$. Now, at levels 0 and 1 we have a span $\Gamma 0 \xleftarrow{-[0]_\Gamma} \Gamma 1 \xrightarrow{-[1]_\Gamma} \Gamma 0$ instead of a relation $\Gamma 0 \rightarrow \Gamma 1 \rightarrow \mathbf{Set}$. Relation-based presheaves are called Reedy fibrant (relative to families of sets as the underlying notion of “fibration”) [Kraus and Sattler 2017], and it should be possible to construct a Reedy fibrant presheaf model of internal parametricity, but we leave this for future work. A model of internal parametricity based on refined presheaves similar to Reedy fibrant ones is [Bernardy et al. 2015].

The other roundtrip for the correspondence $\forall \mathbf{U} \leftrightarrow \mathbf{Span}$ is unfortunately not identity in our model. Thus we justify this correspondence up to a section-retraction pair.

Metatheory. As our global theory arose from a presheaf model, it is not surprising that it is modelled by the exact same presheaf category. We extend gluing [Kaposi et al. 2019a] to the global theory, and define a global section functor satisfying the necessary conditions from the syntax to our presheaf model. We know that the syntaxes of the global and local theories are isomorphic, hence, as a result, our local theory satisfies canonicity and has an external parametricity translation. We conjecture that a version of our theory without equality reflection satisfies normalisation.

1.1 Structure of the Paper

After summarising related work and our notations, we introduce our local theory in Section 2, and describe some applications including well-known usages of internal parametricity. This section can be understood without prior familiarity with models of type theory or presheaves. For the rest of the paper we try to be as self-contained as possible, and refer to the relevant literature.

In Section 3, we define the global version of the theory as a generalised algebraic theory (GAT). The syntax of the theory is given by the initial algebra (model) which exists for any GAT. We also show that the global theory has a presheaf model. Section 4 shows the isomorphism of the local and global syntaxes. Then in Section 5 we prove that our global theory has a gluing model and as a consequence satisfies canonicity: every closed term of the boolean type is equal to either true or false. By the previous isomorphism this result holds for both the local and global theories.

1.2 Related Work

The first type theory with internal parametricity was defined by Bernardy and Moulin [Bernardy and Moulin 2012]. It contains a syntax for arbitrary dimensional cubes. The apd operator in our local syntax is very similar to their double bracket operator, but our theory only mentions cubes up to dimension three. Bernardy and Moulin simplified their syntax later using named dimensions [Bernardy and Moulin 2013] and further refined it using a sort of intervals [Bernardy et al. 2015] similar to that of cubical type theories (e.g. [Cohen et al. 2015]). Using the same (BCH) cube category as the first cubical set

²Although if we observe that both sides are the type of objects of some category, we can say that it extends to an equivalence of categories.

model of univalence [Bezem et al. 2013], the paper [Bernardy et al. 2015] defines a presheaf model of internal parametricity. It uses a refined notion of dependent presheaf for interpreting types similar to Reedy fibrancy [Kraus and Sattler 2017]. Our presheaf model uses ordinary presheaves and avoids the need of Reedy fibrancy by having span-based instead of relational parametricity. [Bernardy et al. 2015] has an equation “SURJ-TYP” the analog of which we were not able to justify in our model. This would correspond to unspan being an isomorphism, not only a section (see (7) in Problem 3.7). It seems that having Reedy fibrant types does not make a difference in this respect.

Cavallo and Harper [Cavallo 2021; Cavallo and Harper 2021] define a cubical type theory with univalence and internal parametricity at the same time. They support a double-presheaf model with cartesian cubes for the identity type and BCH cubes for parametricity. They justify relational parametricity, but the correspondence between the logical relation at U and relation space only holds up to internal equivalence (and hence propositional equality, by univalence), and not definitional section-retraction as in our theory. This is enough to derive the consequences of internal parametricity, however. Van Muylder, Nuyts and Devriese [Muylder et al. 2024] extend Cubical Agda with internal parametricity following Cavallo and Harper. Inside this theory they shallowly embed a “relational observational type theory” in which logical relations are computed as in [Bernardy and Moulin 2012], but it does not feature iterated parametricity.

Nuyts et al. [Nuyts and Devriese 2018; Nuyts et al. 2017] analyse the presheaf model of internal parametricity, and define type theories where the parametricity relation and the (non-univalent) identity type are special cases of a general construction “relatedness”. These syntaxes use two different kind of Π types (parametric and non-parametric ones) and there is no proof of canonicity.

Our global syntax is very close to the naive syntax in [Altenkirch and Kaposi 2015]. By closely following a model, we make sure that we do not miss any equations, and we manage to prove canonicity, even being “naive” in their sense.

Recent work on cubical type theories [Angiuli et al. 2021; Cohen et al. 2015; Vezzosi et al. 2021] has used different cube categories that contain diagonals and sometimes connections as well, due to their advantages when formulating higher inductive types. The resulting presheaf categories satisfy a different computation rule for \forall of Π (function extensionality), which is correct homotopically but inappropriate for parametricity. On the other hand, earlier work on cubical homotopy theory used a cube category lacking symmetries, which also fails to have our desired computation rule for \forall of Π . The BCH cube category is “just right”.

Finally, although we explicitly discuss only “binary” parametricity, one can consider n -ary parametricity for any natural number n , in which case there are n possible values for k wherever it appears. Unary parametricity is also common in the literature. Nothing that we say should be sensitive to the choice of n , and often even our notation can be applied directly in the n -ary case.

1.3 Metalanguage and Notation

Our metalanguage is extensional type theory with quotients and propositional extensionality (unlike in the above paragraph “Local theory”, this extensional type theory is not the outer level of a two-level type theory). Our constructions can be also understood as taking place in a constructive set theory. We use Agda-style notation with implicit arguments usually omitted or written in curly braces $\{\dots\}$ and we employ implicit coercions and overloaded projections.

We use categories with families (CwFs, [Castellan et al. 2019]) as the notion of model of type theory. The components of the category part are denoted Con , Sub , $- \circ -$, id , the terminal object (empty context) \diamond , the empty substitution ϵ . The families of types and terms are Ty , Tm , their instantiation of substitution operations are both denoted $-[-]$. We write $\Gamma \triangleright A$ for the context Γ extended by the type A . We write $(\sigma, t) : \text{Sub } \Delta(\Gamma \triangleright A)$ for $\sigma : \text{Sub } \Delta \Gamma$ and $t : \text{Tm } \Delta(A[\sigma])$, and denote the projections by $p : \text{Sub } (\Gamma \triangleright A) \Gamma$ and $q : \text{Tm } (\Gamma \triangleright A)(A[p])$. We write $(\sigma \uparrow)$ for $(\sigma \circ p, q)$.

$$\begin{array}{c}
\frac{}{\top} \quad \frac{}{\text{tt} : \top} \quad \frac{t : \top}{t = \text{tt}} \\[1ex]
\frac{A \quad x : A \vdash B}{\Sigma(x : A).B} \quad \frac{u : A \quad v : B[v \mapsto u]}{(u, v) : \Sigma(x : A).B} \quad \frac{t : \Sigma(x : A).B}{\pi_1 t : A} \quad \frac{t : \Sigma(x : A).B}{\pi_2 t : B[x \mapsto \pi_1 t]} \\[1ex]
\pi_1(u, v) = u \quad \pi_2(u, v) = v \quad t = (\pi_1 t, \pi_2 t) \\[1ex]
\frac{a_0 : A \quad a_1 : A}{\text{Eq}_A a_0 a_1} \quad \frac{a : A}{\text{refl}_a : \text{Eq}_A a a} \quad \frac{e : \text{Eq}_A a_0 a_1}{a_0 = a_1} \quad \frac{e : \text{Eq}_A a a}{e = \text{refl}_a} \\[1ex]
\frac{A \quad x : A \vdash B}{\Pi(x : A).B} \quad \frac{x : A \vdash t : B}{\lambda x.t : \Pi(x : A).B} \quad \frac{t : \Pi(x : A).B \quad u : A}{t \$ u : B[x \mapsto u]} \\[1ex]
(\lambda x.t) \$ u = t[x \mapsto u] \quad t = \lambda x.t \$ x \\[1ex]
\frac{}{\text{U}} \quad \frac{a : \text{U}}{\text{El } a} \quad \frac{A}{\text{c } A : \text{U}} \quad \text{El}(\text{c } A) = A \quad \text{c}(\text{El } a) = a \\[1ex]
\frac{}{\text{Bool}} \quad \frac{\text{true} : \text{Bool}}{\text{false} : \text{Bool}} \quad \frac{}{x : \text{Bool} \vdash A \quad t : \text{Bool} \quad u : A[x \mapsto \text{true}] \quad v : A[x \mapsto \text{false}]} \\[1ex]
\text{ite}_{x.A} t u v : A[x \mapsto t] \\[1ex]
\text{ite true } u v = u \quad \text{ite false } u v = v
\end{array}$$

Fig. 2. The core theory. See Definition 3.1 for an external description.

We write $p^2 = p \circ p$, $p^3 = p^2 \circ p$, and natural numbers for De Bruijn indices: $n = q[p^n]$. We denote an isomorphism between two contexts by $\sigma : \Gamma \cong \Delta$ which means that $\sigma : \text{Sub } \Gamma \Delta$ and there is also a $\sigma^{-1} : \text{Sub } \Delta \Gamma$ and both compositions $\sigma \circ \sigma^{-1}$ and $\sigma^{-1} \circ \sigma$ are the identity. Similarly, for types we write $A \cong B$ to mean that we have terms $t : \text{Tm}(\Gamma \triangleright A)(B[p])$ and $t^{-1} : \text{Tm}(\Gamma \triangleright B)(A[p])$ such that $t[p, t^{-1}] = q$ and $t^{-1}[p, t] = q$.

2 THE LOCAL THEORY AND APPLICATIONS

In this section we list and explain the rules of our local theory with internal parametricity and show how to apply it to derive consequences of parametricity. This section can be understood without previous knowledge of models of type theory.

We list the operations and equations of our core type theory in Figure 2 and the rules providing internal parametricity in Figure 3. The notation can be understood as listing the operations and equations of a second-order generalised algebraic theory (SOGAT) [Bocquet et al. 2023; Uemura 2019]. This is why we don't have to list rules about contexts or substitutions. (The external presentation of the core theory is Definition 3.1, the external presentation of the local theory is described in Section 4.1.) We have two sorts, types are denoted A , terms are $t : A$. In case an operation is a binder, some of its arguments have extra dependencies listed before a \vdash . Some operations have implicit arguments, for example the constructor for Σ types $(-, -)$ has two implicit inputs A and B which we omit for readability. Similarly, the equation $\pi_1(u, v) = u$ has four implicit arguments: A, B, u, v . We omit the premises of most equations. Note that the η rule for Σ types written $t = (\pi_1 t, \pi_2 t)$ only makes sense for t having a Σ type, so we don't have to add this as an explicit assumption. If one views Figures 2 and 3 as constructors for a syntax, then we have well-typed (intrinsic) terms which are quotiented by conversion.

Ad Figure 2. We have extensional Martin-Löf type theory with unit type \top , Σ types, identity types with reflection and uniqueness of identity (UIP), Π types which all satisfy β and η rules. We have a Coquand-universe [Coquand 2018] which we don't index for convenience, but everything

The new operations:

$$\begin{array}{c}
 \frac{A \quad B \quad x : A \vdash t : B \quad a_2 : \forall A}{\forall A \quad \text{ap}(x.t) a_2 : \forall B} \quad \frac{x : A \vdash B \quad a_2 : \forall A}{\forall d(x.B) a_2} \quad \frac{x : A \vdash B \quad x : A \vdash t : B \quad a_2 : \forall A}{\text{apd}(x.t) a_2 : \forall d(x.B) a_2} \\
 \frac{a_2 : \forall A}{k_A a_2 : A} \quad \frac{a : A}{R_A a : \forall A} \quad \frac{a_{22} : \forall(\forall A)}{S_A a_{22} : \forall(\forall A)} \quad \frac{A_k \quad A}{\text{unspan } A_k A} \quad \frac{x : A \vdash t_k : A_k}{x : A \vdash B} \\
 \frac{x : A \vdash B}{x : A, y : B \vdash C} \\
 \frac{x_2 : \forall A}{t_k : \Pi(y : B[x \mapsto k_A a_2]).C[x \mapsto k_A a_2]} \\
 \frac{t : \Pi(y_2 : \forall d(x.B) a_2).\forall d((x,y).C) (a_2, y_2)}{y_2 : \forall d(x.B) a_2 \vdash t_k \$ \text{kd}_{x,B} a_2 y_2 = \text{kd}_{(x,y).C} (a_2, y_2) (t \$ y_2)} \\
 \frac{\text{mk}\forall\Pi a_2 t_k t : \forall d(x.\Pi(y : B).C) a_2}{}
 \end{array}$$

An abbreviation:

$$\frac{x : A \vdash B \quad a_2 : \forall A \quad b_2 : \forall d(x.B) a_2}{\text{kd}_{x,B} a_2 b_2 : B[x \mapsto k_A a_2] \quad \text{kd}_{x,B} a_2 b_2 := \pi_2(k_{\Sigma(x:A).B} (a_2, b_2))} \quad \boxed{\text{kd}_{x,B} a_2 b_2 : B[x \mapsto k_A a_2] \quad \text{kd}_{x,B} a_2 b_2 := \pi_2(k_{\Sigma(x:A).B} (a_2, b_2))} \\$$

The new equations:

$$\begin{array}{lll}
 \text{Core} & \text{ap}(x.g[y \mapsto f]) a_2 = \text{ap}(y.g) (\text{ap}(x.f) a_2) & \text{ap}(x.x) a_2 = a_2 \quad \forall T = T \\
 & \forall d(x.C[y \mapsto f]) a_2 = \forall d(y.C) (\text{ap}(x.f) a_2) & \text{apd}(x.t[y \mapsto f]) a_2 = \text{apd}(y.t) (\text{ap}(x.f) a_2) \\
 & \forall(\Sigma(x : A).B) = \Sigma(x_2 : \forall A).\forall d(x.B) x_2 & \text{ap}(x.(u,v)) a_2 = (\text{ap}(x.u) a_2, \text{apd}(x.v) a_2) \\
 \\
 \text{Const} & \forall d(_.B) a_2 = \forall B \quad \frac{B \quad x : A \vdash t : B}{\text{apd}(x.t) a_2 = \text{ap}(x.t) a_2} & \\
 k, R, S & k_B (\text{ap}(x.f) a_2) = f[x \mapsto k_A a_2] \quad \text{ap}(x.f) (R_A a) = R_B (f[x \mapsto a]) & \\
 & \text{ap}(x_2.\text{ap}(x.f) x_2) (S_A a_{22}) = S_B (\text{ap}(x_2.\text{ap}(x.f) x_2) a_{22}) \quad k_A (R_A a) = a \\
 & k_{\forall A} (S_A a_{22}) = \text{ap}(x_2.k_A x_2) a_{22} \quad S_A (R_{\forall A} a_2) = \text{ap}(x.R_A x) a_2 \quad S_A (S_A a_{22}) = a_{22} \\
 & S_{\forall A} (\text{ap}(x_{22}.S_A x_{22}) (S_{\forall A} a_{222})) = \text{ap}(x_{22}.S_A x_{22}) (S_{\forall A} (\text{ap}(x_{22}.S_A x_{22}) a_{222})) \\
 \\
 \Sigma & \forall d(x.\Sigma(y : B).C) a_2 = \Sigma(y_2 : \forall d(x.B) a_2).\forall d((x,y).C) (a_2, y_2) & \\
 & \text{apd}(x.(u,v)) a_2 = (\text{apd}(x.u) a_2, \text{apd}(x.v) a_2) & \\
 \\
 \text{Eq} & \forall d(x.\text{Eq}_B u v) a_2 = \text{Eq}_{\forall d(x.B) a_2} (\text{apd}(x.u) a_2) (\text{apd}(x.v) a_2) & \\
 \\
 \Pi & \text{kd}_{x.\Pi(y:B).C} a_2 (\text{mk}\forall\Pi a_2 t_k t) = t_k & \\
 & \lambda y_2.\text{apd}((x,f,y).f \$ y) (a_2, \text{mk}\forall\Pi a_2 t_k t, y_2) = t & \\
 & \text{mk}\forall\Pi a_2 (\text{kd}_{x.\Pi(y:B).C} a_2 t_2) (\lambda y_2.\text{apd}((x,f,y).f \$ y) (a_2, t_2, y_2)) = t_2 & \\
 \\
 U & \text{El}(k_U (\text{unspan } A_k A t_k)) = A_k \quad \forall d(x.\text{El } x) (\text{unspan } A_k A t_k) = A & \\
 & \text{kd}_{x.\text{El } x} (\text{unspan } A_k A t_k) a = t_k [x \mapsto a] & \\
 \\
 \text{Bool} & \forall \text{Bool} = \text{Bool} \quad \text{ap}(x.\text{true}) a_2 = \text{true} \quad \text{ap}(x.\text{false}) a_2 = \text{false} & \\
 & \text{apd}(x.\text{ite}_y.B t u v) a_2 = \text{ite}_y_{\forall d(x.B) a_2} (\text{ap}(x.t) a_2) (\text{ap}(x.u) a_2) (\text{ap}(x.v) a_2) & \\
 \end{array}$$

Fig. 3. The rules extending the core theory with internal parametricity (local theory). The equations are grouped by type former.

in this paper can be redone using universe indexing. In that case we would need to index types (and terms) by their level as well. Finally, we have a type of booleans with dependent elimination ite.

Ad Figure 3. We have a nondependent \vee on types which computes the logical span from a type. We know that nondependent maps (terms $x : A \vdash t : B$ where B does not depend on x) preserve logical spans, we call this ap as an homage to the “apply on path” congruence operation in HoTT. For types depending on a variable, we have dependent logical spans Vd which depend on a base logical span. These are also preserved by terms witnessed by apd (this is our main internal parametricity operation). Then we have projections from $\vee A$ to A , in the binary case k can be either 0 or 1. The k_A provide the legs of the logical span $\vee A$. (The dependent version of this projection can be derived using Σ types.) We can build degenerate logical spans using R and we can apply a symmetry operation S to a double-span (two-dimensional span). We explain $\text{mk}\vee\Pi$ and unspan below. The k , R and S operations are natural with respect to ap and satisfy five additional equations (which reflect the equations of the BCH cube category):

- (1) the bases of a degenerate span are the same as the element we started with,
- (2) the two different ways of taking the bases of a double-span are related by S ,
- (3) the two different ways of degenerating a span into a double-span are related by S ,
- (4) double symmetry is identity,
- (5) we can apply symmetry to a triple span in two different ways: we can swap dimensions 0 and 1, or we can swap dimensions 1 and 2; now first swapping 0 – 1, then 1 – 2, then 0 – 1 is the same as first swapping 1 – 2, then 0 – 1, then 1 – 2. Combined with naturality of S , this ensures that the induced symmetries of an n -fold span form the n -ary symmetric group.

In addition to \vee , Vd , ap , apd , k , R , S , we have two more operations which concern Π types and the universe ($\text{mk}\vee\Pi$ and unspan). We will explain them below.

The core equations say that ap , Vd and apd are functorial with respect to nondependent maps, \vee on unit returns unit and \vee , ap on Σ types is pointwise.

The constant equations express that \vee is a special case of Vd when the type is actually nondependent. We have an analogous equation relating apd and ap .

For Σ types, Vd is defined in a pointwise way. C has two dependencies $x : A, y : B \vdash C$, so we have $x : A, y : \forall B \vdash \text{Vd}(y.C)$, which is not what we want. Instead, we collect x and y into a Σ type and use $w : \Sigma(x : A).B \vdash C[x \mapsto \pi_1 w, y \mapsto \pi_2 w]$ which we abbreviate $(x, y) : \Sigma(x : A).B \vdash C$ by “pattern matching” on w . apd preserves pairing and preservation of π_1 and π_2 are provable.

Just as Σ , strict identity Eq is strictly preserved. Preservation of refl is automatic by UIP.

Π types are only preserved by Vd up to isomorphism. However we don’t have that $\vee(\Pi(x : B).C)$ is isomorphic to $\Pi(y_2 : \forall B).\text{Vd}(y.C) y_2$ because we can use $k_{\Pi(y:B).C}$ to obtain a function $\Pi(y : B).C$ from an element of the first type, but we cannot obtain such a function from an element of the second type. Hence we have to add more information to the second type: functions at the bases that are compatible with the function at the apex. Moreover B itself can be dependent. So the final statement is that the obvious map from $\text{Vd}(x.\Pi(y : B).C) a_2$ to a t and t_k s that make the following diagram commute is an isomorphism.

$$\begin{array}{ccc}
 B[k_A a_2] & \xrightarrow{t_k} & C[k_A a_2] \\
 \uparrow k_{\text{d}(x,B)a_2} & & \uparrow k_{\text{d}(x,y).C}(a_2, y_2) \\
 (y_2 : \text{Vd}(x.B) a_2) & \xrightarrow{t} & \text{Vd}((x, y).C)(a_2, y_2)
 \end{array}$$

The t is computed as follows.

$$\frac{t_2 : \forall d(x.\Pi(y:B).C) a_2}{t := \lambda y_2.apd((x,f,y).f \$ y) (a_2, t_2, y_2) : \Pi(y_2 : \forall d(x.B) a_2).\forall d((x.y).C) (a_2, y_2)}$$

The t_k s are just given by the $kd_{x.\Pi(y:B).C}$ s. The inverse of this map is called $mk\forall\Pi$.

Just as Π is not preserved up to equality by \forall , U is also not preserved up to equality. There is an obvious map from $a_2 : \forall U$ to a span: the apex is $\forall d(x.El x) a_2$, the bases $El(k_U a_2)$, and the legs are $x_2 : \forall d(x.El x) a_2 \vdash kd_{x.El x} a_2 x_2$. This map has a section called *unspan*. The premises of *unspan* are universally quantified over k , hence include *two* types A_0 and A_1 , and two terms t_0 and t_1 .

Bool, its constructors and eliminator are preserved strictly by \forall and *apd*.

2.1 Some Derivable Equations

For Σ , it was enough to state that $(-, -)$ is preserved by *apd*, the other direction is automatic:

$$apd(x.\pi_k t) a_2 = \pi_k (apd(x.\pi_1 t) a_2, apd(x.\pi_2 t) a_2) = \pi_k (apd(x.(\pi_1 t, \pi_2 t)) a_2) = \pi_k (apd(x.t) a_2)$$

R_A is a special case of *ap*: $R_A a = R_A (a[_ \mapsto tt]) = ap(_.a) (R_T tt) = ap(_.a) tt$.

Constant *ap* is constant: $ap(_.b) a_2 = ap(_.b[_ \mapsto tt]) a_2 = ap(_.b) (ap(_.tt) a_2) = ap(_.b) tt = ap(_.b) (ap(_.tt) a'_2) = ap(_.b[_ \mapsto tt]) a'_2 = ap(_.b) a'_2$.

k is distributive over Σ : $k_{\Sigma(x:A).B} (a_2, b_2) = (k_A a_2, kd_{x.B} a_2 b_2)$.

kd on composition: $kd_{x.B[y \mapsto t]} c_2 = kd_{y.B} (ap(x.t) c_2)$.

Relationship between *kd* and *apd*: $kd_{x.B} a_2 (apd(x.t) a_2) = t[x \mapsto k_A a_2]$.

2.2 Applications

Polymorphic identity. We revisit the “hello world” example of internal parametricity: a term witnessing that there is only one function with the type of the polymorphic identity. More precisely, given a function $f : \Pi(x : \Sigma(y : U).El y).El (\pi_1 x)$, a type A , a predicate $(x : A) \vdash P$, a term $a : A$ and a proof $p : P[x \mapsto a]$ that the predicate holds for a , we construct the term

$$apd(x.f \$ x) (\text{unspan } A (\Sigma(x : A).P) (x.\pi_1 x), (a, p))$$

which has type

$$\begin{aligned} \forall d(x.El (\pi_1 x)) (\text{unspan } A (\Sigma(x : A).P) (x.\pi_1 x), (a, p)) &= \\ \forall d(x.El x) \left(ap(x.\pi_1 x) (\text{unspan } A (\Sigma(x : A).P) (x.\pi_1 x), (a, p)) \right) &= \\ \forall d(x.El x) (\text{unspan } A (\Sigma(x : A).P) (x.\pi_1 x)) &= \\ \Sigma(x : A).P & \end{aligned}$$

The unary version of our theory suffices (thus $k = 0$). We compute the first projection of this term.

$$\begin{aligned}
 \pi_1 & \left(\text{apd}(x.f \$ x) (\text{unspan } A (\Sigma(x : A).P) (x.\pi_1 x), (a, p)) \right) & = \\
 & \quad (\text{kd}_{x.\text{El } x} (\text{unspan } A_k A t_k) a = t_k[x \mapsto a]) \\
 \text{kd}_{x.\text{El } x} & (\text{unspan } A (\Sigma(x : A).P) (x.\pi_1 x)) & \\
 & \left(\text{apd}(x.f \$ x) (\text{unspan } A (\Sigma(x : A).P) (x.\pi_1 x), (a, p)) \right) & = \\
 & \quad (\text{kd on composition}) \\
 \text{kd}_{x.\text{El } (\pi_1 x)} & (\text{unspan } A (\Sigma(x : A).P) (x.\pi_1 x), (a, p)) & \\
 & \left(\text{apd}(x.f \$ x) (\text{unspan } A (\Sigma(x : A).P) (x.\pi_1 x), (a, p)) \right) & = \\
 & \quad (\text{relation of kd and apd}) \\
 (f \$ x)[x \mapsto k_{\Sigma(y:U).\text{El } y} & (\text{unspan } A (\Sigma(x : A).P) (x.\pi_1 x), (a, p))] & = \\
 & \quad (\text{distributivity of } k) \\
 f \$ (c A, a)
 \end{aligned}$$

Thus the second projection provides

$$\pi_2 \left(\text{apd}(x.f \$ x) (\text{unspan } A (\Sigma(x : A).P) (x.\pi_1 x), (a, p)) \right) : P[x \mapsto f \$ (c A, a)]$$

and we obtain the desired result by choosing the predicate P to be $(x : A) \vdash \text{Eq}_A a x$.

Induction for Church-encoded natural numbers. We can also prove results for higher order polymorphic types using $\text{mk}\forall\Pi$. As an example, we prove the induction principle for Church encoded natural numbers following [Wadler 1990]. The (uncurried) type of Church natural numbers is

$$N = \Pi \left(x : \Sigma \left(y : \Sigma(z : U).\text{El } z \right).\text{El } (\pi_1 y) \rightarrow \text{El } (\pi_1 (\pi_1 y)) \right).\text{El } (\pi_1 (\pi_1 x))$$

A natural number algebra is given by a type A , $z_A : A$ and $s_A : A \rightarrow A$. A morphism of algebras between (A, z_A, s_A) and (B, z_B, s_B) is a function $f : A \rightarrow B$ such that $f \$ z_A = z_B$ and $f \$ (s_A \$ n) = s_B (f n)$ for every n . These form a category. We define:

$$\begin{aligned}
 \text{zero} & := \lambda x.\pi_2 (\pi_1 x) \\
 \text{suc} & := \lambda n.\lambda x.\pi_2 x \$ (n \$ x) \\
 \text{ite}_{(A,z_A,s_A)} & := \lambda n.n \$ (A, z_A, s_A)
 \end{aligned}$$

The induction principle that we want says that $(N, \text{zero}, \text{suc})$ is initial in the category of algebras of N , ite_A being the unique morphism from N to the algebra (A, z_A, s_A) . (This is equivalent to saying that every displayed model over N has a section, see [Kaposi et al. 2019b] for a generic proof.) Using binary parametricity, we first show that ite respects morphisms, that is, $f \$ (\text{ite}_A \$ n) = \text{ite}_B \$ n$. Assuming

$$\begin{aligned}
 n & : N \\
 A, z_A & : A, s_A : A \rightarrow A \\
 B, z_B & : B, s_B : B \rightarrow B \\
 x & : A \times B \vdash Q \\
 z & : Q[x \mapsto (z_A, z_B)] \\
 s & : \Pi(y : \Sigma(x : A \times B).Q).Q[x \mapsto (s_A \$ (\pi_1 (\pi_1 y)), s_B \$ (\pi_1 (\pi_2 y)))]
 \end{aligned}$$

we build

$$\begin{aligned} ((s_A, s_B), s) : \Sigma(x : A \times B).Q &\rightarrow \Sigma(x : A \times B).Q \\ c_2 := \text{unspan } AB(\Sigma(x : A \times B).Q) (y.\pi_1(\pi_1 y)) (y.\pi_2(\pi_1 y)) &: \forall U \\ s^+ := \text{mk}\forall\Pi c_2 s_A s_B ((s_A, s_B), s) &: \forall d(x.EI x \rightarrow EI x) c_2. \end{aligned}$$

Finally we have

$$\pi_2 \left(\text{apd}(x.n \$ x) (c_2, ((z_A, z_B), z), s^+) \right) : Q[x \mapsto (n \$ (c A, z_A, s_A), n \$ (c B, z_B, s_B))].$$

Now specialising this to $Q[x \mapsto (a, b)] = Eq_B(f \$ a) b$ we obtain that ite respects ite_X .

Then we prove that the only function from N to X is ite_X using the fact that ite respects ite_X . This means $ite_X \$ (ite_N \$ n) = ite_X \$ n$, which is the same as $(n \$ (N, zero, suc)) \$ (X, z_X, s_X) = n \$ (X, z_X, s_X)$. Using function extensionality we obtain $n \$ (N, zero, suc) = n$, which means that ite_N is the identity function. We can then conclude by remarking that this gives, for any morphism f from N to A , $f \$ n = f \$ (ite_N \$ n) = ite_A \$ n$.

Compute ap on R using symmetry. One reason for needing symmetry S_A in our syntax is to compute the $\text{ap}(x.R_A x) a_2$ way of turning a witness of a span $a_2 : \forall A$ into a witness of a double span. Without symmetry, there seems to be no way to compute the value of the closed boolean $\text{ap}(x.R_{\text{Bool}} x) \text{ true}$. Having symmetry and the rule $S_A (R_{\forall A} a_2) = \text{ap}(x.R_A x) a_2$ we compute

$$\begin{aligned} \text{ap}(x.R_{\text{Bool}} x) \text{ true} &= S_{\text{Bool}} (R_{\forall \text{Bool}} \text{ true}) = S_{\text{Bool}} (\text{ap}(_.\text{true}) \text{ tt}) = S_{\text{Bool}} (\text{ap}(x_2.\text{ap}(_.\text{true}) x_2) \text{ tt}) = \\ &= \text{ap}(x_2.\text{ap}(_.\text{true}) x_2) (S_{\top} \text{ tt}) = \text{ap}(x_2.\text{ap}(_.\text{true}) x_2) \text{ tt} = \text{ap}(x_2.\text{true}) \text{ tt} = \text{true}. \end{aligned}$$

3 THE GLOBAL THEORY AND ITS PRESHEAF MODEL

In this section, we define our global theory which directly comes from the presheaf model on BCH cubes [Bezem et al. 2013]. We assume basic knowledge of working with models of type theory as categories with families (CwFs, [Castellan et al. 2019]) and the CwF of presheaves [Hofmann 1997].

We define the global theory as a generalised algebraic theory (GAT, [Kapsi et al. 2019b]) by saying what a model of this theory is. First we present the core theory externally, this is the common part of the local and global theories. This corresponds to the core theory listed in Figure 2, with the additional structure of strict democracy which is not expressible internally.

Definition 3.1 (Core theory, externally). A model of the core theory is a CwF with the following features:

- \top, Σ with β and η and strict identity types Eq with uniqueness of identity proofs. This turns the CwF into a finite limit CwF (flCwF, [Kovács and Kapsi 2020]).
- Strict democracy, that is, an operation K that turns a context into a type satisfying the following equations. Note that this includes a sort equation. (Non-strict) democracy requires $\text{Sub } \Gamma \Theta = \text{Tm } \Gamma (K \Theta)$ only up to isomorphism.

$$\begin{array}{ll} K : \text{Con} \rightarrow \text{Ty } \Gamma & K \diamond = \top \\ (K \Theta)[\sigma] = K \Theta & K(\Theta \triangleright A) = \Sigma(K \Theta)(A[q]) \\ \text{Sub } \Gamma \Theta = \text{Tm } \Gamma (K \Theta) & (\sigma_{\triangleright} t) = (\sigma_{\Sigma} t) \end{array}$$

In the equation relating K of \triangleright and Σ , q can be used as a substitution because of the previous equations: $q : \text{Tm}(\Gamma \triangleright K \Theta)(K \Theta[p])$, hence $q : \text{Tm}(\Gamma \triangleright K \Theta)(K \Theta)$, and because of the sort equation we have $q : \text{Sub}(\Gamma \triangleright K \Theta) \Theta$.

- Π types given by an isomorphism $\text{lam} : \text{Tm}(\Gamma \triangleright A)B \cong \text{Tm}\Gamma(\Pi AB) : \text{app}$ natural in Γ . We abbreviate $t \$ u := \text{app } t[id, u]$.
- A hierarchy of universes à la Coquand, that is $U : \text{Ty}\Gamma$ with an isomorphism $c : \text{Ty}\Gamma \cong \text{Tm}\Gamma U : \text{El}$. We don't write universe indices for convenience, but every construction in this paper can be redone in a setting where types, terms and universes are all indexed by levels, and $U_i : \text{Ty}_{i+1}\Gamma$.
- Bool with dependent elimination ite and two β rules (no η).

Now we list the components specific to the global theory (c.f. Figure 1). We use **brick red colour** to distinguish from the operations of the local theory.

Definition 3.2 (Global theory). A model of the core theory is a model of the global theory if it comes equipped with the following structure:

- (1) A strict flCwF endomorphism on the model \mathbb{V} that also preserves K strictly. We denote the four different maps by \mathbb{V}_{Con} , \mathbb{V}_{Sub} , \mathbb{V}_{Ty} and \mathbb{V}_{Tm} or just \mathbb{V} . Strict preservation of K means $\mathbb{V}(K\Theta) = K(\mathbb{V}\Theta)$ and $\mathbb{V}_{\text{Sub}}\{\Gamma\}\{\Theta\} = \mathbb{V}_{\text{Tm}}\{\Gamma\}\{K\Theta\}$.
- (2) Natural transformations k from \mathbb{V} to the identity functor on the CwF, for $k = 0$ and $k = 1$. This means for each $\Gamma : \text{Con}$, a substitution $k_\Gamma : \text{Sub}(\mathbb{V}\Gamma)\Gamma$ with $k_\Gamma \circ \mathbb{V}\sigma = \sigma \circ k_\Delta$. We define the action of k on types as follows.

$$\begin{aligned} k_A &: \text{Tm}(\mathbb{V}\Gamma \triangleright \mathbb{V}A)(A[k_\Gamma \circ p]) \\ k_A &:= q[k_{\Gamma \triangleright A}] \end{aligned}$$

- (3) A natural transformation R from the identity to \mathbb{V} .
- (4) A natural transformation S from $\mathbb{V} \circ \mathbb{V}$ to itself.
- (5) The following five equations relating the above three natural transformations:

$$\begin{aligned} k_\Gamma \circ R_\Gamma &= \text{id}_\Gamma \\ k_{\mathbb{V}\Gamma} \circ S_\Gamma &= \mathbb{V}k_\Gamma \\ S_\Gamma \circ R_{\mathbb{V}\Gamma} &= \mathbb{V}R_\Gamma \\ S_\Gamma \circ S_\Gamma &= \text{id}_{\mathbb{V}(\mathbb{V}\Gamma)} \\ S_{\mathbb{V}\Gamma} \circ \mathbb{V}S_\Gamma \circ S_{\mathbb{V}\Gamma} &= \mathbb{V}S_\Gamma \circ S_{\mathbb{V}\Gamma} \circ \mathbb{V}S_\Gamma \end{aligned}$$

- (6) The map from $\mathbb{V}(\Pi AB)$ to compatible $(\Pi AB[k])$ s and $(\Pi(\mathbb{V}A)(\mathbb{V}B))$ has an inverse $\text{mk}\mathbb{V}\Pi$. Precisely we require the operation $\text{mk}\mathbb{V}\Pi$ with the following equations.

$$\begin{aligned} \text{mk}\mathbb{V}\Pi &: (\sigma : \text{Sub}\Delta(\mathbb{V}\Gamma))(t_k : \text{Tm}\Delta(\Pi AB[k_\Gamma \circ \sigma]))(t : \text{Tm}\Delta(\Pi(\mathbb{V}A)(\mathbb{V}B)[\sigma])) \rightarrow \\ &\quad (t_k[p] \$ (k_A[\sigma \uparrow]) = k_B[\sigma \uparrow, \text{app } t]) \rightarrow \text{Tm}\Delta(\mathbb{V}(\Pi AB)[\sigma]) \\ \text{mk}\mathbb{V}\Pi \sigma t_k t[\rho] &= \text{mk}\mathbb{V}\Pi(\sigma \circ \rho)(t_k[\rho])(t[\rho]) \\ k_{\Pi AB}[\sigma, \text{mk}\mathbb{V}\Pi \sigma t_k t] &= t_k \\ \text{lam}(\mathbb{V}(\text{app } q))[\sigma, \text{mk}\mathbb{V}\Pi \sigma t_k t] &= t \\ \text{mk}\mathbb{V}\Pi \sigma(k_{\Pi AB}[\sigma, t_2]) (\text{lam}(\mathbb{V}(\text{app } q))[\sigma, t_2]) &= t_2 \end{aligned}$$

When we quantify over a subscripted variable such as t_k , we mean to quantify over *two* variables t_0 and t_1 with types obtained by substituting $k = 0, 1$ in the type of t_k .

(7) The map from VU to a span has a section called unspan . Precisely we require:

$$\begin{aligned}\text{unspan} : (A_k A : \text{Ty } \Gamma) &\rightarrow \text{Tm}(\Gamma \triangleright A) (A_k[\text{p}]) \rightarrow \text{Tm } \Gamma (\text{VU}[\epsilon]) \\ \text{unspan } A_k A t_k[\sigma] &= \text{unspan } (A_k[\sigma]) (A[\sigma]) (t_k[\sigma \uparrow]) \\ \text{El } (\text{k}_U)[\epsilon, \text{unspan } A_k A t_k] &= A_k \\ \forall (\text{El } q)[\epsilon, \text{unspan } A_k A t_k] &= A \\ k_{\text{El } q}[(\epsilon, \text{unspan } A_k A t_k) \uparrow] &= t_k\end{aligned}$$

We do not require the other roundtrip equation.

(8) V preserves Bool strictly.

We define the BCH cube category as depicted in Figure 1.

Definition 3.3 (Cube category \square). The objects of the category are natural numbers, the morphisms are generated by the following quotient inductive set. First of all we have a category, that is, composition and id with the categorical laws, then we have the following constructors and equality constructors. When we write k we always mean both a copy for 0 and a copy for 1.

$$\begin{array}{lll}suc : \square(J, I) \rightarrow \square(1 + J, 1 + I) & suc(f \circ g) = suc f \circ suc g & suc \text{id} = \text{id} \\ k_I : \square(I, 1 + I) & k_I \circ f = suc f \circ k_J & \\ R_I : \square(1 + I, I) & R_I \circ suc f = f \circ R_J & \\ S_I : \square(2 + I, 2 + I) & S_I \circ suc(suc f) = suc(suc f) \circ S_J & \\ & R_I \circ k_I = \text{id}_I & \\ & S_I \circ k_{1+I} = suc k_I & \\ & R_{1+I} \circ S_I = suc R_I & \\ & S_I \circ S_I = \text{id}_{2+I} & \\ & S_{1+I} \circ suc S_I \circ S_{1+I} = suc S_I \circ S_{1+I} \circ suc S_I & \end{array}$$

Comment on the definition of \square . Another way to describe this category is that it is the free symmetric semicartesian strict monoidal category over a cylinder. In particular, the final equation on S is the “braid equation”, which together with naturality forms a presentation of the symmetric group; thus the automorphisms of I are the permutations of an I -element set. This category also has a named variant, in which the objects are finite sets and a morphism from J to I is a function from J to $I + \{0, 1\}$ that is injective on the subset of J that is not mapped to inr .

S_I allows us to swap the first two dimensions, but we would like to swap the first with any other dimension, on both directions.

Definition 3.4 (Generalised symmetries in \square). We define the following morphisms by induction on natural numbers.

$$\begin{array}{ll}(sym_{I_0,1}|\text{id}_{I_1}) & : \square(I_0 + 1 + I_1, 1 + I_0 + I_1) \\ (sym_{0,1}|\text{id}_{I_1}) & := \text{id}_{1+I_1} \\ (sym_{1+I_0,1}|\text{id}_{I_1}) & := S_{I_0+I_1} \circ suc(sym_{I_0,1}|\text{id}_{I_1})\end{array}\quad\begin{array}{ll}(sym_{1,I_0}|\text{id}_{I_1}) & : \square(1 + I_0 + I_1, I_0 + 1 + I_1) \\ (sym_{1,0}|\text{id}_{I_1}) & := \text{id}_{1+I_1} \\ (sym_{1,1+I_0}|\text{id}_{I_1}) & := suc(sym_{1,1+I_0}|\text{id}_{I_1}) \circ S_{I_0+I_1}\end{array}$$

By induction we prove that the generalised symmetries form an isomorphism in \square :

$$(sym_{I_0,1}|\text{id}_{I_1}) \circ (sym_{1,I_0}|\text{id}_{I_1}) = \text{id}_{1+I_0+I_1} \quad \text{and} \quad (sym_{1,I_0}|\text{id}_{I_1}) \circ (sym_{I_0,1}|\text{id}_{I_1}) = \text{id}_{I_0+1+I_1}$$

We would like to characterise the morphisms in \square without equations, so that we can have a powerful case analysis (which will be needed for defining the V -preservation of Π and U). We will need

a case analysis on a morphism $f : \square(J, 1 + I)$ to know where the 1 in the codomain is coming from: either it is coming from a face map k_I and $f = k_I \circ f'$ or $J = J_0 + 1 + J_1$ for some J_0 and J_1 and the 1 in the codomain comes from the 1 in the domain, with the rest being mapped by some $f' : \square(J_0 + J_1, I)$. We generalise so that the 1 in the codomain can be in the middle.

PROBLEM 3.5 (CASE ANALYSIS ON MORPHISMS IN \square). *For a morphism $f : \square(J, I_0 + 1 + I_1)$ either*

- (i) *there is a k and an $f' : \square(J, I_0 + I_1)$ such that $f = (\text{sym}_{1, I_0}|\text{id}_{I_1}) \circ k_{I_0+I_1} \circ f'$,*
- (ii) *or there are J_0, J_1 with $J = J_0 + 1 + J_1$ and $f' : \square(J_0 + J_1, I_0 + I_1)$ such that $f = (\text{sym}_{1, I_0}|\text{id}_{I_1}) \circ \text{suc } f' \circ (\text{sym}_{J_0, 1}|\text{id}_{J_1})$.*

CONSTRUCTION. The first case means that the 1 in the codomain of the morphism comes from a face map k , the second case means that the 1 in the codomain comes from a 1 in the domain.

We perform induction by building a displayed model of the quotient inductive set in Definition 3.3, and then using the induction principle of the quotient inductive set [Kaposi et al. 2019b]. Displayed morphisms are given by

$$\begin{aligned} \square^*(J, I) f &:= \{I_0 I_1 : \square\} \{I = I_0 + 1 + I_1\} \rightarrow (\exists k. (f' : \square(J, I_0 + I_1)) \times f = (\text{sym}_{1, I_0}|\text{id}_{I_1}) \circ k_{I_0+I_1} \circ f') + \\ &\quad ((J_0 : \square) \times (J_1 : \square) \times (f' : \square(J_0 + J_1, I_0 + I_1)) \times f = (\text{sym}_{1, I_0}|\text{id}_{I_1}) \circ \text{suc } f' \circ (\text{sym}_{J_0, 1}|\text{id}_{J_1})). \end{aligned}$$

Displayed composition is given by

$$\begin{aligned} \text{inl}(k, f') \circ^* g^* &:= \text{inl}(k, f' \circ g) \\ \text{inr}(J_0, J_1, f') \circ^* \text{inl}(l, g') &:= \text{inl}(l, f' \circ g') \\ \text{inr}(J_0, J_1, f') \circ^* \text{inr}(K_0, K_1, g') &:= \text{inr}(K_0, K_1, f' \circ g'). \end{aligned}$$

We would like to define $f^* \circ^* g^*$ which says our induction motive for $f \circ g$. We match on the induction hypothesis for f denoted f^* : if it is an inl , that is, $f = (\text{sym}_{1, I_0}|\text{id}_{I_1}) \circ k_{I_0+I_1} \circ f'$, then we have $f \circ g = (\text{sym}_{1, I_0}|\text{id}_{I_1}) \circ k_{I_0+I_1} \circ f' \circ g$, so we still know that the 1 in the codomain of $f \circ g$ comes from k . If f^* is an inr (that is, the 1 in the codomain comes from a 1 in the domain of f), then we have to match on g^* to learn whether that 1 in the codomain of g comes from k or from the domain of g . We then check that $- \circ^* -$ satisfies associativity.

For the rest of the constructors of \square , we list the computational parts of the displayed model.

$$\begin{aligned} \text{id}^* \{I_0\} \{I_1\} &:= \text{inr}(I_0, I_1, \text{id}) \\ \text{suc}^* f^* \{0\} \{I\} &:= \text{inr}(0, J, f) \\ \text{suc}^* f^* \{1 + I_0\} \{I_1\} &:= \text{inl}(k, \text{suc } f'), \text{ if } f^* \{I_0\} \{I_1\} = \text{inl}(k, f') \\ \text{suc}^* f^* \{1 + I_0\} \{I_1\} &:= \text{inl}(1 + J_0, J_1, \text{suc } f'), \text{ if } f^* \{I_0\} \{I_1\} = \text{inr}(J_0, J_1, f') \\ k^* \{0\} \{I\} &:= \text{inl}(k, \text{id}_I) \\ k^* \{1 + I_0\} \{I_1\} &:= \text{inr}(I_0, I_1, k_{I_0+I_1}) \\ R^* \{I_0\} \{I_1\} &:= \text{inr}(1 + I_0, I_1, R_{I_0+I_1}) \\ S^* \{0\} \{1 + I\} &:= \text{inr}(1, I, \text{id}_{1+I}) \\ S^* \{1\} \{I\} &:= \text{inr}(0, 1 + I, \text{id}_{1+I}) \\ S^* \{2 + I'_0\} \{I'_1\} &:= \text{inr}(2 + I'_0, I'_1, S_{I'_0+I'_1}) \end{aligned}$$

The displayed versions of the equations all hold. The displayed model induces a dependent function from $\square(J, I_0 + 1 + I_1)$ to the sum type by the induction principle of the quotient inductive set \square . \square

This case analysis will be used in parts (6) and (7) of Problem 3.7.

Construction 3.6 (Presheaf model of the core theory). We recall the computational parts of the presheaf model over \square [Hofmann 1997] providing a model of the core theory (Definition 3.1).

A $\Gamma : \text{Con}$ is a presheaf, that is a family of sets $\Gamma : \square \rightarrow \text{Set}$ together with reindexing $\gamma_I[f]_\Gamma : \Gamma J$ for $\gamma_I : \Gamma I$ and $f : \square(J, I)$ such that $\gamma_I[f \circ g] = \gamma_I[f][g]$ and $\gamma_I[\text{id}] = \gamma_I$. A $\sigma : \text{Sub } \Delta \Gamma$ is a polymorphic function $\sigma : \{I : \square\} \rightarrow \Delta I \rightarrow \Gamma I$ with $(\sigma \delta_I)[f]_\Gamma = \sigma(\delta_I[f])_\Delta$. A type $A : \text{Ty } \Gamma$ is a dependent presheaf, that is $A : (I : \square) \rightarrow \Gamma I \rightarrow \text{Set}$ equipped with $-[-]_A : A I \gamma_I \rightarrow (f : \square(J, I)) \rightarrow A J (\gamma_I[f])$ with two equations. Finally a term is a dependent natural transformation $t : (\gamma_I : \Gamma I) \rightarrow A I \gamma_I$ with $t \gamma_I[f]_A = t(\gamma_I[f])_\Gamma$. Context extension and the types \top, Σ, Eq are defined pointwise and we have strict democracy by $\text{K } \Theta I \gamma_I = \Theta I$ and $\theta_I[f]_{\text{K } \Theta} = \theta_I[f]_\Theta$. The Yoneda embedding is a functor from \square to presheaves over \square , we denote it by $y : \square \rightarrow \text{Con}$ and $y : \square(J, I) \rightarrow \text{Sub}(y J)(y I)$, and we have the Yoneda lemma $y|_\Gamma : \Gamma I \rightarrow \text{Sub}(y I) \Gamma$ defined by $y|_\Gamma \gamma_I f = \gamma_I[f]_\Gamma$. Function space is defined as $\Pi A B I \gamma_I = \text{Tm}(y I \triangleright A[y \gamma_I])(B[y \gamma_I \uparrow])$ with $t[f]_{\Pi AB} = t[y f \uparrow], \text{lam } t \gamma_I = t[y \gamma_I \uparrow]$ and $\text{app } t(\gamma_I, a_I) = t \gamma_I(\text{id}_I, a_I)$. The universe is defined as $\text{U } I \gamma_I = \text{Ty}(y I)$ with $a[f]_\text{U} := a[y f]$. Decoding is $\text{El } a I \gamma_I = a \gamma_I I \text{id}_I$ with $u_I[f]_{\text{El } a} = u_I[f]_{a \gamma_I}$, and encoding is $c A \gamma_I = A[y \gamma_I]$. We have $\text{El } a[y \gamma_I] = a \gamma_I$. Bool is pointwise.

PROBLEM 3.7 (PRESHEAF MODEL OF THE GLOBAL THEORY). *The presheaf model extends to the operations and equations of the global theory (Definition 3.2). This justifies the notion of the global theory, which was in turn extracted from this exact presheaf model.*

CONSTRUCTION. Recall the two diagrams of Figure 1. We define the components in order.

(1) \forall is defined as precomposition with suc :

$$\begin{array}{lll} \forall \Gamma I & := \Gamma(1+I) & \forall A I \gamma_{1+I} := A(1+I)\gamma_{1+I} \\ \gamma_{1+I}[f]_{\forall \Gamma} & := \gamma_{1+I}[\text{suc } f]_\Gamma & a_{1+I}[f]_{\forall A} := a_{1+I}[\text{suc } f]_A \\ & & \forall t \gamma_{1+I} := t \gamma_{1+I} \end{array}$$

This preserves the flCwF structure strictly, e.g. we have $\forall(\Gamma \triangleright A) I = (\Gamma \triangleright A)(1+I) = (\gamma_{1+I} : \Gamma(1+I) \times A(1+I)\gamma_{1+I}) = (\gamma_{1+I} : \forall \Gamma I) \times \forall A I \gamma_{1+I} = (\forall \Gamma \triangleright \forall A) I$, strict preservation of K is by $\forall(\text{K } \Theta) I \gamma_{1+I} = \text{K } \Theta(1+I)\gamma_{1+I} = \Theta(1+I) = \forall \Theta I = \text{K}(\forall \Theta) I \gamma_{1+I}$ and by the fact that \forall_{Sub} and \forall_{Tm} coincide.

- (2) k is defined as $k_\Gamma \gamma_{1+I} := \gamma_{1+I}[k_I]_\Gamma$. This is natural as $(k_\Gamma \circ \forall \sigma) \delta_I = (\sigma \delta_I)[k_I]_\Gamma \stackrel{\sigma \text{ natural}}{=} \sigma(\delta_I[k_I]\Delta) = (\sigma \circ k_\Delta) \delta_I$
- (3) $R_\Gamma \gamma_I := \gamma_I[R_I]_\Gamma$
- (4) $S_\Gamma \gamma_{2+I} := \gamma_{2+I}[S_I]_\Gamma$
- (5) The five equations follow from their corresponding equations in \square . For example, $(k_\Gamma \circ R_\Gamma) \gamma_I = \gamma_I[R_I]_\Gamma[k_I]_\Gamma = \gamma_I[R_I \circ k_I]_\Gamma = \gamma_I[\text{id}_I]_\Gamma = \gamma_I = \text{id}_\Gamma \gamma_I$.
- (6) $\text{mkVII } \sigma t_k t \delta_I$ is in $\forall(\Pi A B) I (\sigma \delta_I)$, that is, in $\text{Tm}(y(1+I) \triangleright A[y|_\Gamma (\sigma \delta_I)])(B[(y|_\Gamma (\sigma \delta_I)) \uparrow])$. We do case analysis (Problem 3.5) on the input morphism given by Yoneda $y(1+I)$:

$$\begin{array}{lll} \text{mkVII } \sigma t_k t \delta_I \{J\} & (k_I \circ f, a_I) & := \text{app } t_k(\delta_I[f]_\Delta, a_I) \\ \text{mkVII } \sigma t_k t \delta_I \{J_0 + 1 + J_1\} ((\text{suc } f \circ (\text{sym}_{J_0, 1} | \text{id}_{J_1})), a_{J_0+1+J_1}) & & \\ & \text{app } t(\delta_I[f]_\Delta, a_{J_0+1+J_1}[(\text{sym}_{1, J_0} | \text{id}_{J_1})]_A) [\text{sym}_{J_0, 1} | \text{id}_{J_1}]_B & \end{array}$$

If the morphism selects a projection k , then we use the map at the base of the span t_k . If the morphism selects an existing dimension (1 in the middle of $J_0 + 1 + J_1$), then we use the function at the apex t , but we have to apply symmetry to the input to make it well-typed. $\text{app } t$ requires an input in $\forall A[\sigma](J_0 + J_1)(\delta_I[f]) = A(1+J_0+J_1)(\sigma(\delta_I[f]))$, but our $a_{J_0+1+J_1}$ is in $A(J_0+1+J_1)(\sigma \delta_I[\text{suc } f \circ (\text{sym}_{J_0, 1} | \text{id}_{J_1})])$. Then we have to apply symmetry at the output again: $\text{app } t(\delta_I[f]_\Delta, a_{J_0+1+J_1}[(\text{sym}_{1, J_0} | \text{id}_{J_1})]_A)$ is in $B(1+J_0+J_1)(\sigma(\delta_I[f]), a_{J_0+1+J_1}[\text{sym}_{1, J_0} | \text{id}_{J_1}])$, but we need an element of $B(J_0 + 1 + J_1)(\sigma \delta_I[\text{suc } f \circ (\text{sym}_{J_0, 1} | \text{id}_{J_1})], a_{J_0+1+J_1})$.

This $\text{mk}\forall\Pi$ operation is natural in three different senses. From naturality of t_k and t and from the equation $t_k[p] \$ (\textcolor{red}{k}_A[\sigma \uparrow]) = \textcolor{red}{k}_B[\sigma \uparrow, \text{app } t]$, we get $\text{mk}\forall\Pi \sigma t_k t \delta_I(f, a_J)[g]_B = \text{mk}\forall\Pi \sigma t_k t \delta_I(f \circ g, a_J[g]_A)$. The naturalities $\text{mk}\forall\Pi \sigma t_k t \delta_I[f]_{\text{V}(\text{ILAB})} = \text{mk}\forall\Pi \sigma t_k t (\delta_I[f]_\Delta)$ and $\text{mk}\forall\Pi \sigma t_k t [\rho] = \text{mk}\forall\Pi (\sigma \circ \rho)(t_k[\rho])(t[\rho])$ are also easy consequences.

The three equations which express the roundtrips follow by unfolding the definitions.

- (7) $\text{unspan } A_k A t_k \gamma_I$ is in $\forall I \gamma_I = \text{Ty}(y(1 + I))$. We define this type by case analysis on the morphism given by Yoneda.

$$\begin{aligned} \text{unspan } A_k A t_k \gamma_I J &= (k_I \circ f) & := A_k J(\gamma_I[f]) \\ \text{unspan } A_k A t_k \gamma_I (J_0 + 1 + J_1) (\text{suc } f \circ (\text{sym}_{J_0,1}[\text{id}_{J_1}])) &:= A (J_0 + J_1) (\gamma_I[f]) \end{aligned}$$

The restriction operation on this type is given by restriction in A_k or A , unless the morphism takes us from A to A_k : in this case we apply t_k .

$$\begin{aligned} a_{k_J}[g]_{\text{unspan } A_k A t_k \gamma_I} &:= a_{k_J}[g]_{A_k} \\ a_{J_0+J_1}[(\text{sym}_{1,J_0}[\text{id}_{J_1}] \circ k_{J_0+J_1} \circ g)]_{\text{unspan } A_k A t_k \gamma_I} &:= t_k(\gamma_I[f \circ g], a_{J_0+J_1}[g]_A) \\ a_{J_0+J_1}[(\text{sym}_{1,J_0}[\text{id}_{J_1}] \circ \text{suc } g \circ (\text{sym}_{K_0,1}[\text{id}_{K_1}]))]_{\text{unspan } A_k A t_k \gamma_I} &:= a_{J_0+J_1}[g]_A \end{aligned}$$

In the second case we expected a result in the set

$$\begin{aligned} \text{unspan } A_k A t_k \gamma_I K (\text{suc } f \circ (\text{sym}_{J_0,1}[\text{id}_{J_1}] \circ (\text{sym}_{1,J_0}[\text{id}_{J_1}] \circ k_{J_0+J_1} \circ g))) &= \\ \text{unspan } A_k A t_k \gamma_I K (\text{suc } f \circ k_{J_0+J_1} \circ g) &= \\ \text{unspan } A_k A t_k \gamma_I K (k_I \circ f \circ g) &= \\ A_k K(\gamma_I[f \circ g]), \end{aligned}$$

this is why we used the leg of the span t_k .

The equation $\text{El}(\textcolor{red}{k}_U)[\epsilon, \text{unspan } A_k A t_k] = A_k$ holds by definition of $\textcolor{red}{k}_U$ which becomes $-[k_I]_U$. The equation $\forall(\text{El } q)[\epsilon, \text{unspan } A_k A t_k] = A$ also holds by definition, here we rely on the fact that El applies the id morphism to the code for the type inside. Finally, $\textcolor{red}{k}_{\text{El } q}([\epsilon, \text{unspan } A_k A t_k]) = t_k$ comes from the definition of restriction for $\text{unspan } A_k A t_k \gamma_I$ which applies t_k for morphisms ending in k .

Note that this model *does not* justify the other roundtrip, so unspan is not an isomorphism, just a section. Given an $a_2 : \text{Tm } \Gamma(\forall U[\epsilon])$, we don't necessarily have

$$\text{unspan } (\text{El } \textcolor{red}{k}_U[\epsilon, a_2]) (\forall(\text{El } q)[\epsilon, a_2]) (\textcolor{red}{k}_{\text{El } q}[(\epsilon, a_2) \uparrow]) = a_2.$$

Given a $\gamma_I : \Gamma I$ we have that $a_2 \gamma_I$ is in $\text{Ty}(y(1 + I))$, but

$$\begin{aligned} \text{unspan } (\dots) (\forall(\text{El } q)[\epsilon, a_2]) (\dots) \gamma_I (J_0 + 1 + J_1) (\text{suc } f \circ (\text{sym}_{J_0,1}[\text{id}_{J_1}])) &= \\ \forall(\text{El } q)[\epsilon, a_2] (J_0 + J_1) (\gamma_I[f]) &= \\ \text{El } q[\epsilon, a_2] (1 + J_0 + J_1) (\gamma_I[f]) &= \\ a_2 (\gamma_I[f]_\Gamma) (1 + J_0 + J_1) \text{id} &= \\ (a_2 \gamma_I)[f]_{\text{VU}} (1 + J_0 + J_1) \text{id} &= \\ (a_2 \gamma_I)[\text{suc } f]_{\text{U}} (1 + J_0 + J_1) \text{id} &= \\ (a_2 \gamma_I)[y(\text{suc } f)] (1 + J_0 + J_1) \text{id} &= \\ a_2 \gamma_I (1 + J_0 + J_1) (\text{suc } f) &\neq \\ a_2 \gamma_I (J_0 + 1 + J_1) (\text{suc } f \circ (\text{sym}_{J_0,1}[\text{id}_{J_1}])). \end{aligned}$$

Note the inequality in the penultimate line.

- (8) Preservation of booleans is trivial. □

4 ISOMORPHISM OF THE LOCAL AND GLOBAL SYNTAXES

In this section we construct a model of the local theory using the global syntax and show that it is isomorphic to the local syntax. And similarly, we construct a model of the global theory using the local syntax.

4.1 Defining the Local Syntax Using the Global Syntax

The types in the local theory correspond to the types in a fixed context Γ of the core theory. To define the local \forall from the global \forall , we apply \forall and apply the substitution that goes back to Γ :

$$\begin{aligned}\forall : \text{Ty } \Gamma &\rightarrow \text{Ty } \Gamma \\ \forall A &= (\forall A)[R_\Gamma]\end{aligned}$$

All the other operations follow the same idea:

$$\begin{array}{ll} \text{ap} : \text{Tm } (\Gamma \triangleright A) (B[p]) \rightarrow \text{Tm } (\Gamma \triangleright \forall A) (\forall B[p]) & k_- : (A : \text{Ty } \Gamma) \rightarrow \text{Tm } (\Gamma \triangleright \forall A) (A[p]) \\ \text{ap } t := (\forall t)[R_\Gamma \uparrow] & k_A := q[k_{\Gamma \triangleright A}][R_\Gamma \uparrow] \\ \forall d : \text{Ty } (\Gamma \triangleright A) B \rightarrow \text{Ty } (\Gamma \triangleright \forall A) (\forall B) & R_- : (A : \text{Ty } \Gamma) \rightarrow \text{Tm } (\Gamma \triangleright A) (\forall A[p]) \\ \forall d B := (\forall B)[R_\Gamma \uparrow] & R_A := q[R_{\Gamma \triangleright A}] \\ \text{apd} : \text{Tm } (\Gamma \triangleright A) B \rightarrow \text{Tm } (\Gamma \triangleright \forall A) (\forall d B) & S_- : (A : \text{Ty } \Gamma) \rightarrow \text{Tm } (\Gamma \triangleright \forall (\forall A)) (\forall (\forall A)[p]) \\ \text{apd } t := (\forall t)[R_\Gamma \uparrow] & S_A := q[S_{\Gamma \triangleright A}][R_{\forall \Gamma} \circ R_\Gamma \uparrow] \end{array}$$

The equations of the local theory follow from the corresponding equations in the global theory and some naturality principles. We give an example:

$$\begin{aligned}k_A[p, R_A] &= q[k_{\Gamma \triangleright A}][R_\Gamma \uparrow][p, q[R_{\Gamma \triangleright A}]] \\ &= q[k_{\Gamma \triangleright A}][R_\Gamma \circ p, q[R_{\Gamma \triangleright A}]] \\ &= q[k_{\Gamma \triangleright A}][p \circ R_{\Gamma \triangleright A}, q[R_{\Gamma \triangleright A}]] \\ &= q[k_{\Gamma \triangleright A}][R_{\Gamma \triangleright A}] \\ &= q && \text{(naturality of } R\text{)} \\ &&& \text{(corresponding global equation)}\end{aligned}$$

`unspan` is simply `unspan` and `mkVΠ` $t_k t := \text{mkVΠ } (R_\Gamma \uparrow) t_k t$.

4.2 Defining the Global Syntax Using the Local Syntax

We have an operation \forall on types and we want to extend it to contexts. Since the syntax is contextual, it is natural to do an induction to define \forall . In fact, since we have strict democracy, we have another natural way to proceed which is to apply \forall on the type corresponding to the context, so we need a mutual induction to make the link between these two, by defining an isomorphism between them.

$$\begin{array}{ll} \forall : \text{Con} \rightarrow \text{Con} & \forall^\cong : (\Gamma : \text{Con}) \rightarrow \forall \Gamma \cong \diamond \triangleright \forall(\text{K}\Gamma) \\ \forall \diamond := \diamond & \forall^\cong \diamond := (\epsilon, \text{id}) \\ \forall(\Gamma \triangleright A) := \forall \Gamma \triangleright \forall d(A[q])[\forall^\cong \Gamma] & \forall^\cong(\Gamma \triangleright A) := (\epsilon, (q[\forall^\cong \Gamma \circ p], q)) \\ & \forall^{\cong -1}(\Gamma \triangleright A) := (\forall^{\cong -1} \Gamma \circ (p, \pi_1 q), \pi_2 q) \end{array}$$

Here we used $q : \text{Tm } (\diamond \triangleright \text{K}\Gamma) (\text{K}\Gamma)$ as a substitution $\text{Sub } (\diamond \triangleright \text{K}\Gamma) \Gamma$, as a consequence of strict democracy. In fact, it is one direction of an isomorphism:

$$q : \diamond \triangleright \text{K}\Gamma \cong \Gamma : (\epsilon, \text{id})$$

We also have to do an induction for substitutions.

$$\begin{aligned}\textcolor{red}{V} : \{\Gamma : \text{Con}\}(\sigma : \text{Sub } \Delta \Gamma) &\rightarrow \text{Sub } (\textcolor{red}{V} \Delta) (\textcolor{red}{V} \Gamma) \\ \textcolor{red}{V} \{\diamond\} \quad \sigma &:= \epsilon \\ \textcolor{red}{V} \{\Gamma \triangleright A\} (\sigma, t) &:= (\textcolor{red}{V} \{\Gamma\} \sigma, \text{apd } (t[q]) [\textcolor{red}{V}^{\cong} \Delta])\end{aligned}$$

We can also prove a naturality property for $\textcolor{red}{V}^{\cong}$:

$$\textcolor{red}{V}^{\cong} : (\Gamma : \text{Con})(\sigma : \text{Sub } \Delta \Gamma) \rightarrow \textcolor{red}{V}^{\cong} \Gamma \circ \textcolor{red}{V} \sigma = (\text{p}, \text{ap}(\sigma[q])) \circ \textcolor{red}{V}^{\cong} \Delta$$

The definition of $\textcolor{red}{V}$ on types and terms is enforced:

$$\textcolor{red}{V} A := \text{vd}(A[q]) [\textcolor{red}{V}^{\cong} \Gamma] \qquad \qquad \textcolor{red}{V} t := \text{apd } (t[q]) [\textcolor{red}{V}^{\cong} \Delta]$$

Finally we define:

$$k_{\Gamma} := k_{K\Gamma} [\textcolor{red}{V}^{\cong} \Gamma] \qquad \textcolor{red}{R}_{\Gamma} := \textcolor{red}{V}^{\cong -1} \Gamma \circ (\epsilon, \text{ap } p[\text{id}, \text{tt}]) \qquad \textcolor{red}{S}_{\Gamma} := q \circ (p, S_{K\Gamma}) \circ (\epsilon, \text{id})$$

and $\text{mkV}\Pi \sigma t_k t := \text{mkV}\Pi (q[\textcolor{red}{V}^{\cong} \Gamma \circ \sigma]) t_k t$. Once again, `unspan` is straightforward, and the equations come from the corresponding equations in the local theory.

4.3 Roundtrips

The 9 operations of the local syntax coincide with the same ones after globalising and localising them. The proofs essentially rely on manipulations on substitutions and naturality properties.

In addition, the 9 operations of the global syntax coincide with the same ones after localising and globalising them. To prove that the $\textcolor{red}{V}'$ that we define by induction is the same as the initial $\textcolor{red}{V}$, we do a mutual induction proving that $\textcolor{red}{V}^{\cong} \Gamma = (\epsilon, \text{id})$. The proof for substitutions is also an induction, the others rely on naturality.

4.4 Isomorphism

Subsection 4.1 says that the global syntax Syn is also a model of the local theory. Thus we obtain a map α from the local syntax to this global model. Note that it is identity on the core calculus (Figure 2 or Definition 3.1). Similarly, Subsection 4.2 decorates the local syntax with a model of the global syntax providing a morphism β from Syn to the local syntax. By induction by Syn and Syn , we prove the compositions of α and β are identities. The only nontrivial cases of this induction are handled in Subsection 4.3. Thus we obtain e.g. that for all $\Gamma : \text{Con}_{\text{Syn}}$, $\beta(\alpha \Gamma) = \Gamma$, and so on.

5 GLUING FOR THE GLOBAL THEORY

In this section we extend the gluing proof of [Kaposi et al. 2019a] to the global theory. We first define the notion of weak morphism which also has to respect $\textcolor{red}{V}$, then we define the gluing model. Finally we show how to make use of gluing by defining a global section functor from the syntax of the global theory Syn to the presheaf model $\text{PSh}(\square)$ (Problem 3.7), and applying gluing to this, which shows that our theory enjoys canonicity.

5.1 Weak Morphism of Models Respecting $\textcolor{red}{V}$

A weak morphism is a functor which has an action on types and terms and preserves instantiation of substitution strictly and we require that the empty context and context extension are preserved up to isomorphism. Furthermore, we require that the functor preserves $\textcolor{red}{V}$, k , R and S strictly.

Definition 5.1 (Weak morphisms of models of the global theory). Given two models of the global theory C and D , a weak morphism F from C to D contains the following functions and satisfies

the following equations. We usually omit the subscripts saying which model is meant because it is clear from the context, and we also overload the names for the four maps in F .

Maps	Preservation of core structure	Preservation of \vee, k, R, S
$F : \text{Con}_C \rightarrow \text{Con}_{\mathcal{D}}$	$F(\sigma \circ \rho) = F\sigma \circ F\rho$	$F(\vee \Gamma) = \vee(F\Gamma)$
$F : \text{Sub } \Delta \Gamma \rightarrow \text{Sub}(F\Delta)(F\Gamma)$	$F\text{id} = \text{id}$	$F(\vee \sigma) = \vee(F\sigma)$
$F : \text{Ty } \Gamma \rightarrow \text{Ty}(F\Gamma)$	$F_{\epsilon}^{-1} : \text{Sub } \diamond(F\diamond)$	$F(\vee A) = \vee(FA)$
$F : \text{Tm } \Gamma A \rightarrow \text{Tm}(F\Gamma)(FA)$	$F_{\epsilon}^{-1} \circ \epsilon = \text{id}_F \diamond$ $F(A[\sigma]) = FA[F\sigma]$ $F(t[\sigma]) = Ft[F\sigma]$	$F(\vee t) = \vee(Ft)$ $Fk_{\Gamma} = k_{F\Gamma}$ $F R_{\Gamma} = R_{F\Gamma}$
	$F_{\triangleright}^{-1} : \text{Sub}(F\Gamma \triangleright FA)(F(\Gamma \triangleright A))$ $F_{\triangleright}^{-1} \circ (Fp, Fq) = \text{id}$ $(Fp, Fq) \circ F_{\triangleright}^{-1} = \text{id}$	$F S_{\Gamma} = S_{F\Gamma}$

For any such F we can define comparison maps expressing that K, T, Σ, Eq are preserved automatically up to isomorphism, Π and U are preserved in a lax way and $Bool$ in an oplax way. The specification is on the left, the implementations are on the right.

$$\begin{array}{lll}
 F_K : F(K\Theta) \cong K(F\Theta) & F_K := Fq \circ F_{\triangleright}^{-1} \\
 F_{\Sigma} : F(\Sigma AB) \cong \Sigma(FA)(FB[F_{\triangleright}^{-1}]) & F_{\Sigma} := (F(\pi_1 q), F(\pi_2 q))[F_{\triangleright}^{-1}] \\
 F_{Eq} : F(Eq_A u v) \cong Eq_{FA}(Fu)(Fv) & F_{Eq} := \text{refl}[F_{\triangleright}^{-1}] \\
 F_{\Pi} : \text{Tm}(F\Gamma \triangleright F(\Pi AB))(\Pi(FA)(FB[F_{\triangleright}^{-1}]])[p]) & F_{\Pi} := \text{lam}(F(\text{app } q)[F_{\triangleright}^{-1}])[F_{\triangleright}^{-1}] \\
 F_U : \text{Tm}(F\Gamma \triangleright FU) U & F_U := c(F(\text{El } q)[F_{\triangleright}^{-1}]) \\
 F_{Bool}^{-1} : \text{Tm}(F\Gamma \triangleright Bool)(FBool[p]) & F_{Bool}^{-1} := \text{ite } q(F\text{true}[p])(F\text{false}[p])
 \end{array}$$

We note that a consequence of preservation of \vee is that $\vee F_{\triangleright}^{-1} = F_{\triangleright}^{-1}$.

5.2 The Gluing Displayed Model

A displayed model over a base model is equivalent to a model with a morphism into the base. It can also be seen as the collection of motives and methods for the induction principle of the syntax (the initial model). The components of a displayed model can be computed for any generalised algebraic theory using the methods in [Kaposi et al. 2019b]. We mark displayed components with a bullet. For example, displayed contexts are a set indexed over contexts: $\text{Con}^{\bullet} : \text{Con} \rightarrow \text{Set}$ where Con is the set of contexts of the base model. Displayed types are indexed implicitly over a base context, a displayed context at the base context and a base type: $\text{Ty}^{\bullet} : \{\Gamma : \text{Con}\} \rightarrow \text{Con}^{\bullet}\Gamma \rightarrow \text{Ty } \Gamma \rightarrow \text{Set}$. The displayed variants of equations are well-typed because of the corresponding equations in the base model.

PROBLEM 5.2 (GLUING). *Given a weak morphism F from C to \mathcal{D} , we construct a displayed model of the global theory over C .*

CONSTRUCTION. Gluing of a weak morphism from C to \mathcal{D} gives a displayed model over C . This model construction is a generalisation of logical predicates. A displayed context is a predicate over F applied to the context, a displayed type is a dependent predicate over F applied to the type. A

displayed substitution/term says expresses that F of the substitution/term respects the predicates.

$$\begin{array}{ll} \text{Con}^\bullet \Gamma & := \text{Ty}(F\Gamma) \\ \text{Sub}^\bullet \Delta^\bullet \Gamma^\bullet \sigma & := \text{Tm}(F\Delta \triangleright \Delta^\bullet)(\Gamma^\bullet[F\sigma \circ p]) \end{array} \quad \begin{array}{ll} \text{Ty}^\bullet \Gamma^\bullet A & := \text{Ty}(F\Gamma \triangleright \Gamma^\bullet \triangleright FA[p]) \\ \text{Tm}^\bullet \Gamma^\bullet A^\bullet t & := \text{Tm}(F\Gamma \triangleright \Gamma^\bullet)(A^\bullet[\text{id}, F t[p]]) \end{array}$$

The core calculus is glued using the same technique as the standard model: composition and substitution are modelled by substitution, context extension by Σ types.

$$\begin{array}{ll} \sigma^\bullet \circ^\bullet \rho^\bullet & := \sigma^\bullet[F\rho \circ p, \rho^\bullet] \\ \text{id}^\bullet & := q \\ \diamond^\bullet & := \top \\ \epsilon^\bullet & := \text{tt} \\ A^\bullet[\sigma^\bullet]^\bullet & := A^\bullet[(F\sigma \circ p, \sigma^\bullet) \uparrow] \end{array} \quad \begin{array}{ll} t^\bullet[\sigma^\bullet]^\bullet & := t^\bullet[F\sigma \circ p, \sigma^\bullet] \\ \Gamma^\bullet \triangleright^\bullet A^\bullet & := \Sigma(\Gamma^\bullet[Fp])(A^\bullet[Fp \circ p, q, Fq[p]]) \\ (\sigma^\bullet, t^\bullet) & := (\sigma^\bullet, t^\bullet) \\ p^\bullet & := \pi_1 q \\ q^\bullet & := \pi_2 q \end{array}$$

The type formers $K, \top, \Sigma, \text{Eq}, \Pi, U$ and Bool in the gluing displayed model are defined as follows. K^\bullet simply returns its argument context, $\top^\bullet, \Sigma^\bullet, \text{Eq}^\bullet$ are pointwise. Π^\bullet expresses that if the predicate holds for an input, it holds for the output. U^\bullet gives predicate space, Bool^\bullet for a b in $F\text{Bool}$ says that there is a boolean which is equal to b when mapped using F_{Bool}^{-1} . The definitions are straightforward, but technical: adjustments using F_\triangleright^{-1} and the comparison maps (e.g. F_Σ, F_Π) have to make things match. The definitions below satisfy all the equations of the displayed model.

$$\begin{array}{ll} K^\bullet \Theta^\bullet & := \Theta^\bullet[F_K][F\epsilon \circ p^2, q] \\ \top^\bullet & := \top \\ \text{tt}^\bullet & := \text{tt} \\ \Sigma^\bullet A^\bullet B^\bullet [\gamma, \gamma^\bullet, F_\Sigma^{-1}[\gamma, (a, b)]] & := \Sigma(A^\bullet[\gamma, \gamma^\bullet, a])(B^\bullet[F_\triangleright^{-1} \circ (\gamma, a) \circ p, (\gamma^\bullet[p], q), b[p]]) \\ (u^\bullet, v^\bullet) & := (u^\bullet, v^\bullet) \\ \pi_1^\bullet t^\bullet & := \pi_1 t^\bullet \\ \pi_2^\bullet t^\bullet & := \pi_2 t^\bullet \\ \text{Eq}_A^\bullet u^\bullet v^\bullet & := \text{Eq}_{A^\bullet[p, Fu[p^2]]}(u^\bullet[p])(v^\bullet[p]) \\ \text{refl}^\bullet & := \text{refl} \\ \Pi^\bullet A^\bullet B^\bullet [\gamma, \gamma^\bullet, t] & := \Pi(FA[\gamma])(\Pi(A^\bullet[(\gamma, \gamma^\bullet) \uparrow])) \\ & \quad (B^\bullet[F_\triangleright^{-1} \circ (\gamma \uparrow) \circ p, (\gamma^\bullet[p^2], q), \text{app}(F_\Pi[\gamma, t])[p]])) \\ \text{lam}^\bullet t^\bullet & := \text{lam}, (\text{lam}(t^\bullet[F_\triangleright^{-1} \circ (p^3, 1), (2, 0)])) \\ \text{app}^\bullet t^\bullet & := \text{app}, (\text{app } t)[Fp \circ p, \pi_1 q, Fq[p], \pi_2 q] \\ U^\bullet[\gamma, \gamma^\bullet, a] & := \text{El}(F_U[\gamma, a]) \Rightarrow U \\ \text{El}^\bullet a^\bullet & := \text{El}(\text{app } a^\bullet) \\ c^\bullet A^\bullet & := \text{lam}(c A^\bullet) \\ \text{Bool}^\bullet[\gamma, \gamma^\bullet, b] & := \Sigma \text{Bool}(\text{Eq}_{F\text{Bool}[\gamma][p]}(F_{\text{Bool}}^{-1}[\gamma \uparrow])(b[p])) \\ \text{true}^\bullet & := (\text{true}, \text{refl}) \\ \text{false}^\bullet & := (\text{false}, \text{refl}) \\ \text{ite}^\bullet t^\bullet u^\bullet v^\bullet & := \text{ite}(\pi_1 t^\bullet) u^\bullet v^\bullet \end{array}$$

The displayed versions of the \forall , k , R and S operators are defined as follows.

$$\forall^\bullet \Gamma^\bullet := \forall \Gamma^\bullet \quad \forall^\bullet \sigma^\bullet := \forall \sigma^\bullet \quad \forall^\bullet A^\bullet := \forall A^\bullet \quad \forall^\bullet t^\bullet := \forall t^\bullet$$

$$k^\bullet \Gamma^\bullet := q[k_{F\Gamma \triangleright \Gamma^\bullet}] \quad R^\bullet \Gamma^\bullet := q[R_{F\Gamma \triangleright \Gamma^\bullet}] \quad S^\bullet \Gamma^\bullet := q[S_{F\Gamma \triangleright \Gamma^\bullet}]$$

$\forall^\bullet \Gamma^\bullet$ has to be in $\text{Ty}(F(\forall \Gamma))$ and $\forall \Gamma^\bullet$ is in $\text{Ty}(\forall(F\Gamma))$, but these are equal because F respects \forall strictly. The situation is similar for the other operators.

For $\text{mk}\forall\Pi^\bullet \sigma^\bullet t_k^\bullet t^\bullet$, we need a term in

$$\text{Tm}(F \Delta \triangleright \Delta^\bullet)(\forall(\Pi^\bullet A^\bullet B^\bullet)[F \sigma \circ p, \sigma^\bullet, F(\text{mk}\forall\Pi \sigma t_k t)[p]]) = \\ (\text{introduce abbreviation } \rho)$$

$$\text{Tm}(F \Delta \triangleright \Delta^\bullet)(\forall(\Pi^\bullet A^\bullet B^\bullet)[\rho]) =$$

$$\text{Tm}(F \Delta \triangleright \Delta^\bullet)\left(\forall\left(\Pi(F A[p^2]) (\Pi(A^\bullet[p^2, q]) (B^\bullet[F_\triangleright^{-1} \circ (p^4, 1), (3, 0), F_\Pi[p^4, 2] \$ 1]))\right)[\rho]\right) \cong \\ (\text{curry-uncurry})$$

$$\text{Tm}(F \Delta \triangleright \Delta^\bullet)\left(\forall\left(\Pi(\Sigma(F A[p^2])) (A^\bullet[p^2, q]) (B^\bullet[F_\triangleright^{-1} \circ (p^3, \pi_1 0), (2, \pi_2 0), F_\Pi[p^3, 1] \$ \pi_1 0])\right)[\rho]\right).$$

We can directly apply $\text{mk}\forall\Pi$ to build an element of this latter type reusing the uncurried versions of t_k^\bullet and t^\bullet , respectively.

$$w := \text{mk}\forall\Pi \rho \left(\text{lam}(\text{app}(\text{app} t_k^\bullet)[p, \pi_1 q, \pi_2 q]) \left(\text{lam}(\text{app}(\text{app} t^\bullet)[p, \pi_1 q, \pi_2 q]) \right) \right)$$

Now we curry the resulting term under \forall to obtain the definition of $\text{mk}\forall\Pi^\bullet$:

$$\text{mk}\forall\Pi^\bullet \sigma^\bullet t_k^\bullet t^\bullet := \forall(\text{lam}(\text{lam}(2 \$ (1, 0))))[\rho, w]$$

For $\text{unspan}^\bullet A_k^\bullet A^\bullet t_k^\bullet$, we need a term in

$$\text{Tm}(F \Gamma \triangleright \Gamma^\bullet)(\forall^\bullet U^\bullet [\epsilon^\bullet]^\bullet [\text{id}, F(\text{unspan } A_k A t_k)[p]]) =$$

$$\text{Tm}(F \Gamma \triangleright \Gamma^\bullet)\left(\forall(F(\text{El } q)[F_\triangleright^{-1}] \Rightarrow U)[F \epsilon \circ p, F(\text{unspan } A_k A t_k)[p]]\right) = \\ (\text{introduce abbreviation } \rho)$$

$$\text{Tm}(F \Gamma \triangleright \Gamma^\bullet)\left(\forall(F(\text{El } q)[F_\triangleright^{-1}] \Rightarrow U)[\rho]\right),$$

which is the span of a predicate. We already have predicates on $F A_k$ and $F A$ with a map from witnesses of the second one to the first one that lies over $F t_k$:

$$\begin{aligned} A_k^\bullet &: \text{Ty}(F \Gamma \triangleright \Gamma^\bullet \triangleright F A_k[p]) \\ A^\bullet &: \text{Ty}(F \Gamma \triangleright \Gamma^\bullet \triangleright F A[p]) \\ t_k^\bullet &: \text{Tm}(F(\Gamma \triangleright A) \triangleright \Gamma^\bullet \triangleright A^\bullet)(A_k^\bullet[F p \circ p, \pi_1 q, F t_k[p]]) \end{aligned}$$

The way to make terms in \forall of a function space is by $\text{mk}\forall\Pi$. The function (the predicate) at the base is essentially A_k^\bullet , the predicate at the apex is given by $\text{unspan}(A_k^\bullet[\dots]) A^\bullet(t_k^\bullet[\dots])$ where A_k^\bullet and t_k^\bullet have to be substituted to plug in their dependencies properly.

$$\text{unspan}^\bullet A_k^\bullet A^\bullet t_k^\bullet :=$$

$$\text{mk}\forall\Pi \rho \left(\text{lam}(\text{c } A_k^\bullet) \left(\text{lam} \left(\text{unspan}(A_k^\bullet[p, F t_k[F_\triangleright^{-1}[p^2, q]]]) A^\bullet(t_k^\bullet[F_\triangleright^{-1} \circ (p^3, 1), (2, 0)]) \right) \right) \right)$$

The equations for the global syntax follow from the preservation properties of the weak morphism. Here is a typical example:

$$\begin{aligned}
 & \forall^\bullet k_{\Gamma^\bullet} & = \\
 & q[\forall k_{F\Gamma^\bullet\Gamma^\bullet}] & = \\
 & & \text{(one of the five equations)} \\
 & q[k_{\forall(F\Gamma)\triangleright\forall\Gamma^\bullet}\circ S_{F\Gamma\triangleright\Gamma^\bullet}] & = \\
 & q[k_{\forall(F\Gamma)\triangleright\forall\Gamma^\bullet}][S_{F\Gamma\triangleright\Gamma^\bullet}] & = \\
 & & \text{(} F \text{ commutes with } \forall \text{ on contexts)} \\
 & q[k_{F(\forall\Gamma)\triangleright\forall\Gamma^\bullet}][S_{F\Gamma\triangleright\Gamma^\bullet}] & = \\
 & q[k_{F(\forall\Gamma)\triangleright\forall\Gamma^\bullet}][p\circ S_{F\Gamma\triangleright\Gamma^\bullet}, q[S_{F\Gamma\triangleright\Gamma^\bullet}]] & = \\
 & q[k_{F(\forall\Gamma)\triangleright\forall\Gamma^\bullet}][\forall(\forall p)\circ S_{F\Gamma\triangleright\Gamma^\bullet}, q[S_{F\Gamma\triangleright\Gamma^\bullet}]] & = \\
 & & \text{(naturality of } S \text{)} \\
 & q[k_{F(\forall\Gamma)\triangleright\forall\Gamma^\bullet}][S_{F\Gamma}\circ\forall(\forall p), q[S_{F\Gamma\triangleright\Gamma^\bullet}]] & = \\
 & q[k_{F(\forall\Gamma)\triangleright\forall\Gamma^\bullet}][S_{F\Gamma}\circ p, q[S_{F\Gamma\triangleright\Gamma^\bullet}]] & = \\
 & & \text{(} F \text{ commutes with } S \text{)} \\
 & q[k_{F(\forall\Gamma)\triangleright\forall\Gamma^\bullet}][F S_\Gamma\circ p, q[S_{F\Gamma\triangleright\Gamma^\bullet}]] & = \\
 & k_{\forall^\bullet\Gamma^\bullet}^\bullet\circ^\bullet S_\Gamma^\bullet &
 \end{aligned}$$

This completes the construction of the gluing displayed model. \square

5.3 The Global Section Functor

We define a way to interpret morphisms in \square in our syntax (see Definition 3.3). The substitution $\lceil f \rceil_\Gamma$ for an $f : \square(J, I)$ is defined mutually with three equations by induction on f :

$$\begin{aligned}
 \lceil f \rceil_\Gamma & : \text{Sub}(\forall^I \Gamma)(\forall^J \Gamma) \\
 \lceil f \rceil_{\forall^n \Gamma} \circ \forall^{I+n} k_\Gamma & = \forall^{J+n} k_\Gamma \circ \lceil f \rceil_{\forall^{1+n} \Gamma} \\
 \forall^{J+n} R_\Gamma \circ \lceil f \rceil_{\forall^n \Gamma} & = \lceil f \rceil_{\forall^{1+n} \Gamma} \circ \forall^{I+n} R_\Gamma \\
 \forall^{J+n} S_\Gamma \circ \lceil f \rceil_{\forall^{2+n} \Gamma} & = \lceil f \rceil_{\forall^{2+n} \Gamma} \circ \forall^{I+n} S_\Gamma
 \end{aligned}$$

$\forall^I \Gamma$ denotes the I times iteration of \forall on Γ . The different cases³ of $\lceil - \rceil_\Gamma$:

$$\begin{aligned}
 \lceil f \circ g \rceil_\Gamma & := \lceil g \rceil_\Gamma \circ \lceil f \rceil_\Gamma & \lceil \text{suc } f \rceil_\Gamma & := \lceil f \rceil_{\forall \Gamma} & \lceil R_I \rceil_\Gamma & := \forall^I R_\Gamma \\
 \lceil \text{id} \rceil_\Gamma & := \text{id} & \lceil k_I \rceil_\Gamma & := \forall^I k_\Gamma & \lceil S_I \rceil_\Gamma & := \forall^I S_\Gamma
 \end{aligned}$$

The fact that $\lceil - \rceil_\Gamma$ preserves the equations for the morphisms in \square is direct except for the three naturality equations $k_I \circ f = \text{suc } f \circ k_J$, $R_I \circ \text{suc } f = f \circ R_J$ and $S_I \circ \text{suc } (\text{suc } f) = \text{suc } (\text{suc } f) \circ S_J$ which follow from the $n = 0$ cases of the above three equations.

We also prove by induction on f that for any σ , $\lceil f \rceil_\sigma : \lceil f \rceil_\Gamma \circ \forall^I \sigma = \forall^J \sigma \circ \lceil f \rceil_\Delta$.

Now we construct the weak morphism on which we will glue for canonicity. This can be seen as a cubical nerve functor.

PROBLEM 5.3 (GLOBAL SECTION FUNCTOR). *We construct a weak morphism G from Syn to $\text{PSh}(\square)$.*

³An alternative definition of $\lceil - \rceil_\Gamma$ uses $\lceil \text{suc } f \rceil_\Gamma = \forall \lceil f \rceil_\Gamma$, $\lceil k_I \rceil_\Gamma = k_{\forall I \Gamma}$, and so on. This version generates a global section functor which only satisfies $G(\forall \Gamma) \cong \forall(G\Gamma)$ up to isomorphism. An isomorphism is enough to build a gluing model, however it is much more tedious than our current, stricter approach.

CONSTRUCTION.

$$G : \text{Con}_{\text{Syn}} \rightarrow \text{Con}_{\text{PSH}(\square)}$$

$$G \Gamma I := \text{Sub}_{\text{Syn}} \diamond (\vee^I \Gamma)$$

$$\gamma^I[f]_{G\Gamma} := \lceil f \rceil_\Gamma \circ \gamma^I$$

$$G : \text{Sub}_{\text{Syn}} \Delta \Gamma \rightarrow \text{Sub}_{\text{PSH}(\square)} (G \Delta) (G \Gamma)$$

$$G \sigma \delta^I := \vee^I \sigma \circ \delta^I$$

The functor laws for $G \Gamma$ follow from the definition of $\lceil \cdot \rceil_\Gamma$ and from the categorical laws in Syn . Naturality of $G \sigma$ is a consequence of $\lceil f \rceil_\sigma$. On types and terms:

$$G : \text{Ty}_{\text{Syn}} \Gamma \rightarrow \text{Ty}_{\text{PSH}(\square)} (G \Gamma)$$

$$G A I \gamma^I := \text{Tm}_{\text{Syn}} \diamond (\vee^I A[\gamma^I])$$

$$a^I[f]_{GA} := q[\lceil f \rceil_{\Gamma \triangleright A}][\gamma^I, a^I]$$

$$G : \text{Tm}_{\text{Syn}} \Gamma A \rightarrow \text{Tm}_{\text{PSH}(\square)} (G \Gamma) (G A)$$

$$G t \gamma^I := \vee^I t[\gamma^I]$$

The functor laws for $G A$ follow directly and naturality for $G t$ is proven as follows.

$$G t \gamma^I[f]_{GA} = q[\lceil f \rceil_{\Gamma \triangleright A}][\gamma^I, \vee^I t[\gamma^I]] = q[\lceil f \rceil_{\Gamma \triangleright A}][\vee^I(\text{id}, t)][\gamma^I] \stackrel{\lceil f \rceil_\sigma}{=} q[\vee^J(\text{id}, t)][\lceil f \rceil_\Gamma][\gamma^I] = q[\text{id}, \vee^J t][\lceil f \rceil_\Gamma][\gamma^I] = \vee^J t[\lceil f \rceil_\Gamma \circ \gamma^I] = G t(\gamma^I[f]_{G\Gamma})$$

It is easy to see that G is a functor, it is more interesting that it preserves substitution of types:

$$G(A[\sigma]) I \delta^I = \text{Tm}_{\text{Syn}} \diamond (\vee^I(A[\sigma])[\delta^I]) = \text{Tm}_{\text{Syn}} \diamond (\vee^I A[\vee^I \sigma][\delta^I]) = G A I (\vee^I \sigma \circ \delta^I) =$$

$$G A I (G \sigma \delta^I) = G A [G \sigma] I \delta^I$$

$$a^I[f]_{G(A[\sigma])} = q[\lceil f \rceil_{\Delta \triangleright A[\sigma]}][\delta^I, a^I] = q[\vee^J(\sigma \uparrow)][\lceil f \rceil_{\Delta \triangleright A[\sigma]}][\delta^I, a^I] \stackrel{\lceil f \rceil_\sigma}{=} q[\lceil f \rceil_{\Gamma \triangleright A}][\vee^I(\sigma \uparrow)][\delta^I, a^I] = q[\lceil f \rceil_{\Gamma \triangleright A}][\vee^I \sigma \circ \delta^I, a^I] = a^I[f]_{GA[G\sigma]}$$

A similar proof shows that G preserves term substitution. We define preservation of \diamond and \dashv as follows. In the latter case we replace a metatheoretic pairing operation with a syntactic extended substitution former.

$$G^{-1}_\diamond : \text{Sub}_{\text{PSH}(\square)} \diamond (G \diamond)$$

$$G^{-1}_\diamond := \epsilon_{\text{Syn}}$$

$$G^{-1}_\triangleright : \text{Sub}_{\text{PSH}(\square)} (G \Gamma \triangleright G A) (G (\Gamma \triangleright A))$$

$$G^{-1}_\triangleright (\gamma^I, a^I) := (\gamma^I, \triangleright_{\text{Syn}} a^I)$$

They obviously satisfy the necessary equations using the universal properties of \diamond and \dashv .

We verify that G preserves \vee and commutes with k, R, S . G commutes with \forall on contexts

$$G(\forall \Gamma) I = \text{Sub}_{\text{Syn}} \diamond (\vee^I(\forall \Gamma)) = \text{Sub}_{\text{Syn}} \diamond (\vee^{1+I} \Gamma) = G \Gamma (1 + I) = \forall(G \Gamma) I$$

$$\gamma^{1+I}[f]_{G(\forall \Gamma)} = \lceil f \rceil_\Gamma \circ \gamma^{1+I} = \lceil \text{suc } f \rceil_\Gamma \circ \gamma^{1+I} = \gamma^{1+I}[\text{suc } f]_{G\Gamma} = \gamma^{1+I}[f]_{\forall(G\Gamma)},$$

substitutions

$$G(\forall \sigma) \delta^{1+I} = \vee^I(\forall \sigma) \circ \delta^{1+I} = \vee^{1+I} \sigma \circ \delta^{1+I} = G \sigma \delta^{1+I} = \forall(G \sigma) \delta^{1+I},$$

types

$$G(\forall A) I \gamma^{1+I} = \text{Tm} \diamond (\vee^{1+I} A[\gamma^{1+I}]) = G A (1 + I) \gamma^{1+I} = \forall(G A) I \gamma^{1+I}$$

$$a^{1+I}[f]_{G(\forall A)} = q[\lceil f \rceil_{\forall \Gamma \triangleright \forall A}][\gamma^{1+I}, a^{1+I}] = q[\lceil \text{suc } f \rceil_{\Gamma \triangleright A}][\gamma^{1+I}, a^{1+I}] = a^{1+I}[f]_{\forall(GA)},$$

and terms

$$G(\forall t) \gamma^{1+I} = \vee^I(\forall t)[\gamma^{1+I}] = \vee^{1+I} t[\gamma^{1+I}] = G t \gamma^{1+I} = \forall(G t) \gamma^{1+I}.$$

Preservation of k is given by

$$G k_\Gamma \gamma^{1+I} = \vee^I k_\Gamma \circ \gamma^{1+I} = \lceil k_I \rceil_\Gamma \circ \gamma^{1+I} = \gamma^{1+I}[k_I]_{G\Gamma} = k_{G\Gamma} \gamma^{1+I},$$

preservation of R and S are shown analogously. \square

5.4 Reaping the Fruits

THEOREM 5.4 (CANONICITY). *In the local or global syntax, for a $t : \text{Tm} \diamond \text{Bool}$, we have $t = \text{true}$ or $t = \text{false}$.*

PROOF. By gluing along the global section functor, we obtain a displayed model over the syntax of the global theory, and given a $t : \text{Tm}_{\text{Syn}} \diamond \text{Bool}$, by induction of Syn , we get a

$$\text{Tm}_{\text{PSH}(\square)}(G \diamond \triangleright T)(\Sigma \text{Bool}(\text{Eq}(\text{ite } q(G \text{ true}[p])(G \text{ false}[p]))(G t[p])))$$

and providing the $\text{id}_\diamond : G \diamond \diamond$ and the element of the metatheoretic unit set as input we obtain

$$(b : \mathbb{2}) \times \text{if } b \text{ then true else false} = t.$$

Given a term in the local syntax $t : \text{Tm}_{\text{Syn}} \diamond \text{Bool}$, we have $\alpha t : \text{Tm}_{\text{Syn}}(\alpha \diamond)(\alpha \text{ Bool})$ where α is the map from the global syntax to the local syntax defined in Section 4.4. As α does not affect the core syntax this is $\alpha t : \text{Tm}_{\text{Syn}} \diamond \text{Bool}$ and we learn from canonicity say that $\alpha t = \text{true}$. Now applying β to such an equation we obtain $t = \beta(\alpha t) = \beta \text{ true} = \text{true}$ as β also does not affect that core calculus. \square

6 FUTURE WORK

We presented a type theory with internal parametricity, a presheaf model and a canonicity proof. It can be seen as a baby version of higher observational type theory (HOTT). To obtain HOTT, we plan to add the following additional features to our theory:

- a bridge type which can be seen as an indexed version of \vee ,
- Reedy fibrancy, which replaces spans by relations,
- a strictification construction which turns the isomorphism for Π types into a definitional equality (in case of bridge, we also need the same for Σ),
- Kan fibrancy, which adds transport and turns the bridge type into a proper identity type. This would also change the correspondence between $\forall U$ and spans into $\forall U$ and equivalences.

We would also like to include general (higher) inductive and coinductive types. Concerning the metatheory, we plan to use internal language techniques [Bocquet et al. 2023; Sterling 2022] to obtain a higher level canonicity proof, and extend it to normalisation.

Several of our constructions in this paper follow a generic pattern: most of the global theory and gluing should be derivable from the 2-category in Figure 1 similarly to the way it is done for multimodal type theory [Gratzer 2023].

ACKNOWLEDGMENTS

We thank the anonymous reviewers for their comments and suggestions. We thank Rafaël Bocquet, Hugo Herbelin, András Kovács and Christian Sattler for discussions related to the topics of this paper.

The first and third authors were supported by project no. TKP2021-NVA-29 which has been implemented with the support provided by the Ministry of Culture and Innovation of Hungary from the National Research, Development and Innovation Fund, financed under the TKP2021-NVA funding scheme.

This material is based upon work supported by the Air Force Office of Scientific Research under award number FA9550-21-1-0009.

REFERENCES

- Thorsten Altenkirch, Paolo Capriotti, and Nicolai Kraus. 2016. Extending Homotopy Type Theory with Strict Equality. In *25th EACSL Annual Conference on Computer Science Logic, CSL 2016, August 29 - September 1, 2016, Marseille, France*

- (LIPICS, Vol. 62), Jean-Marc Talbot and Laurent Regnier (Eds.). Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 21:1–21:17. <https://doi.org/10.4230/LIPIcs.CSL.2016.21>
- Thorsten Altenkirch and Ambrus Kaposi. 2015. Towards a Cubical Type Theory without an Interval. In *21st International Conference on Types for Proofs and Programs, TYPES 2015, May 18–21, 2015, Tallinn, Estonia (LIPICS, Vol. 69)*, Tarmo Uustalu (Ed.). Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 3:1–3:27. <https://doi.org/10.4230/LIPIcs.TYPES.2015.3>
- Thorsten Altenkirch and Ambrus Kaposi. 2016. Type theory in type theory using quotient inductive types. In *Proceedings of the 43rd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2016, St. Petersburg, FL, USA, January 20 – 22, 2016*, Rastislav Bodík and Rupak Majumdar (Eds.). ACM, 18–29. <https://doi.org/10.1145/2837614.2837638>
- Carlo Angiuli, Guillaume Brunerie, Thierry Coquand, Robert Harper, Kuen-Bang Hou (Favonia), and Daniel R. Licata. 2021. Syntax and models of Cartesian cubical type theory. *Math. Struct. Comput. Sci.* 31, 4 (2021), 424–468. <https://doi.org/10.1017/S0960129521000347>
- Danil Annenkov, Paolo Capriotti, and Nicolai Kraus. 2017. Two-Level Type Theory and Applications. *CoRR* abs/1705.03307 (2017). arXiv:1705.03307 <http://arxiv.org/abs/1705.03307>
- Jean-Philippe Bernardy, Thierry Coquand, and Guilhem Moulin. 2015. A Presheaf Model of Parametric Type Theory. In *The 31st Conference on the Mathematical Foundations of Programming Semantics, MFPS 2015, Nijmegen, The Netherlands, June 22–25, 2015 (Electronic Notes in Theoretical Computer Science, Vol. 319)*, Dan R. Ghica (Ed.). Elsevier, 67–82. <https://doi.org/10.1016/j.entcs.2015.12.006>
- Jean-Philippe Bernardy, Patrik Jansson, and Ross Paterson. 2010. Parametricity and dependent types. In *Proceeding of the 15th ACM SIGPLAN international conference on Functional programming, ICFP 2010, Baltimore, Maryland, USA, September 27–29, 2010*, Paul Hudak and Stephanie Weirich (Eds.). ACM, 345–356. <https://doi.org/10.1145/1863543.1863592>
- Jean-Philippe Bernardy and Guilhem Moulin. 2012. A Computational Interpretation of Parametricity. In *Proceedings of the 27th Annual IEEE Symposium on Logic in Computer Science, LICS 2012, Dubrovnik, Croatia, June 25–28, 2012*. IEEE Computer Society, 135–144. <https://doi.org/10.1109/LICS.2012.25>
- Jean-Philippe Bernardy and Guilhem Moulin. 2013. Type-theory in color. In *ACM SIGPLAN International Conference on Functional Programming, ICFP’13, Boston, MA, USA - September 25 - 27, 2013*, Greg Morrisett and Tarmo Uustalu (Eds.). ACM, 61–72. <https://doi.org/10.1145/2500365.2500577>
- Marc Bezem, Thierry Coquand, and Simon Huber. 2013. A Model of Type Theory in Cubical Sets. In *19th International Conference on Types for Proofs and Programs, TYPES 2013, April 22–26, 2013, Toulouse, France (LIPICS, Vol. 26)*, Ralph Matthes and Aleksy Schubert (Eds.). Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 107–128. <https://doi.org/10.4230/LIPIcs.TYPES.2013.107>
- Rafaël Bocquet, Ambrus Kaposi, and Christian Sattler. 2023. For the Metatheory of Type Theory, Internal Sconing Is Enough. In *8th International Conference on Formal Structures for Computation and Deduction, FSCD 2023, July 3–6, 2023, Rome, Italy (LIPIcs, Vol. 260)*, Marco Gaboardi and Femke van Raamsdonk (Eds.). Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 18:1–18:23. <https://doi.org/10.4230/LIPIcs.FSCD.2023.18>
- Auke Bart Booij, Martín Hötzl Escardó, Peter LeFanu Lumsdaine, and Michael Shulman. 2016. Parametricity, Automorphisms of the Universe, and Excluded Middle. In *22nd International Conference on Types for Proofs and Programs, TYPES 2016, May 23–26, 2016, Novi Sad, Serbia (LIPIcs, Vol. 97)*, Silvia Ghilezan, Herman Geuvers, and Jelena Ivetić (Eds.). Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 7:1–7:14. <https://doi.org/10.4230/LIPIcs.TYPES.2016.7>
- Simon Boulier, Pierre-Marie Pédrot, and Nicolas Tabareau. 2017. The next 700 syntactical models of type theory. In *Proceedings of the 6th ACM SIGPLAN Conference on Certified Programs and Proofs, CPP 2017, Paris, France, January 16–17, 2017*, Yves Bertot and Viktor Vafeiadis (Eds.). ACM, 182–194. <https://doi.org/10.1145/3018610.3018620>
- Ulrik Buchholtz and Edward Morehouse. 2017. Varieties of Cubical Sets. In *Relational and Algebraic Methods in Computer Science - 16th International Conference, RAMiCS 2017, Lyon, France, May 15–18, 2017, Proceedings (Lecture Notes in Computer Science, Vol. 10226)*, Peter Höfner, Damien Pous, and Georg Struth (Eds.), 77–92. https://doi.org/10.1007/978-3-319-57418-9_5
- Simon Castellan, Pierre Clairambault, and Peter Dybjer. 2019. Categories with Families: Untyped, Simply Typed, and Dependently Typed. *CoRR* abs/1904.00827 (2019). arXiv:1904.00827 <http://arxiv.org/abs/1904.00827>
- Evan Cavallo. 2021. *Higher Inductive Types and Internal Parametricity for Cubical Type Theory*. Ph. D. Dissertation. Carnegie Mellon University, USA. <https://doi.org/10.1184/r1/14555691>
- Evan Cavallo and Robert Harper. 2021. Internal Parametricity for Cubical Type Theory. *Log. Methods Comput. Sci.* 17, 4 (2021). [https://doi.org/10.46298/lmcs-17\(4:5\)2021](https://doi.org/10.46298/lmcs-17(4:5)2021)
- Cyril Cohen, Thierry Coquand, Simon Huber, and Anders Mörtberg. 2015. Cubical Type Theory: A Constructive Interpretation of the Univalence Axiom. In *21st International Conference on Types for Proofs and Programs, TYPES 2015, May 18–21, 2015, Tallinn, Estonia (LIPIcs, Vol. 69)*, Tarmo Uustalu (Ed.). Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 5:1–5:34. <https://doi.org/10.4230/LIPIcs.TYPES.2015.5>
- Thierry Coquand. 2018. Presheaf model of type theory. (2018). <https://www.cse.chalmers.se/~coquand/presheaf.pdf>.

- Daniel Gratzer. 2023. Normalization for multimodal type theory. *CoRR* abs/2301.11842 (2023). <https://doi.org/10.48550/arXiv.2301.11842> arXiv:2301.11842
- Daniel Gratzer, G. A. Kavvos, Andreas Nuyts, and Lars Birkedal. 2021. Multimodal Dependent Type Theory. *Log. Methods Comput. Sci.* 17, 3 (2021). [https://doi.org/10.46298/lmcs-17\(3:11\)2021](https://doi.org/10.46298/lmcs-17(3:11)2021)
- Martin Hofmann. 1997. Syntax and Semantics of Dependent Types. In *Semantics and Logics of Computation*. Cambridge University Press, 79–130.
- Ambrus Kaposi, Simon Huber, and Christian Sattler. 2019a. Gluing for Type Theory. In *4th International Conference on Formal Structures for Computation and Deduction, FSCD 2019, June 24-30, 2019, Dortmund, Germany (LIPIcs, Vol. 131)*, Herman Geuvers (Ed.). Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 25:1–25:19. <https://doi.org/10.4230/LIPIcs.FSCD.2019.25>
- Ambrus Kaposi, András Kovács, and Thorsten Altenkirch. 2019b. Constructing quotient inductive-inductive types. *Proc. ACM Program. Lang.* 3, POPL (2019), 2:1–2:24. <https://doi.org/10.1145/3290315>
- András Kovács and Ambrus Kaposi. 2020. Large and Infinitary Quotient Inductive-Inductive Types. In *LICS ’20: 35th Annual ACM/IEEE Symposium on Logic in Computer Science, Saarbrücken, Germany, July 8–11, 2020*, Holger Hermanns, Lijun Zhang, Naoki Kobayashi, and Dale Miller (Eds.). ACM, 648–661. <https://doi.org/10.1145/3373718.3394770>
- Nicolai Kraus and Christian Sattler. 2017. Space-Valued Diagrams, Type-Theoretically (Extended Abstract). *CoRR* abs/1704.04543 (2017), arXiv:1704.04543 <http://arxiv.org/abs/1704.04543>
- Antoine Van Muylster, Andreas Nuyts, and Dominique Devriese. 2024. Internal and Observational Parametricity for Cubical Agda. In *To appear in: Proceedings of the 51st Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2024, London, UK, January 17 – 19, 2024*. ACM.
- Andreas Nuyts and Dominique Devriese. 2018. 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*, Anuj Dawar and Erich Grädel (Eds.). ACM, 779–788. <https://doi.org/10.1145/3209108.3209119>
- Andreas Nuyts, Andrea Vezzosi, and Dominique Devriese. 2017. Parametric quantifiers for dependent type theory. *Proc. ACM Program. Lang.* 1, ICFP (2017), 32:1–32:29. <https://doi.org/10.1145/3110276>
- John C. Reynolds. 1983. Types, Abstraction and Parametric Polymorphism. In *Information Processing 83, Proceedings of the IFIP 9th World Computer Congress, Paris, France, September 19–23, 1983*, R. E. A. Mason (Ed.). North-Holland/IFIP, 513–523.
- Jonathan Sterling. 2022. *First Steps in Synthetic Tait Computability: The Objective Metatheory of Cubical Type Theory*. Ph.D. Dissertation. Carnegie Mellon University, USA. <https://doi.org/10.1184/r1/19632681.v1>
- Taichi Uemura. 2019. A General Framework for the Semantics of Type Theory. *CoRR* abs/1904.04097 (2019). arXiv:1904.04097 <http://arxiv.org/abs/1904.04097>
- Andrea Vezzosi, Anders Mörtberg, and Andreas Abel. 2021. Cubical Agda: A dependently typed programming language with univalence and higher inductive types. *J. Funct. Program.* 31 (2021), e8. <https://doi.org/10.1017/S0956796821000034>
- Philip Wadler. 1990. Recursive types for free! (1990). <https://homepages.inf.ed.ac.uk/wadler/papers/free-rectypes/free-rectypes.txt>.

Received 2023-07-11; accepted 2023-11-07