Large and Infinitary Quotient Inductive-Inductive Types

Anonymous Author(s)

Abstract

Quotient inductive-inductive types (QIITs) are generalized inductive types which allow sorts to be indexed over previously declared sorts, and allow usage of equality constructors. QIITs are especially useful for algebraic descriptions of type theories and constructive definitions of real, ordinal and surreal numbers. We develop new metatheory for large QIITs, large elimination, recursive equations and infinitary constructors. As in prior work, we describe QIITs using a type theory where each context represents a QIIT signature. However, in our case the theory of signatures can also describe its own signature, modulo universe sizes. We bootstrap the model theory of signatures using self-description and a Church-coded notion of signature, without using complicated raw syntax or assuming an existing internal QIIT of signatures. We give semantics to described QIITs by modeling each signature as a finitely complete CwF (category with families) of algebras. Compared to the case of finitary QIITs, we additionally need to show invariance under algebra isomorphisms in the semantics. We do this by modeling signature types as isofibrations. Finally, we show by a term model construction that every QIIT is constructible from the syntax of the theory of signatures.

Keywords inductive types, infinitary inductive types, quotient inductive-inductive types, categories with families

1 Introduction

The aim of this work is to provide theoretical underpinning to a general notion of inductive types, called quotient inductive-inductive types (QIITs). QIITs are of interest because there are many commonly used mathematical structures, which can be conveniently described as QIITs in type theory, but cannot be defined as less general inductive types, or doing so incurs large encoding overhead.

Categories are a prime example for a structure which is described by a quotient inductive-inductive signature. Signatures for QIITs allow having multiple sorts, with later ones indexed over previous ones, and equations as well. We need both features in order to write down the signature of categories.

The benefit of having a QIIT signature is getting a model theory "for free", from the metatheory of QIITs. This model theory includes a category of algebras which has an initial object and also some additional structure. For the signature of categories, we get the empty category as the initial object, but it is common to consider categories with more structure, which have more interesting initial models.

Algebraic notions of models of type theories are examples for this. Here, initial models represent syntax, and initiality corresponds to induction on syntax. Several variants have been used, from contextual categories [12] and comprehension categories [21] to categories with families [18]. Additionally, real, surreal [27] and ordinal numbers [26] can be defined as *infinitary* QIITs, which are represented as possibly infinitely branching inductive trees.

1.1 Contributions

The current paper extends the syntax and semantics of QIITs as previously described in the literature [2, 17, 24]. We add the following features:

1. **Large constructors, large elimination** and algebras at different universe levels. This fills in an important formal gap; large models are routinely used in the metatheory of type theories, but they have not been presented explicitly in previous QIIT literature.
2. **Infinitary constructors**. Of special note here is that the theory of QIIT signatures is itself large and infinitary, thus it can "eat itself", i.e. include its own signature and provide its own metatheory. This was not possible previously in [24], where only finitary QIITs were described. In this paper we use self-representation to bootstrap the model theory of signatures, without having to assume any pre-existing internal syntax.
3. **Recursive equations**, i.e. equations appearing as assumptions of constructors. These have occurred previously in syntaxes of cubical type theories, as boundary conditions[**?**].

To provide semantics, we show that for each signature, there is a CwF (category with families) of algebras, extended with Σ-types, extensional identity, and constant families. This additional structure corresponds to a type-theoretic flavor of finite limits, as it was shown in [14] that the category of such CwFs is biequivalent to the category of finitely complete categories.

Compared to the case of finitary QIITs, the addition of infinitary constructors and recursive equations requires a significant change in semantics: instead of strict CwF morphisms, we need to consider weak ones, and instead of modeling types as displayed CwFs, we need to model them as CwF isofibrations. The latter amounts to showing that signature extension respects algebra isomorphisms.

PL'18, January 01–03, 2018, New York, NY, USA
2018.

We also show, by a term model construction, that all QIITs are reducible to the syntax of signatures. This construction also essentially relies on invariance under isomorphisms.

1.2 Outline of the Paper

In Section 2, we describe the metatheory used in the rest of the paper. In Section 3, we introduce the theory of QIIT signatures. In Section 4 we give categorical semantics to signatures. In Section 5 we build model theory for the theory of QIIT signatures. In Section 6 we give a term model construction of QIITs. We discuss related work and conclude in Sections 7-8.

2 Metatheory

The metatheory used in this paper is extensional type theory, extended with a form of cumulativity and an external notion of universe polymorphism. We refer to this theory as cETT. We review the used features and notations in the following.

2.1 Core Extensional Theory

We have Russell-style predicative universes Set_i indexed by natural numbers, dependent functions as $(x : A) \to B$, and dependent pairs as $(x : A) \times B$ with projections proj1 and proj2. We sometimes leave parameters implicit in dependent function types, e.g. write $\mathsf{id} : A \to A$ instead of $\mathsf{id} : (A : \mathsf{Set}_i) \to A \to A$. We also use subscripts as a field projection notation for iterated pairs. For example, for $t : (A : \mathsf{Set}_i) \times (B : \mathsf{Set}_i) \times (f : A \to B)$, we use B_t to denote the projection of the second component. Sometimes we omit the subscript if it is clear from context. When we write "exists" in this paper, we always mean chosen structure given by a Σ-type.

Both for function types and Σ, the output universe level is given as the maximum of the levels of the constituent types, e.g. $(x : A) \to B : \mathsf{Set}_{\max(i,j)}$ when $A : \mathsf{Set}_i$ and $B : \mathsf{Set}_j$.

We write propositional equality as $t = u$, with refl_t for reflexivity. We have equality reflection and uniqueness of identity proofs (UIP). The unit type is $\top : \mathsf{Set}_0$, with inhabitant tt.

2.2 Cumulativity

We use cumulative universes and cumulative subtyping as described in [30]. Concretely, we have a $- \le -$ subtyping relation on types, specified by the following rules:

$$\frac{i \le j}{\Gamma \vdash \mathsf{Set}_i \le \mathsf{Set}_j} \qquad \frac{\Gamma, x : A \vdash B \le B'}{\Gamma \vdash (x : A) \to B \le (x : A) \to B'}$$

$$\frac{\Gamma \vdash A \le A' \qquad \Gamma, x : A \vdash B \le B'}{\Gamma \vdash (x : A) \times B \le (x : A') \times B'} \qquad \frac{}{\Gamma \vdash A \le A}$$

$$\frac{\Gamma \vdash A \le B \qquad \Gamma \vdash B \le C}{\Gamma \vdash A \le C} \qquad \frac{\Gamma \vdash A \le A' \qquad \Gamma \vdash t : A}{\Gamma \vdash t : A'}$$

Additionally, we have an internal type for $- \le -$, analogously to $- = -$ which is an internalization of definitional equality.

$$\frac{\Gamma \vdash A : \mathsf{Set}_i \qquad \Gamma \vdash B : \mathsf{Set}_j}{\Gamma \vdash \mathsf{Subtype}\, A\, B : \mathsf{Set}_{\max(i,j)}}$$

$$\frac{\Gamma \vdash A \le B}{\Gamma \vdash \mathsf{subtype} : \mathsf{Subtype}\, A\, B} \qquad \frac{\Gamma \vdash t : \mathsf{Subtype}\, A\, B}{\Gamma \vdash A \le B}$$

We use cumulativity to reduce bureaucratic overhead when dealing with constructions at different universe levels. The internal Subtype is used in Section 6 to prove cumulativity for general QIIT algebras. For example, consider natural number algebras at level i, given as the Σ-type $NatAlg_i :\equiv (Nat : \mathsf{Set}_i) \times Nat \times (Nat \to Nat)$. It follows from the subtyping rules that $i \le j$ implies $NatAlg_i \le NatAlg_j$. However, cumulativity for arbitrary QIIT algebras does not follow judgmentally; it can only be proven by induction on signatures, hence the need for Subtype.

Internal subtyping is not included in [30], but it can be justified by the set-theoretic model given there.

2.3 Universe Polymorphism

We need to talk about constructions at arbitrary universe levels. For the sake of simplicity, we do not assume a notion of universe polymorphism in cETT, instead we quantify over levels in an unspecified theory outside of cETT. Hence, a universe polymorphic cETT term is understood as a $\mathbb{N}$-indexed family of cETT terms. We reuse the notation of cETT functions for universe polymorphism, e.g. as in the following function:

$$\lambda\, i.\, \mathsf{Set}_i : (i : \mathbb{N}) \to \mathsf{Set}_{i+1}$$

3 QIIT Signatures

Signatures are given as contexts in a certain type theory, the theory of signatures. We shall abbreviate it as ToS. However, ToS turns out to be a large infinitary QIIT itself, and we would like to define ToS and a notion of signature without referring to QIITs, only using features present in cETT. As a first step, we define a notion of model.

Definition 3.1 (Notion of model for the theory of signatures)**.** For levels i and j, $\mathsf{ToS}_{i,j} : \mathsf{Set}_{\max(i+1,\, j+1)}$ is a cETT type whose elements are ToS models (or ToS-algebras). $\mathsf{ToS}_{i,j}$ is an iterated Σ-type, containing all of the following components.

A **category with families** (CwF), where all four underlying sets (of objects, morphisms, types and terms) are in Set_i. Following notation in [24], we denote these respectively as $\mathsf{Con} : \mathsf{Set}_i$, $\mathsf{Sub} : \mathsf{Con} \to \mathsf{Con} \to \mathsf{Set}_i$, $\mathsf{Ty} : \mathsf{Con} \to \mathsf{Set}_i$ and $\mathsf{Tm} : (\Gamma : \mathsf{Con}) \to \mathsf{Ty}\, \Gamma \to \mathsf{Set}_i$. We use id and $- \circ -$ to denote identity and composition for substitution. We denote the empty context as $\bullet : \mathsf{Con}$, and the unique substitution into the empty context as $\epsilon : \mathsf{Sub}\, \Gamma\, \bullet$. Context extension is $- \triangleright - :$

$(\Gamma : \mathsf{Con}) \to \mathsf{Ty}\,\Gamma \to \mathsf{Con}$. Substitution on types and terms is written as $-[-]$. Projections are noted as $\mathsf{p} : \mathsf{Sub}\,(\Gamma \triangleright A)\,\Gamma$ and $\mathsf{q} : \mathsf{Tm}\,(\Gamma \triangleright A)\,(A[\mathsf{p}])$, and substitution extension is $-, - : (\sigma : \mathsf{Sub}\,\Gamma\,\Delta) \to \mathsf{Tm}\,\Gamma\,(A[\sigma]) \to \mathsf{Sub}\,\Gamma\,(\Delta \triangleright A)$.

A **universe** $\mathsf{U} : \mathsf{Ty}\,\Gamma$ with decoding $\mathsf{El} : (a : \mathsf{Tm}\,\Gamma\,\mathsf{U}) \to \mathsf{Ty}\,\Gamma$.

Inductive function space $\Pi : (a : \mathsf{Tm}\,\Gamma\,\mathsf{U}) \to \mathsf{Ty}\,(\Gamma \triangleright \mathsf{El}\,a) \to \mathsf{Ty}\,\Gamma$, with application as $\mathsf{app} : \mathsf{Tm}\,\Gamma\,(\Pi\,a\,B) \to \mathsf{Tm}\,(\Gamma \triangleright \mathsf{El}\,a)\,B$ and its inverse lam.

External function space $\Pi^{\mathsf{ext}} : (A : \mathsf{Set}_j) \to (A \to \mathsf{Ty}\,\Gamma) \to \mathsf{Ty}\,\Gamma$, with $\mathsf{app}^{\mathsf{ext}} : \mathsf{Tm}\,\Gamma\,(\Pi^{\mathsf{ext}}\,A\,B) \to ((x : A) \to \mathsf{Tm}\,\Gamma\,(B\,x))$ and its inverse $\mathsf{lam}^{\mathsf{ext}}$.

Infinitary function space $\Pi^{\mathsf{inf}} : (A : \mathsf{Set}_j) \to (A \to \mathsf{Tm}\,\Gamma\,\mathsf{U}) \to \mathsf{Tm}\,\Gamma\,\mathsf{U}$, with $\mathsf{app}^{\mathsf{inf}} : \mathsf{Tm}\,\Gamma\,(\mathsf{El}\,(\Pi^{\mathsf{ext}}\,A\,b)) \to ((x : A) \to \mathsf{Tm}\,\Gamma\,(\mathsf{El}\,(b\,x)))$ and its inverse $\mathsf{lam}^{\mathsf{inf}}$.

An identity type $\mathsf{Id} : (a : \mathsf{Tm}\,\Gamma\,\mathsf{U}) \to \mathsf{Tm}\,\Gamma\,(\mathsf{El}\,a) \to \mathsf{Tm}\,\Gamma\,(\mathsf{El}\,a) \to \mathsf{Tm}\,\Gamma\,\mathsf{U}$, with $\mathsf{Refl} : (t : \mathsf{Tm}\,\Gamma\,(\mathsf{El}\,a)) \to \mathsf{Tm}\,\Gamma\,(\mathsf{El}\,(\mathsf{Id}\,a\,t\,t))$, equality reflection and UIP.

In the above listing, we omit equations for substitution and $\beta\eta$-conversion, but these should be understood to be also part of $\mathsf{ToS}_{i,j}$.

Notational conventions. We name elements of Con as Γ, Δ, Θ, elements of $\mathsf{Sub}\,\Gamma\,\Delta$ as σ, δ, ν, elements of $\mathsf{Ty}\,\Gamma$ as A, B, C, and elements of $\mathsf{Tm}\,\Gamma\,A$ as t, u, v. CwF components by default support de Bruijn indices, which are not easily readable. We use instead a nameful notation for binders in context extension, Π and lam, e.g. as $(\bullet \triangleright (a : U) \triangleright (t : \mathsf{El}\,a))$. We also define a type-theoretic flavor of app for convenience:

$$- \mathbin{@} - : \mathsf{Tm}\,\Gamma\,(\Pi\,a\,B) \to (u : \mathsf{Tm}\,\Gamma\,(\mathsf{El}\,a)) \to \mathsf{Tm}\,\Gamma\,(B[\mathsf{id}, u])$$
$$t \mathbin{@} u \quad :\equiv (\mathsf{app}\,t)[\mathsf{id}, u]$$

We abbreviate non-dependent inductive Π as $- \Rightarrow -$, and likewise we use $- \Rightarrow^{\mathsf{ext}} -$ and $- \Rightarrow^{\mathsf{inf}} -$ for non-dependent external and infinitary functions.

Definition 3.2 (Notion of signature)**.** A QIIT signature at level j is a context in an arbitrary $M : \mathsf{ToS}_{i,j}$ model. We define the type of such signatures as follows:

$$\mathsf{Sig}_j :\equiv (i : \mathbb{N}) \to (M : \mathsf{ToS}_{i,j}) \to \mathsf{Con}_M$$

Here, j refers to the level of external types appearing in the signature, in the domains of Π^{ext} and Π^{inf} functions, while the quantified i level is required to allow interpreting a signature in arbitrary-sized ToS models. Note that Sig_j is universe-polymorphic, so it is a family of cETT types and it is not in any cETT universe.

Example 3.3. Signature for natural numbers. Here, no external types appear, so the level can be chosen as 0.

$$\mathsf{NatSig} : \mathsf{Sig}_0$$
$$\mathsf{NatSig} :\equiv \lambda(i : \mathbb{N})(M : \mathsf{ToS}_{i,0}).$$
$$(\bullet_M \triangleright_M (N : \mathsf{U}_M) \triangleright_M (\mathit{zero} : \mathsf{El}_M\,N)$$
$$\triangleright_M (\mathit{suc} : N \Rightarrow_M \mathsf{El}_M\,N))$$

With this, we are able to specify QIITs, and we can also interpret each signature in an arbitrary ToS model, by applying a signature to a model. Sig_j can be viewed as a precursor to a Church-encoding for the theory of signatures, but we only need contexts encoded in this way, and not other ToS components. In functional programming, this is sometimes called "finally tagless" [11], and it is used for defining and interpreting embedded languages.

In the following examples, we leave the abstracted $M : \mathsf{ToS}_{i,j}$ implicit.

Example 3.4. Infinitary constructors. The universe U is closed under the Π^{inf} function type, which allows such functions to appear in the domains of Π types. This allows, for example, a signature for trees branching with arbitrary small types. This is a signature at level 1, since we have Set_0 as a Π^{ext} domain type.

$$\mathsf{TreeSig} :\equiv$$
$$\bullet \triangleright (\mathit{Tree} : \mathsf{U})$$
$$\triangleright (\mathit{node} : \Pi^{\mathsf{ext}}\,\mathsf{Set}_0\,(\lambda A.\,(A \Rightarrow^{\mathsf{inf}} \mathit{Tree}) \Rightarrow \mathsf{El}\,\mathit{Tree}))$$

Example 3.5. Recursive equations. Again, the universe is closed under Id, which allows us to write equations in Π domains. A minimal (and trivial) example:

$$\mathsf{RecEqSig} :\equiv$$
$$\bullet \triangleright (A : \mathsf{U}) \triangleright (a : \mathsf{El}\,A) \triangleright (f : \Pi\,(x : A)\,(\mathsf{Id}\,A\,x\,a \Rightarrow \mathsf{El}\,A))$$

More interesting (and complicated) examples for recursive equations are boundary conditions in various cubical type theories [3, 4, 15]. Note that our Id allows iterated equations as well, but these are all trivial in the semantics, where we assume UIP.

Remark. Since signatures are parametrized by a single universe level, all external types in constructors must be contained in the same Set_j universe. We opted for this setup for the sake of simplicity. Cumulativity helps here: it allows us to pick a j level which is large enough to accommodate all external types in a signature.

4 Semantics

4.1 Overview

For each signature, we would like to have at least

1. A category of algebras, with homomorphisms as morphisms.
2. A notion of induction, which requires a notion of dependent algebras.
3. A proof that for algebras, initiality is equivalent to supporting induction.

Following [24], we do this by creating a model of ToS, where contexts are categories supporting the above requirements and substitutions are appropriate structure-preserving functors. Then, each signature can applied to this model,

yielding an interpretation of the signature as a structured category of algebras.

Our semantics has a "type-theoretic" flavor, which is inspired by the cubical set model of Martin-Löf type theory by Bezem et al. [8]. The core idea is to avoid strictness issues by starting from basic ingredients which are already strict enough. Hence, instead of modeling types as certain slices and substitution by pullback, we model types as displayed categories with extra structure, which naturally support strict reindexing.

We make a similar choice in the interpretation of signatures themselves: we use structured CwFs instead of lex categories. The reason here is that CwFs allow us to compute induction principles in strictly the same way as one would write them down in type theory, since we have Ty and Tm for a primitive notion of dependent objects and morphisms. In contrast, dependent objects in lex categories is a derived notion, and the induction principles we get are only up to isomorphism. This issue is perhaps not relevant from a purely categorical perspective, but we are concerned with eventually implementing QIITs in proof assistants, so we prefer if our semantics computes strictly.

In the following, for given i and j levels, we define a model $\mathbf{M}_{i,j} : \mathrm{ToS}_{\max(i+1,j)+1,j}$ such that $\mathrm{Con}_{\mathbf{M}_{i,j}}$ is a type of structured categories (of algebras). The level i marks the level of all internal sorts in an algebra, and the level j marks the level of all external sets in function domains. Hence, every algebra has level $\max(i+1, j)$. The bump is only needed for i, since algebras merely contain elements of $A : \mathrm{Set}_j$ types, while inductive sets are themselves elements of Set_i. For example, $NatAlg_i : \mathrm{Set}_{\max(i+1,0)} : \mathrm{Set}_{\max(i+1,0)+1}$.

We present the components of the model in order. In the following, we use **bold** font to disambiguate components of $\mathbf{M}_{i,j}$ from components of other structures. For example, we use $\boldsymbol{\sigma} : \mathbf{Sub}\ \boldsymbol{\Gamma}\ \boldsymbol{\Delta}$ to denote a substitution in $\mathbf{M}_{i,j}$.

The model involves a large amount of technical detail; we omit a significant part of this, and only present the most salient parts.

4.2 Contexts

We define $\mathbf{Con} : \mathrm{Set}_{\max(i+1,j)+1}$ as $\mathrm{flCwF}_{\max(i+1,j)}$.

Definition 4.1 (Finite limit CwFs)**.** For each level i we define $\mathrm{flCwF}_i : \mathrm{Set}_{i+1}$ as an iterated Σ-type with the following components:

1. A CwF with underlying sets all in Set_i. We reuse the component notations from Definition 3.1.
2. Σ-type $\Sigma : (A : \mathrm{Ty}\,\Gamma) \to \mathrm{Ty}\,(\Gamma \triangleright A) \to \mathrm{Ty}\,\Gamma$, with term formers proj1, proj2 and $-,-$.
3. Identity type $\mathrm{Id} : (A : \mathrm{Ty}\,\Gamma) \to \mathrm{Tm}\,\Gamma\,A \to \mathrm{Tm}\,\Gamma\,A \to \mathrm{Ty}\,\Gamma$, with refl, equality reflection and UIP.
4. Constant families. This includes a type former $\mathsf{K} : \mathrm{Con} \to \mathrm{Ty}\,\Gamma$, where Γ is implicitly quantified, together with $\mathrm{lam}^{\mathsf{K}} : \mathrm{Sub}\,\Gamma\,\Delta \to \mathrm{Tm}\,\Gamma\,(\mathsf{K}\,\Delta)$ and its inverse $\mathrm{app}^{\mathsf{K}}$. The idea is that $\mathsf{K}\,\Delta$ is a representation of Δ as a type in any context. Clairambault and Dybjer called constant families "democracy" in [14].

Definition 4.2. We abbreviate the additional structure on CwFs consisting of Σ, Id and K as *fl-structure*.

Definition 4.3 (Notion of induction in an flCwF)**.** Given $\Gamma : \mathrm{flCwF}_i$, we have the following predicate on contexts:

$$\mathrm{Inductive} : \mathrm{Con}_\Gamma \to \mathrm{Set}_i$$
$$\mathrm{Inductive}\,\Gamma :\equiv (A : \mathrm{Ty}_\Gamma\,\Gamma) \to \mathrm{Tm}_\Gamma\,\Gamma\,A$$

In the categorical semantics for a concrete signature, this informally means that an algebra supports induction if for any dependent algebra over it (which is a bundle of induction motives and methods), there is a dependent morphism into it (which is a bundle of eliminator functions and their β-rules).

Theorem 4.4 (Equivalence of initiality and induction, c.f. [24])**.** *An object $\Gamma : \mathrm{Con}_\Gamma$ supports induction if and only if it is initial. Moreover, induction and initiality are both proof-irrelevant predicates.* □

The reason for the "finite limit CwF" naming is the following: Clairambault and Dybjer showed that the 2-category of flCwFs is biequivalent to the 2-category of finitely complete categories [14]. In particular, in an flCwF the categorical product of Γ and Δ can be given as $\Gamma \triangleright \mathsf{K}\,\Delta$, and the equalizer of σ and δ as $\Gamma \triangleright \mathrm{Id}\,(\mathsf{K}\,\Delta)\,(\mathrm{lam}^{\mathsf{K}}\,\sigma)\,(\mathrm{lam}^{\mathsf{K}}\,\delta)$.

In order to talk about weak structure-preservation in the interpretation of substitutions, we need to specify isomorphisms for contexts and types.

Definition 4.5. A *context isomorphism* is an invertible morphism $\sigma : \mathrm{Sub}\,\Gamma\,\Delta$. We note the inverse as σ^{-1}. We also use the notation $\sigma : \Gamma \simeq \Delta$.

Definition 4.6 (Type categories, c.f. [14])**.** For each $\Gamma : \mathrm{Con}$, there is a category whose objects are types $A : \mathrm{Ty}\,\Gamma$, and morphisms from A to B are terms $t : \mathrm{Tm}\,(\Gamma \triangleright A)\,B[\mathsf{p}]$. Identity morphisms are given by $\mathsf{q} : \mathrm{Tm}\,(\Gamma \triangleright A)\,A[\mathsf{p}]$, and composition $t \circ u$ by $t[\mathsf{p}, u]$. The assignment of type categories to contexts extends to a split indexed category. For each $\sigma : \mathrm{Sub}\,\Gamma\,\Delta$, there is a functor from $\mathrm{Ty}\,\Delta$ to $\mathrm{Ty}\,\Gamma$, which sends A to $A[\sigma]$ and $t : \mathrm{Tm}\,(\Gamma \triangleright A)\,B[\mathsf{p}]$ to $t[\sigma \circ \mathsf{p}, \mathsf{q}]$.

Definition 4.7. A *type isomorphism*, notated $t : A \simeq B$ is an isomorphism in a type category. We note the inverse as t^{-1}.

4.3 Substitutions

A *weak flCwF morphism* $\boldsymbol{\sigma} : \mathbf{Sub}\ \boldsymbol{\Gamma}\ \boldsymbol{\Delta}$ is a functor between underlying categories, which additionally maps types to types and terms to terms, and satisfies the following:

1. $\boldsymbol{\sigma}\,(A[\sigma]) = (\boldsymbol{\sigma}\,A)\,[\boldsymbol{\sigma}\,\sigma]$
2. $\boldsymbol{\sigma}\,(t[\sigma]) = (\boldsymbol{\sigma}\,t)\,[\boldsymbol{\sigma}\,\sigma]$
3. The unique map $\epsilon : \mathrm{Sub}\,(\boldsymbol{\sigma}\,\bullet)\,\bullet$ has a retraction.
4. Each $(\boldsymbol{\sigma}\,\mathsf{p},\ \boldsymbol{\sigma}\,\mathsf{q}) : \mathrm{Sub}\,(\boldsymbol{\sigma}\,(\Gamma \triangleright A))\,(\boldsymbol{\sigma}\,\Gamma \triangleright \boldsymbol{\sigma}\,A)$ has an inverse.

In short, $\boldsymbol{\sigma}$ preserves substitution strictly and preserves empty context and context extension up to isomorphism. We notate the evident isomorphisms as $\boldsymbol{\sigma}_\bullet : \boldsymbol{\sigma}\,\bullet \simeq \bullet$ and $\boldsymbol{\sigma}_\triangleright : \boldsymbol{\sigma}\,(\Gamma \triangleright A) \simeq \boldsymbol{\sigma}\,\Gamma \triangleright \boldsymbol{\sigma}\,A$. Our notion of weak morphism is the same as in [9], when restricted to CwFs.

Note that the definition we just gave lives in $\mathrm{Set}_{\max(i+1,j)}$, but by cumulativity it is also in $\mathrm{Set}_{\max(i+1,j)+1}$, as required by our $\mathbf{M}_{i,j} : \mathrm{ToS}_{\max(i+1,j)+1,j}$ specification of the model being defined.

Theorem 4.8. *Every $\boldsymbol{\sigma} : \mathbf{Sub}\,\boldsymbol{\Gamma}\,\boldsymbol{\Delta}$ preserves fl-structure up to type isomorphism. That is, we have*

$$\boldsymbol{\sigma}_\Sigma : \boldsymbol{\sigma}\,(\Sigma\,A\,B) \simeq \Sigma\,(\boldsymbol{\sigma}\,A)\,((\boldsymbol{\sigma}\,B)[\boldsymbol{\sigma}_\triangleright^{-1}])$$
$$\boldsymbol{\sigma}_\mathsf{K} : \boldsymbol{\sigma}\,(\mathsf{K}\,\Delta) \simeq \mathsf{K}\,(\boldsymbol{\sigma}\,\Delta)$$
$$\boldsymbol{\sigma}_\mathsf{Id} : \boldsymbol{\sigma}\,(\mathsf{Id}\,t\,u) \simeq \mathsf{Id}\,(\boldsymbol{\sigma}\,t)\,(\boldsymbol{\sigma}\,u)$$

These are all natural in the following sense: for $\sigma : \mathrm{Sub}_\Gamma\,\Gamma\,\Delta$, the functorial action of $\boldsymbol{\sigma}\,\sigma : \mathrm{Sub}_\Delta\,(\boldsymbol{\sigma}\,\Gamma)\,(\boldsymbol{\sigma}\,\Delta)$ on $\boldsymbol{\sigma}_\Sigma$ (in the $\boldsymbol{\sigma}\,\Gamma$ context) is equal to $\boldsymbol{\sigma}_\Sigma$ (in $\boldsymbol{\sigma}\,\Delta$), and similarly for $\boldsymbol{\sigma}_\mathsf{K}$ and $\boldsymbol{\sigma}_\mathsf{Id}$.

Moreover, $\boldsymbol{\sigma}$ preserves all term and substitution formers in the fl-structure. For example, $\boldsymbol{\sigma}\,(\mathrm{proj1}\,t) = \mathrm{proj1}\,(\boldsymbol{\sigma}_\Sigma[\mathrm{id}, \boldsymbol{\sigma}\,t])$.

Proof. For $\boldsymbol{\sigma}_\Sigma$, we construct the following context isomorphism:

$$(\boldsymbol{\sigma}\,\Gamma \triangleright \boldsymbol{\sigma}\,(\Sigma\,A\,B)) \simeq (\boldsymbol{\sigma}\,\Gamma \triangleright \boldsymbol{\sigma}\,A \triangleright (\Sigma\,B)[\boldsymbol{\sigma}_\triangleright^{-1}])$$
$$\simeq (\boldsymbol{\sigma}\,\Gamma \triangleright \Sigma\,(\boldsymbol{\sigma}\,A)\,((\Sigma\,B)[\boldsymbol{\sigma}_\triangleright^{-1}]))$$

This isomorphism is the identity on $\boldsymbol{\sigma}\,\Gamma$, hence we can extract the desired $\boldsymbol{\sigma}_\Sigma : \boldsymbol{\sigma}\,\Sigma\,A\,B) \simeq \Sigma\,(\boldsymbol{\sigma}\,A)\,((\boldsymbol{\sigma}\,B)[\boldsymbol{\sigma}_\triangleright^{-1}])$ from it.

For $\boldsymbol{\sigma}_\mathsf{K}$, note the following:

$$(\bullet \triangleright \boldsymbol{\sigma}\,(\mathsf{K}\,\Delta)) \simeq (\boldsymbol{\sigma}\,\bullet \triangleright \boldsymbol{\sigma}\,(\mathsf{K}\,\Delta)) \simeq \boldsymbol{\sigma}\,(\bullet \triangleright \mathsf{K}\,\Delta)$$
$$\simeq \boldsymbol{\sigma}\,\Delta \simeq (\bullet \triangleright K\,(\boldsymbol{\sigma}\,\Delta))$$

This yields a type isomorphism $\boldsymbol{\sigma}\,(\mathsf{K}\,\Delta)) \simeq \mathsf{K}\,(\boldsymbol{\sigma}\,\Delta)$ in the empty context, and we use the functorial action of $\epsilon : \mathrm{Sub}\,\Gamma\,\bullet$ to weaken it to any Γ context.

For $\boldsymbol{\sigma}_\mathsf{Id}$, both component morphisms can be constructed by refl and equality reflection, and the morphisms are inverses by UIP. We omit here the verification of naturality and that $\boldsymbol{\sigma}$ preserves term and substitution formers in the fl-structure. □

4.4 Identity and Composition

$\mathbf{id} : \mathbf{Sub}\,\boldsymbol{\Gamma}\,\boldsymbol{\Gamma}$ is defined in the obvious way, with identities for underlying functions and for preservation morphisms.

For $\boldsymbol{\sigma} \circ \boldsymbol{\delta}$, the underlying functions are given by function composition, and the preservation morphisms are given as follows:

$$(\boldsymbol{\sigma} \circ \boldsymbol{\delta})_\bullet^{-1} :\equiv \boldsymbol{\sigma}\,\boldsymbol{\delta}_\bullet^{-1} \circ \boldsymbol{\delta}_\bullet^{-1}$$
$$(\boldsymbol{\sigma} \circ \boldsymbol{\delta})_\triangleright^{-1} :\equiv \boldsymbol{\sigma}\,\boldsymbol{\delta}_\triangleright^{-1} \circ \boldsymbol{\delta}_\triangleright^{-1}$$

It is easy to verify the left and right identity laws and associativity for $- \circ -$.

Lemma 4.9. *The derived preservation isomorphisms for the fl-structure can be decomposed analogously; all derived isomorphisms in* $\mathbf{id}$ *are identities, and we have*

$$(\boldsymbol{\sigma} \circ \boldsymbol{\delta})_\Sigma = \boldsymbol{\sigma}\,\boldsymbol{\delta}_\Sigma \circ \boldsymbol{\delta}_\Sigma$$
$$(\boldsymbol{\sigma} \circ \boldsymbol{\delta})_\mathsf{K} = \boldsymbol{\sigma}\,\boldsymbol{\delta}_\mathsf{K} \circ \boldsymbol{\delta}_\mathsf{K}$$
$$(\boldsymbol{\sigma} \circ \boldsymbol{\delta})_\mathsf{Id} = \boldsymbol{\sigma}\,\boldsymbol{\delta}_\mathsf{Id} \circ \boldsymbol{\delta}_\mathsf{Id}$$

On the right sides, $- \circ -$ refers to composition of type morphisms.

Proof. In the case of Id, the equations hold immediately by UIP. For Σ and K, we prove by flCwF computation and straightforward unfolding of definitions. □

4.5 Empty Context

The empty context $\bullet : \mathbf{Con}$ is the terminal flCwF, which has all underlying sets defined as $\top$ (or constantly $\top$), with an evident unique $\boldsymbol{\epsilon} : \mathbf{Sub}\,\boldsymbol{\Gamma}\,\bullet$. Since $\boldsymbol{\epsilon}$ is a strict flCwF morphism, $\boldsymbol{\epsilon}_\bullet^{-1}$ and $\boldsymbol{\epsilon}_\triangleright^{-1}$ are both identity morphisms.

4.6 Types

We define $\mathbf{Ty}\,\boldsymbol{\Gamma} : \mathrm{Set}_{\max(i+1,j)+1}$ as the type of split flCwF-isofibrations over $\boldsymbol{\Gamma}$, at level $\max(i+1, j)$. We extend Ahrens' and Lumsdaine's displayed categories and their definition of isofibrations [1]. We first define displayed flCwFs, then specify iso-cleaving as additional structure on top of that.

Definition 4.10 (Displayed flCwF)**.** The type of displayed flCwFs at level i is given as the logical predicate interpretation (see e.g. [7] or [23]) of flCwF_i. For each flCwF component in $\boldsymbol{\Gamma}$, there is a component in a displayed flCwF which "lies over" it.

Notation. In situations where we need to refer to both "base" and displayed things, we give underlined names to contexts, substitutions, types and terms in a base flCwF. For example, we may have $\underline{\Gamma} : \mathrm{Con}_\Gamma$ living in $\boldsymbol{\Gamma} : \mathbf{Con}$, and $\Gamma : \mathrm{Con}_{\boldsymbol{A}}\,\underline{\Gamma}$ living in a displayed flCwF over $\boldsymbol{\Gamma}$. We only use underlining on cETT variable names, and overload flCwF component names for displayed counterparts. For example, a Con component is named the same in a base flCwF and a displayed one.

Concretely, a displayed flCwF $\boldsymbol{A}$ over $\boldsymbol{\Gamma}$ has the following underlying sets, which we call displayed contexts, substitutions, types and terms respectively.

$$\mathrm{Con}_{\boldsymbol{A}} : \mathrm{Con}_\Gamma \to \mathrm{Set}_i$$
$$\mathrm{Sub}_{\boldsymbol{A}} : \mathrm{Con}_{\boldsymbol{A}}\,\underline{\Gamma} \to \mathrm{Con}_{\boldsymbol{A}}\,\underline{\Delta} \to \mathrm{Sub}_\Gamma\,\underline{\Gamma}\,\underline{\Delta} \to \mathrm{Set}_i$$
$$\mathrm{Ty}_{\boldsymbol{A}} : \mathrm{Con}_{\boldsymbol{A}}\,\underline{\Gamma} \to \mathrm{Ty}_\Gamma\,\underline{\Gamma} \to \mathrm{Set}_i$$
$$\mathrm{Tm}_{\boldsymbol{A}} : (\Gamma : \mathrm{Con}_{\boldsymbol{A}}\,\underline{\Gamma}) \to \mathrm{Ty}_{\boldsymbol{A}}\,\Gamma\,\underline{A} \to \mathrm{Tm}_\Gamma\,\underline{\Gamma}\,\underline{A} \to \mathrm{Set}_i$$

Above, we implicitly quantify over $\underline{\Gamma}$, $\underline{\Delta}$ and $\underline{A}$ base parameters. We also have the following components for empty context, context extension and substitution. We omit listing

other components here.

$$\begin{aligned}
\bullet_A &: \mathsf{Con}_A\,\bullet_\Gamma\\
\triangleright_A &: (\Gamma : \mathsf{Con}_A\,\underline{\Gamma}) \to \mathsf{Ty}_A\,\Gamma\,\underline{A} \to \mathsf{Con}_A\,\Gamma\,(\underline{\Gamma} \triangleright_\Gamma \underline{A})\\
-[-]_A &: \mathsf{Ty}_A\,\Delta\,\underline{A} \to \mathsf{Sub}_A\,\Gamma\,\Delta\,\underline{\sigma} \to \mathsf{Ty}_A\,\Gamma\,(\underline{A}[\underline{\sigma}]_\Gamma)\\
-[-]_A &: \mathsf{Tm}_A\,\Delta\,A\,\underline{t} \to (\sigma : \mathsf{Sub}_A\,\Gamma\,\Delta\,\underline{\sigma})\\
&\quad \to \mathsf{Tm}_A\,\Gamma\,(A[\sigma]_A)\,(\underline{t}[\underline{\sigma}]_\Gamma)
\end{aligned}$$

In the following we will often omit $_\Gamma$ and $_A$ subscripts on components; for example, in the type $\mathsf{Con}_A\,\bullet$, the $\bullet$ is clearly a base component in $\boldsymbol{\Gamma}$.

We also need displayed counterparts to the previously defined derived notions on flCwFs; these are again given as logical predicate interpretations of the non-displayed definitions.

Definition 4.11 (Displayed type categories)**.** For each $\Gamma : \mathsf{Con}_A\,\underline{\Gamma}$, there is a displayed category over the type category $\mathsf{Ty}_\Gamma\,\underline{\Gamma}$, whose objects over $\underline{A} : \mathsf{Ty}_\Gamma\,\underline{\Gamma}$ are elements of $\mathsf{Ty}_A\,\Gamma\,\underline{A}$, and displayed morphisms over $\underline{t} : \mathsf{Tm}_\Gamma\,(\underline{\Gamma} \triangleright \underline{A})\,(\underline{B}[\mathsf{p}])$ are elements of $\mathsf{Tm}_A\,(\Gamma \triangleright A)\,(B[\mathsf{p}])\,\underline{t}$. The identity morphism is given by q_A, and the composition of t and u is $t[\mathsf{p}_A, u]$. Analogously to Definition 4.6, this extends to a displayed split indexed category.

Definition 4.12 (Displayed isomorphisms)**.** A *displayed context isomorphism* over $\underline{\sigma} : \underline{\Gamma} \simeq \underline{\Delta}$, notated $\sigma : \Gamma \simeq_{\underline{\sigma}} \Delta$, is an invertible displayed morphism $\sigma : \mathsf{Sub}_A\,\Gamma\,\Delta\,\underline{\sigma}$, with inverse $\sigma^{-1} : \mathsf{Sub}_A\,\Delta\,\Gamma\,\underline{\sigma}^{-1}$. A *displayed type isomorphism* over $\underline{t} : \underline{A} \simeq \underline{B}$, notated $t : A \simeq_{\underline{t}} B$, is an isomorphism in a displayed type category.

Definition 4.13. A *vertical morphism* lies over an identity morphism. We use this definition for context morphisms (substitutions) and type morphisms as well.

In contrast to [24], it is not sufficient to model types with displayed flCwFs. In ibid. substitutions are modeled as strict morphisms, which we cannot do because our infinitary function types and identity types cannot be interpreted strictly, as we will see in Section 4.15. Strict preservation is expressed with metatheoretic equality, which is respected by all cETT constructions, but isomorphisms are not necessarily respected. Hence, we need to add additional structure to displayed flCwFs which expresses preservation of base isomorphisms.

Definition 4.14 (Context iso-cleaving)**.** This lifts a base context isomorphism to a displayed one. It consists of

$$\begin{aligned}
\mathsf{coe} &: \underline{\Gamma} \simeq \underline{\Delta} \to \mathsf{Con}_A\,\underline{\Gamma} \to \mathsf{Con}_A\,\underline{\Delta}\\
\mathsf{coh} &: (\underline{\sigma} : \underline{\Gamma} \simeq \underline{\Delta})(\Gamma : \mathsf{Con}_A\,\underline{\Gamma}) \to \Gamma \simeq_{\underline{\sigma}} \mathsf{coe}\,\underline{\sigma}\,\Gamma\\
\mathsf{coe}^{\mathsf{id}} &: \mathsf{coe}\,\mathsf{id}\,\Gamma = \Gamma\\
\mathsf{coe}^{\circ} &: \mathsf{coe}\,(\underline{\sigma} \circ \underline{\delta})\,\Gamma = \mathsf{coe}\,\underline{\sigma}\,(\mathsf{coe}\,\underline{\delta}\,\Gamma)\\
\mathsf{coh}^{\mathsf{id}} &: \mathsf{coh}\,\mathsf{id}\,\Gamma = \mathsf{id}\\
\mathsf{coh}^{\circ} &: \mathsf{coh}\,(\underline{\sigma} \circ \underline{\delta})\,\Gamma = \mathsf{coh}\,\underline{\sigma}\,(\mathsf{coe}\,\underline{\delta}\,\Gamma) \circ \mathsf{coh}\,\underline{\delta}\,\Gamma
\end{aligned}$$

Here, coe and coh abbreviate "coercion" and "coherence" respectively.

Definition 4.15 (Type iso-cleaving)**.** This consists of

$$\begin{aligned}
\mathsf{coe} &: \underline{A} \simeq \underline{B} \to \mathsf{Ty}_A\,\Gamma\,\underline{A} \to \mathsf{Ty}_A\,\Gamma\,\underline{B}\\
\mathsf{coh} &: (\underline{t} : \underline{A} \simeq \underline{B})(A : \mathsf{Ty}_A\,\Gamma\,\underline{A}) \to A \simeq_{\underline{t}} \mathsf{coe}\,\underline{t}\,A\\
\mathsf{coe}^{\mathsf{id}} &: \mathsf{coe}\,\mathsf{id}\,A = A\\
\mathsf{coe}^{\circ} &: \mathsf{coe}\,\underline{t}\,(\mathsf{coe}\,\underline{\delta}\,A) = \mathsf{coe}\,(\underline{t} \circ \underline{\delta})\,A\\
\mathsf{coh}^{\mathsf{id}} &: \mathsf{coh}\,\mathsf{id}\,A = \mathsf{id}\\
\mathsf{coh}^{\circ} &: \mathsf{coh}\,(\underline{t} \circ \underline{\delta})\,A = \mathsf{coh}\,\underline{t}\,(\mathsf{coe}\,\underline{\delta}\,A) \circ \mathsf{coh}\,\underline{\delta}\,A
\end{aligned}$$

Additionally, for $\sigma : \mathsf{Sub}_A\,\Gamma\,\Delta\,\underline{\sigma}$, we have

$$\begin{aligned}
\mathsf{coe}[] &: \mathsf{coe}\,(\underline{t}[\underline{\sigma}])\,(A[\sigma]) = (\mathsf{coe}\,\underline{t}\,A)[\sigma]\\
\mathsf{coh}[] &: \mathsf{coh}\,(\underline{t}[\underline{\sigma} \circ \mathsf{p}, \mathsf{q}])\,(A[\sigma]) = (\mathsf{coh}\,\underline{t}\,A)[\sigma]
\end{aligned}$$

Definition 4.16. A *split flCwF isofibration* is a displayed flCwF equipped with iso-cleaving for contexts and types.

Remark. It is not possible to model types as fibrations or opfibrations, because we have no restriction on the variance of ToS types. For example, the type which extends a pointed set to a natural number signature, is neither a fibration nor an opfibration.

4.7 Type Substitution

We aim to define $-[-] : \mathbf{Ty}\,\Delta \to \mathbf{Sub}\,\Gamma\,\Delta \to \mathbf{Ty}\,\Gamma$, such that $A[\mathbf{id}] = A$ and $A[\sigma \circ \delta] = A[\sigma][\delta]$. The underlying sets are given by simple composition:

$$\begin{aligned}
\mathsf{Con}_{A[\sigma]}\,\underline{\Gamma} &:\equiv \mathsf{Con}_A\,(\boldsymbol{\sigma}\,\underline{\Gamma})\\
\mathsf{Sub}_{A[\sigma]}\,\Gamma\,\Delta\,\underline{\sigma} &:\equiv \mathsf{Sub}_A\,\Gamma\,\Delta\,(\boldsymbol{\sigma}\,\underline{\sigma})\\
\mathsf{Ty}_{A[\sigma]}\,\Gamma\,\underline{A} &:\equiv \mathsf{Ty}_A\,\Gamma\,(\boldsymbol{\sigma}\,\underline{A})\\
\mathsf{Tm}_{A[\sigma]}\,\Gamma\,A\,\underline{t} &:\equiv \mathsf{Tm}_A\,\Gamma\,A\,(\boldsymbol{\sigma}\,\underline{t})
\end{aligned}$$

Moreover, $\mathsf{id}_{A[\sigma]} :\equiv \mathsf{id}_A$, $\sigma \circ_{A[\sigma]} \delta :\equiv \sigma \circ_A \delta$, and likewise substitution components are given by corresponding substitution components in $\boldsymbol{A}$.

Context and type formers are given by coercing $\boldsymbol{A}$ structures along $\boldsymbol{\sigma}$ preservation isomorphisms. For example:

$$\begin{aligned}
\bullet_{A[\sigma]} &:\equiv \mathsf{coe}\,\boldsymbol{\sigma}_\bullet^{-1}\,\bullet_A\\
\Gamma \triangleright_{A[\sigma]} A &:\equiv \mathsf{coe}\,\boldsymbol{\sigma}_\triangleright^{-1}\,(\Gamma \triangleright_A A)\\
\mathsf{Id}_{A[\sigma]}\,t\,u &:\equiv \mathsf{coe}\,\boldsymbol{\sigma}_{\mathsf{Id}}^{-1}\,(\mathsf{Id}_A\,t\,u)
\end{aligned}$$

Term and substitution formers are given by composing coh-lifted isomorphisms with term and substitution formers from $\boldsymbol{A}$. For example:

$$\begin{aligned}
\epsilon_{A[\sigma]} &:\equiv \mathsf{coh}\,\boldsymbol{\sigma}_\bullet^{-1}\,\bullet_A \circ \epsilon_A\\
\mathsf{p}_{A[\sigma]} &:\equiv \mathsf{p}_A \circ (\mathsf{coh}\,\boldsymbol{\sigma}_\triangleright^{-1}\,(\Gamma \triangleright A))^{-1}\\
\mathsf{app}^{\mathsf{K}}_{A[\sigma]}\,t &:\equiv \mathsf{app}^{\mathsf{K}}_A\,((\mathsf{coh}\,\boldsymbol{\sigma}_{\mathsf{K}}\,(\mathsf{K}\,\Delta))^{-1} \circ t)
\end{aligned}$$

Equations for term and type substitution follow from naturality of preservation isomorphisms in $\boldsymbol{\sigma}$, coe[], coh[] and substitution equations in $\boldsymbol{A}$.

Iso-cleaving is given iso-cleaving in $\boldsymbol{A}$ and the action of $\boldsymbol{\sigma}$ on isomorphisms, e.g. we have $\mathsf{coe}_{\boldsymbol{A}[\sigma]}\,\underline{\sigma}\,\Gamma :\equiv \mathsf{coe}_{\boldsymbol{A}}\,(\boldsymbol{\sigma}\,\underline{\sigma})\,\Gamma$.

Functoriality of type substitution, i.e. $\boldsymbol{A}[\mathbf{id}] = \boldsymbol{A}$ and $\boldsymbol{A}[\boldsymbol{\sigma} \circ \boldsymbol{\delta}] = \boldsymbol{A}[\boldsymbol{\sigma}][\boldsymbol{\delta}]$, follows from Lemma 4.9 and split cleaving given by $\mathsf{coe}^{\mathsf{id}}$, coe°, $\mathsf{coh}^{\mathsf{id}}$ and coh° laws in $\boldsymbol{A}$.

4.8 Terms

$\mathbf{Tm}\,\boldsymbol{\Gamma}\,\boldsymbol{A} : \mathsf{Set}_{\max(i+1,j)+1}$ is defined as the type of *weak flCwF sections* of $\boldsymbol{A}$. The underlying functions of $\boldsymbol{t} : \mathbf{Tm}\,\boldsymbol{\Gamma}\,\boldsymbol{A}$ are as follows:

$$\begin{aligned}
&\boldsymbol{t} : (\underline{\Gamma} : \mathsf{Con}_{\Gamma}) \to \mathsf{Con}_{\boldsymbol{A}}\,\underline{\Gamma}\\
&\boldsymbol{t} : (\underline{\sigma} : \mathsf{Sub}_{\Gamma}\,\underline{\Gamma}\,\underline{\Delta}) \to \mathsf{Sub}_{\boldsymbol{A}}\,(\boldsymbol{t}\,\underline{\Gamma})\,(\boldsymbol{t}\,\underline{\Delta})\,\underline{\sigma}\\
&\boldsymbol{t} : (\underline{A} : \mathsf{Ty}_{\Gamma}) \to \mathsf{Ty}_{\boldsymbol{A}}\,(\boldsymbol{t}\,\underline{\Gamma})\,\underline{A}\\
&\boldsymbol{t} : (\underline{t} : \mathsf{Tm}_{\Gamma}\,\underline{\Gamma}\,\underline{A}) \to \mathsf{Tm}_{\boldsymbol{A}}\,(\boldsymbol{t}\,\underline{\Gamma})\,(\boldsymbol{t}\,\underline{A})\,\underline{t}
\end{aligned}$$

Such that

1. $\boldsymbol{t}\,(\underline{A}[\underline{\sigma}]) = (\boldsymbol{t}\,\underline{A})\,[\boldsymbol{t}\,\underline{\sigma}]$
2. $\boldsymbol{t}\,(\underline{t}[\underline{\sigma}]) = (\boldsymbol{t}\,\underline{t})\,[\boldsymbol{t}\,\underline{\sigma}]$
3. The unique map $\epsilon_{\boldsymbol{A}} : \mathsf{Sub}\,(\boldsymbol{t}\,\bullet)\,\bullet\,\mathsf{id}$ has a vertical retraction.
4. Each $(\boldsymbol{t}\,\mathsf{p},\ \boldsymbol{t}\,\mathsf{q}) : \mathsf{Sub}\,(\boldsymbol{t}\,(\underline{\Gamma} \triangleright \underline{A}))\,(\boldsymbol{t}\,\underline{\Gamma} \triangleright \boldsymbol{t}\,\underline{A})\,\mathsf{id}$ has a vertical inverse.

Similarly to Section 4.3, we denote the evident preservation isomorphisms as $\boldsymbol{t}_{\bullet} : \boldsymbol{t}\,\bullet \simeq_{\mathsf{id}} \bullet$ and $\boldsymbol{t}_{\triangleright} : \boldsymbol{t}\,(\underline{\Gamma} \triangleright \underline{A}) \simeq_{\mathsf{id}} \boldsymbol{t}\,\underline{\Gamma} \triangleright \boldsymbol{t}\,\underline{A}$. In short, weak section is a dependently typed analogue of weak morphism, with dependent underlying functions and displayed preservation isomorphisms. We also have the derived fl-preservation isomorphisms.

Theorem 4.17. *A weak section $\boldsymbol{t} : \mathbf{Tm}\,\boldsymbol{\Gamma}\,\boldsymbol{A}$ preserves fl-structure up to vertical type isomorphisms, that is, the following are derivable:*

$$\begin{aligned}
&\boldsymbol{t}_{\Sigma} : \boldsymbol{t}\,(\Sigma\,\underline{A}\,\underline{B}) \simeq_{\mathsf{id}} \Sigma\,(\boldsymbol{t}\,\underline{A})\,((\boldsymbol{t}\,\underline{B})[\boldsymbol{t}_{\triangleright}^{-1}])\\
&\boldsymbol{t}_{\mathsf{K}} : \boldsymbol{t}\,(\mathsf{K}\,\underline{\Delta}) \simeq_{\mathsf{id}} \mathsf{K}\,(\boldsymbol{t}\,\underline{\Delta})\\
&\boldsymbol{t}_{\mathsf{Id}} : \boldsymbol{t}\,(\mathsf{Id}\,\underline{t}\,\underline{u}) \simeq_{\mathsf{id}} \mathsf{Id}\,(\boldsymbol{t}\,\underline{t})\,(\boldsymbol{t}\,\underline{u})
\end{aligned}$$

Also, the above isomorphisms are natural in the sense of Theorem 4.8, and $\boldsymbol{t}$ preserves type and substitution formers in the fl-structure.

Proof. The construction of isomorphisms is the same as in Theorem 4.8. Indeed, every construction there has a displayed counterpart which we can use here. □

We note though that the move from Theorem 4.8 to here is not simply a logical predicate translation, because we are only lifting the codomain of a weak morphism to a displayed version, and we leave the domain non-displayed. We leave to future work the investigation of such asymmetrical (or "modal") logical predicate translations.

4.9 Term Substitution

$-[-] : \mathbf{Tm}\,\boldsymbol{\Delta}\,\boldsymbol{A} \to (\boldsymbol{\sigma} : \mathbf{Sub}\,\boldsymbol{\Gamma}\,\boldsymbol{\Delta}) \to \mathbf{Tm}\,\boldsymbol{\Gamma}\,(\boldsymbol{A}[\boldsymbol{\sigma}])$ is given similarly to $- \circ -$ in Section 4.4. Underlying functions are given by function composition, and preservation morphisms are also similar:

$$\begin{aligned}
(\boldsymbol{t}[\boldsymbol{\sigma}])_{\bullet}^{-1} &:\equiv \boldsymbol{t}\,\boldsymbol{\sigma}_{\bullet}^{-1} \circ \boldsymbol{t}_{\bullet}^{-1}\\
(\boldsymbol{t}[\boldsymbol{\sigma}])_{\triangleright}^{-1} &:\equiv \boldsymbol{t}\,\boldsymbol{\sigma}_{\triangleright}^{-1} \circ \boldsymbol{t}_{\triangleright}^{-1}
\end{aligned}$$

We also have the same decomposition of derived isomorphisms as in Lemma 4.9. We do not have to show functoriality of term substitution here, since that is derivable in any CwF [**?**].

4.10 Context Extension and Comprehension

$\boldsymbol{\Gamma} \triangleright \boldsymbol{A} : \mathbf{Con}$ is defined as the *total flCwF* of $\boldsymbol{A}$. This is given by bundling together all displayed flCwF components in $\boldsymbol{A}$ with corresponding base components in $\boldsymbol{\Gamma}$, using the metatheoretic Σ-type. It is a straightforward extension of total categories in [1].

$\mathbf{p} : \mathbf{Sub}\,(\boldsymbol{\Gamma} \triangleright \boldsymbol{A})\,\boldsymbol{\Gamma}$ is a strict morphism given by taking a first projection for each component. $\mathbf{q} : \mathbf{Tm}\,(\boldsymbol{\Gamma} \triangleright \boldsymbol{A})\,(\boldsymbol{A}[\mathbf{p}])$ is likewise a strict flCwF section given by second projections. Substitution extension $\boldsymbol{\sigma}, \boldsymbol{t}$ is given by pointwise combining $\boldsymbol{\sigma}$ and $\boldsymbol{t}$ with metatheoretic Σ pairing, e.g. $\mathsf{Con}_{(\sigma,t)}\,\underline{\Gamma} :\equiv (\boldsymbol{\sigma}\,\underline{\Gamma}, \boldsymbol{t}\,\underline{\Gamma})$.

4.11 Universe

Definition 4.18. For a level i, we write $\mathbf{Set}_i$ for the flCwF of sets where $\mathsf{Con}_{\mathbf{Set}_i} :\equiv \mathsf{Set}_i$ and $\mathsf{Sub}_{\mathbf{Set}_i}\,\Gamma\,\Delta :\equiv \Gamma \to \Delta$.

We define $\mathbf{U} : \mathbf{Ty}\,\boldsymbol{\Gamma}$ as the isofibration which is constantly $\mathbf{Set}_i$. A constant isofibration does not actually depend on the base flCwF, and has trivial iso-cleaving where coe-s are identity functions. Hence, we have $\mathsf{Con}_{\mathbf{U}}\,\underline{\Gamma} :\equiv \mathsf{Set}_i$ and $\mathsf{Sub}_{\mathbf{U}}\,\Gamma\,\Delta\,\underline{\sigma} :\equiv \Gamma \to \Delta$.

Remark. The type $\mathbf{Tm}\,\boldsymbol{\Gamma}\,\mathbf{U}$ is strictly equal to $\mathbf{Sub}\,\boldsymbol{\Gamma}\,\mathbf{Set}_i$, so it is helpful to think about semantic elements of the universe as weak morphisms from $\boldsymbol{\Gamma}$ to $\mathbf{Set}_i$.

4.12 Elements of the Universe

We define $\mathbf{El} : \mathbf{Tm}\,\boldsymbol{\Gamma}\,\mathbf{U} \to \mathbf{Ty}\,\boldsymbol{\Gamma}$ as discrete isofibration formation. For $\boldsymbol{a} : \mathbf{Tm}\,\boldsymbol{\Gamma}\,\mathbf{U}$, the underlying sets of $\mathbf{El}\,\boldsymbol{a}$ are the following:

$$\begin{aligned}
\mathsf{Con}_{\mathbf{El}\,\boldsymbol{a}}\,\underline{\Gamma} &:\equiv \boldsymbol{a}\,\underline{\Gamma}\\
\mathsf{Sub}_{\mathbf{El}\,\boldsymbol{a}}\,\Gamma\,\Delta\,\underline{\sigma} &:\equiv \boldsymbol{a}\,\underline{\sigma}\,\Gamma = \Delta\\
\mathsf{Ty}_{\mathbf{El}\,\boldsymbol{a}}\,\Gamma\,\underline{A} &:\equiv \boldsymbol{a}\,\underline{A}\,\Gamma\\
\mathsf{Tm}_{\mathbf{El}\,\boldsymbol{a}}\,\Gamma\,A\,\underline{t} &:\equiv \boldsymbol{a}\,\underline{t}\,\Gamma = A
\end{aligned}$$

Hence, in $\mathbf{El}\,\boldsymbol{a}$, Sub and Tm are propositional. We use the isomorphisms $\boldsymbol{a}_{\bullet} : \boldsymbol{a}\,\bullet \simeq \top$ and $\boldsymbol{a}_{\triangleright} : \boldsymbol{a}\,(\underline{\Gamma} \triangleright \underline{A}) \simeq (\Gamma : \boldsymbol{a}\,\underline{\Gamma}) \times (\boldsymbol{a}\,\underline{A}\,\Gamma)$ to define empty context and context extension:

$$\begin{aligned}
\bullet_{\mathbf{El}\,\boldsymbol{a}} &:\equiv \boldsymbol{a}_{\bullet}^{-1}\,\mathsf{tt}\\
(\Gamma \triangleright_{\mathbf{El}\,\boldsymbol{a}} A) &:\equiv \boldsymbol{a}_{\triangleright}^{-1}\,(\Gamma,\ A)
\end{aligned}$$

We likewise use preservation isomorphisms to define K, Id and Σ. Context coercion is $\mathsf{coe}\,\underline{\sigma}\,\Gamma :\equiv \boldsymbol{a}\,\underline{\sigma}\,\Gamma$. Type coercion, for $A : \boldsymbol{a}\,\underline{A}\,\Gamma$ is given as $\mathsf{coe}\,\underline{t}\,A :\equiv \boldsymbol{a}\,\underline{t}\,(\boldsymbol{a}_{\triangleright}^{-1}\,(\Gamma,\ A))$.

4.13 Inductive Function Space

For $\boldsymbol{a} : \mathbf{Tm}\,\Gamma\,\mathbf{U}$ and $\boldsymbol{B} : \mathbf{Ty}\,(\Gamma \triangleright \mathbf{El}\,\boldsymbol{a})$, we aim to define $\mathbf{\Pi}\,\boldsymbol{a}\,\boldsymbol{B} : \mathbf{Ty}\,\Gamma$. We define this as a dependent product of isofibrations, indexed by a discrete domain. The discreteness is essential: with a general $\boldsymbol{A} : \mathbf{Ty}\,\Gamma$ domain, $\mathbf{\Pi}$ would not be definable because of variance issues. Indeed, the category of categories is not locally cartesian closed and does not support a general Π type [22].

Contexts are products of $\boldsymbol{B}$-contexts, and types are products of $\boldsymbol{B}$-types, indexed respectively by contexts and types of $\mathbf{El}\,\boldsymbol{a}$.

$$\mathsf{Con}_{(\Pi\,a\,B)}\,\underline{\Gamma} :\equiv (\gamma : \boldsymbol{a}\,\underline{\Gamma}) \to \mathsf{Con}_{\boldsymbol{B}}\,(\underline{\Gamma}, \gamma)$$
$$\mathsf{Ty}_{(\Pi\,a\,B)}\,\Gamma\,\underline{A} :\equiv (\gamma : \boldsymbol{a}\,\underline{\Gamma})(a : \boldsymbol{a}\,\underline{A}\,\gamma) \to \mathsf{Ty}_{\boldsymbol{B}}\,(\Gamma\,\gamma)\,(\underline{A}, a)$$

Note that since $\boldsymbol{B}$ is over the total $(\Gamma \triangleright \mathbf{El}\,\boldsymbol{a})$, $\mathsf{Con}_{\boldsymbol{B}}$ has a Σ-typed argument, and likewise the last argument of every $\boldsymbol{B}$ component. We could define substitutions similarly, as products of substitutions:

$$\mathsf{Sub}_{(\Pi\,a\,B)}\,\Gamma\,\Delta\,\underline{\sigma} :\equiv (\gamma : \boldsymbol{a}\,\underline{\Gamma})(\delta : \boldsymbol{a}\,\underline{\Delta})(\sigma : \mathsf{Sub}_{(\mathbf{El}\,a)}\,\gamma\,\delta\,\underline{\sigma}) \to \mathsf{Sub}_{\boldsymbol{B}}\,(\Gamma\,\gamma)\,(\Delta\,\delta)\,(\underline{\sigma}, \sigma)$$

This would work, but we know that $\mathsf{Sub}_{(\mathbf{El}\,a)}\,\gamma\,\delta\,\underline{\sigma}$ is defined as $\boldsymbol{a}\,\underline{\sigma}\,\gamma = \delta$, so we can eliminate σ by singleton contraction, and use the following equivalent definition:

$$\mathsf{Sub}_{(\Pi\,a\,B)}\,\Gamma\,\Delta\,\underline{\sigma} :\equiv (\gamma : \boldsymbol{a}\,\underline{\Gamma}) \to \mathsf{Sub}_{\boldsymbol{B}}\,(\Gamma\,\gamma)\,(\Delta\,(\boldsymbol{a}\,\underline{\sigma}\,\gamma)\,(\underline{\sigma}, \mathsf{refl})$$

The benefit of the contracted definition is that it computes preservation laws in algebra homomorphisms strictly as expected, while the non-contracted definition computes homomorphisms as functional logical relations.

Terms are also given as a singleton-contracted version of products of terms. In $\mathbf{\Pi}\,\boldsymbol{a}\,\boldsymbol{B}$, all other structure is given pointwise by $\boldsymbol{B}$-structure.

Iso-cleaving is given by transporting indices backwards in $\mathbf{El}\,\boldsymbol{a}$ and outputs forwards in $\boldsymbol{B}$:

$$\mathsf{coe}\,\underline{\sigma}\,\Gamma :\equiv \lambda\,\gamma.\,\mathsf{coe}_{\boldsymbol{B}}\,(\underline{\sigma}, \mathsf{refl})\,(\Gamma\,(\boldsymbol{a}\,(\underline{\sigma}^{-1})\,\gamma))$$
$$\mathsf{coe}\,\underline{t}\,A :\equiv \lambda\,\gamma\,a.\,\mathsf{coe}_{\boldsymbol{B}}\,(\underline{t}, \mathsf{refl})\,(A\,(\boldsymbol{a}\,(\underline{t}^{-1})\,(\boldsymbol{a}_{\triangleright}^{-1}(\gamma, a))))$$

Likewise, coh-s are given by backwards-forwards coh-s.

$\mathbf{app} : \mathbf{Tm}\,\Gamma\,(\mathbf{\Pi}\,\boldsymbol{a}\,\boldsymbol{B}) \to \mathbf{Tm}\,(\Gamma \triangleright \mathbf{El}\,\boldsymbol{a})\,\boldsymbol{B}$ can be defined as currying of the underlying functions, and $\mathbf{lam}$ as uncurrying.

4.14 External Function Space

For $A : \mathsf{Set}_j$ and $\boldsymbol{B} : A \to \mathbf{Ty}\,\Gamma$, we define $\mathbf{\Pi}^{\mathbf{ext}}\,A\,\boldsymbol{B} : \mathbf{Ty}\,\Gamma$ as the A-indexed direct product of $\boldsymbol{B}$. Since the indexing is given by a metatheoretic function, every component is given in the evident pointwise way.

4.15 Infinitary Function Space

For $A : \mathsf{Set}_j$ and $\boldsymbol{b} : A \to \mathbf{Tm}\,\Gamma\,\mathbf{U}$, we aim to define $\mathbf{\Pi}^{\mathbf{inf}}\,A\,\boldsymbol{b} : \mathbf{Tm}\,\Gamma\,\mathbf{U}$. The underlying functions are:

$$(\mathbf{\Pi}^{\mathbf{inf}}\,A\,\boldsymbol{b})\,\underline{\Gamma} :\equiv (a : A) \to \boldsymbol{b}\,a\,\underline{\Gamma}$$
$$(\mathbf{\Pi}^{\mathbf{inf}}\,A\,\boldsymbol{b})\,\underline{\sigma} :\equiv \lambda\,a.\,\boldsymbol{b}\,a\,\underline{\sigma}$$
$$(\mathbf{\Pi}^{\mathbf{inf}}\,A\,\boldsymbol{b})\,\underline{A} :\equiv \lambda\,\Gamma.\,(a : A) \to \boldsymbol{b}\,a\,\underline{A}\,(\Gamma\,a)$$
$$(\mathbf{\Pi}^{\mathbf{inf}}\,A\,\boldsymbol{b})\,\underline{t} :\equiv \lambda\,a.\,\boldsymbol{b}\,a\,\underline{t}$$

The preservation morphisms are as follows. Note that $\bullet_{\mathbf{U}} = \top$ and $\triangleright_{\mathbf{U}}$ is metatheoretic Σ.

$$(\mathbf{\Pi}^{\mathbf{inf}}\,A\,\boldsymbol{b})_{\bullet}^{-1} : \top \to (\mathbf{\Pi}^{\mathbf{inf}}\,A\,\boldsymbol{b})\,\bullet$$
$$(\mathbf{\Pi}^{\mathbf{inf}}\,A\,\boldsymbol{b})_{\bullet}^{-1} :\equiv \lambda\,\_\,a.\,(\boldsymbol{b}\,a)_{\bullet}^{-1}\,\mathsf{tt}$$
$$(\mathbf{\Pi}^{\mathbf{inf}}\,A\,\boldsymbol{b})_{\triangleright}^{-1} : (\Gamma : (\mathbf{\Pi}^{\mathbf{inf}}\,A\,\boldsymbol{b})\,\underline{\Gamma}) \times ((\mathbf{\Pi}^{\mathbf{inf}}\,A\,\boldsymbol{b})\,\underline{A}\,\Gamma) \to (\mathbf{\Pi}^{\mathbf{inf}}\,A\,\boldsymbol{b})\,(\underline{\Gamma} \triangleright \underline{A})$$
$$(\mathbf{\Pi}^{\mathbf{inf}}\,A\,\boldsymbol{b})_{\triangleright}^{-1} :\equiv \lambda\,(\Gamma, A)\,a.\,(\boldsymbol{b}\,a)_{\triangleright}^{-1}(\Gamma\,a, A\,a)$$

The preservation of $\bullet$ and $- \triangleright -$ here is in fact the main point of divergence from [24]. In ibid., substitutions and terms are modeled as strict morphisms and types as displayed CwFs (with no iso-cleaving). However, it is not the case that $(\mathbf{\Pi}^{\mathbf{inf}}\,A\,\boldsymbol{b})\,\bullet = \top$, which is the statement of strict $\bullet$-preservation. The left side reduces to $(a : A) \to \boldsymbol{b}\,a\,\bullet$, which is isomorphic to $\top$ but not strictly equal to it. Likewise for $\triangleright$-preservation.

Hence, we are forced to interpret terms as weak sections, which in turn forces us to interpret types as isofibrations, since type substitution requires iso-cleaving.

4.16 Identity

For $\boldsymbol{t}$ and $\boldsymbol{u}$ in $\mathbf{Tm}\,\Gamma\,(\mathbf{El}\,\boldsymbol{a})$, we define $\mathbf{Id}\,\boldsymbol{t}\,\boldsymbol{u} : \mathbf{Tm}\,\Gamma\,\mathbf{U}$ as expressing pointwise equality of weak sections.

$$(\mathbf{Id}\,\boldsymbol{t}\,\boldsymbol{u})\,\underline{\Gamma} :\equiv \boldsymbol{t}\,\underline{\Gamma} = \boldsymbol{u}\,\underline{\Gamma}$$
$$(\mathbf{Id}\,\boldsymbol{t}\,\boldsymbol{u})\,\underline{A} :\equiv \lambda\,e.\,\boldsymbol{t}\,\underline{A} = \boldsymbol{u}\,\underline{A}$$

Above, $\boldsymbol{t}\,\underline{A} = \boldsymbol{u}\,\underline{A}$ is well-typed because of $e : \boldsymbol{t}\,\underline{\Gamma} = \boldsymbol{u}\,\underline{\Gamma}$. For substitutions, we have to complete a square of equalities:

$$(\mathbf{Id}\,\boldsymbol{t}\,\boldsymbol{u})\,(\underline{\sigma} : \mathsf{Sub}\,\underline{\Gamma}\,\underline{\Delta}) : \boldsymbol{t}\,\underline{\Gamma} = \boldsymbol{u}\,\underline{\Gamma} \to \boldsymbol{t}\,\underline{\Delta} = \boldsymbol{u}\,\underline{\Delta}$$

This can be given by $\boldsymbol{t}\,\underline{\sigma} : \boldsymbol{a}\,\underline{\sigma}\,(\boldsymbol{t}\,\underline{\Gamma}) = \boldsymbol{t}\,\underline{\Delta}$ and $\boldsymbol{u}\,\underline{\sigma} : \boldsymbol{a}\,\underline{\sigma}\,(\boldsymbol{u}\,\underline{\Gamma}) = \boldsymbol{u}\,\underline{\Delta}$. The action on terms is analogous. We omit preservation morphisms here as they are straightforward. Like $\mathbf{\Pi}^{\mathbf{inf}}$, $\mathbf{Id}$ also does not support strict preservation of $\bullet$ and $\triangleright$. Equality reflection and $\mathbf{refl} : \mathbf{Id}\,\boldsymbol{t}\,\boldsymbol{t}$ are also evident.

With this, we have defined the $\mathbf{M}_{i,j} : \mathsf{ToS}_{\max(i+1,j)+1,j}$ model that we set out to define in Section 4.1.

5 Model Theory of the Theory of Signatures

At this point, we only have a notion of algebra for ToS, from Definition 3.1. In the following sections, we would also like to talk about initial ToS-algebras and ToS-induction. We

get these notions by giving a QIIT signature for ToS, and interpreting it in the **M** model from the previous section.

Definition 5.1 (Signature for ToS)**.** For each level j, we define $\mathsf{ToSSig}_j : \mathsf{Sig}_{j+i}$, as the signature for the theory of signatures with external sets in Set_j. This is a large and infinitary QIIT signature, as we have Π^{ext} and Π^{inf} abstracting over $A : \mathsf{Set}_j$ and branching with $A \to \mathsf{Ty}\,\Gamma$ and $A \to \mathsf{Tm}\,\Gamma\,U$ respectively. We present an excerpt from ToSSig_i below.

$$
\begin{aligned}
&\bullet \triangleright (\mathit{Con} : \mathsf{U})\\
&\quad \triangleright (\mathit{Sub} : \mathit{Con} \Rightarrow \mathit{Con} \Rightarrow \mathsf{U})\\
&\quad \triangleright (\mathit{Ty} : \mathit{Con} \Rightarrow \mathsf{U})\\
&\quad \triangleright (\mathit{Tm} : \Pi(\Gamma : \mathit{Con})(\mathsf{Ty}\,\Gamma \Rightarrow \mathsf{U}))\\
&\quad \ldots\\
&\quad \triangleright (\Pi^{\mathsf{inf}} : \Pi(\Gamma : \mathit{Con})\\
&\qquad\qquad (\Pi^{\mathsf{ext}}\mathsf{Set}_j(\lambda\, A.\, (A \Rightarrow^{\mathsf{inf}} \mathsf{Ty}\,\Gamma) \Rightarrow \mathsf{El}(\mathsf{Ty}\,\Gamma))))\\
&\quad \ldots
\end{aligned}
$$

Now, for each i, the interpretation of ToSSig_j in $\mathbf{M}_{i,j+1}$ yields an flCwF $\mathbf{\Gamma}$ such that $\mathsf{Con}_{\mathbf{\Gamma}} = \mathsf{ToS}_{i,j}$. In short, we can recover ToS algebras from the semantics of ToSSig. This follows by computation of the interpretation and the fact that ToSSig is precisely the internal representation of ToS. Hence, we have self-description modulo the bumping of the j level. Also, as we get an flCwF of $\mathsf{ToS}_{i,j}$-algebras, we can use Definition 4.3 for the notion of ToS-induction.

Remark. By the definition of $\bullet$ and $- \triangleright -$, the types of algebras computed by **M** are always left-nested iterated Σ-types which start with $\top$. Hence, we need to require that Definition 3.1 is similarly left-nested and starts with $\top$, in order to make the match strict.

6 Term Models of QIITs

In this section we construct QIITs from initial ToS-algebras. For this, we need to assume the existence of such algebras.

6.1 Assuming Syntax for the Theory of Signatures

Lemma 6.1 (Cumulativity of ToS)**.** *If $i \le i'$, then $\mathsf{ToS}_{i,j} \le \mathsf{ToS}_{i',j}$. This follows from the definition of ToS and the subtyping rules in Section 2.2.* □

Assumption. For each level j and j' such that $j + 1 \le j'$, we assume the existence of $\mathsf{syn}_j : \mathsf{ToS}_{j+1,j}$, and we assume that syn_j, considered as an element of $\mathsf{ToS}_{j',j}$ by Lemma 6.1, is inductive in the sense of Definition 4.3.

We explain this assumption. The syntax for the theory of signatures is postulated at the lowest possible level $\mathsf{ToS}_{j+1,j}$. This is the lowest because signatures may contain $A : \mathsf{Set}_j$ types, and since we want to view the syntax as freely generated, its inductive sorts must be large enough to contain the A types. Otherwise we would run into Russell's paradox. Then, the induction assumption says that we have induction at all levels larger than $j + 1$.

Example 6.2. We have $\mathsf{syn}_0 : \mathsf{ToS}_{1,0}$, which is the syntax of closed QIIT signatures. We want to define a function $\mathsf{length} : \mathsf{Con}_{\mathsf{syn}_0} \to \mathbb{N}$ by induction, which returns the length of a syntactic context as a metatheoretic natural number. To this end, we define a displayed ToS over syn_0, where Con is defined as constantly $\mathbb{N}$, every other sort is defined as constantly $\top$, $\bullet$ is defined as 0 and $\Gamma \triangleright A$ is defined as $\Gamma + 1$. By the induction assumption, we get a ToS-section from syn_0 to the displayed model, whose action on contexts is exactly the length function. Note that the induction assumption requires that the displayed model is at least at level 1, but this is not problematic because by cumulativity $\mathbb{N} : \mathsf{Set}_1$.

For every $M : \mathsf{ToS}_{j+1,j}$, there is a unique strict ToS-morphism from syn_j to M. This follows from the induction assumption on syn_j and Theorem 4.4. We denote this morphism as $[\![-]\!]_M$. For example, given $\Gamma : \mathsf{Con}_{\mathsf{syn}}$, we have $[\![\Gamma]\!]_M : \mathsf{Con}_M$. Also, for every displayed ToS-model M over syn_j, there is a strict ToS-section of M. We also denote this as $[\![-]\!]_M$, so e.g. for $\underline{\Gamma} : \mathsf{Con}_{\mathsf{syn}}$ we have $[\![\underline{\Gamma}]\!]_M : \mathsf{Con}_M\,\underline{\Gamma}$.

With syn at hand, we can use an alternative, more conventional representation of signatures.

Definition 6.3. We define $\mathsf{SynSig}_j : \mathsf{Set}_{j+1}$, the type of *syntactic signatures* at j, as $\mathsf{Con}_{\mathsf{syn}_j}$.

We can convert a signature to a syntactic one by interpreting it in some syn_j model, and we can convert in the other direction by using ToS-induction to interpret a $\Gamma : \mathsf{Con}_{\mathsf{syn}_j}$ in an arbitrary ToS model.

6.2 Useful Model Fragments of M

First, we need to describe three model fragments of **M**, which can be respectively used to compute notions of algebras, displayed algebras and sections for each syntactic signature. This is a rephrasing of the $-^{\mathsf{A}}$, $-^{\mathsf{D}}$ and $-^{\mathsf{S}}$ interpretations in [24].

Definition 6.4 (The Set model of ToS)**.** For each j, we have $\boldsymbol{A} : \mathsf{ToS}_{j+1,j}$, which can be given by restricting the **M** model of Section 4 so that we only have the first Con components in the interpretations for contexts, substitution, types, term, and we only have actions on contexts in the interpretations of term and substitution formers. Hence, we have:

$$
\begin{aligned}
\mathsf{Con}_{\boldsymbol{A}} &:\equiv \mathsf{Set}_{j+2}\\
\mathsf{Sub}_{\boldsymbol{A}}\,\Gamma\,\Delta &:\equiv \Gamma \to \Delta\\
\mathsf{Ty}_{\boldsymbol{A}}\,\Gamma &:\equiv \Gamma \to \mathsf{Set}_{j+2}\\
\mathsf{Tm}_{\boldsymbol{A}}\,\Gamma\,A &:\equiv (\gamma : \Gamma) \to A\,\gamma
\end{aligned}
$$

Now, for $\Gamma : \mathsf{Con}_{\mathsf{syn}}$, the type of Γ-algebras is given by $[\![\Gamma]\!]_{\boldsymbol{A}}$. E.g. $[\![\mathsf{NatSig}]\!]_{\boldsymbol{A}}$ yields a left-nested Σ-type of pointed sets with an endofunction. Also, $[\![\Gamma]\!]_{\mathbf{M}}$ extends $[\![\Gamma]\!]_{\boldsymbol{A}}$ to an flCwF of Γ-algebras.

Definition 6.5 (Logical predicate model of ToS over the Set model)**.** For each j level we have **D**, which is a displayed

ToS model over $\boldsymbol{A}$ at level j. This model, analogously to $\boldsymbol{A}$, is given by restricting $\mathbf{M}$ to the *third* components everywhere, corresponding to types or actions on types. Hence, we have:

$$\begin{aligned}
\mathsf{Con}_{\mathbf{D}}\,\underline{\Gamma} &:\equiv \underline{\Gamma} \to \mathsf{Set}_{j+2}\\
\mathsf{Sub}_{\mathbf{D}}\,\Gamma\,\Delta\,\underline{\sigma} &:\equiv (\underline{\gamma} : \underline{\Gamma}) \to \Gamma\,\gamma \to \Delta(\llbracket\underline{\sigma}\rrbracket_{\boldsymbol{A}}\,\underline{\gamma})\\
\mathsf{Ty}_{\mathbf{D}}\,\Gamma\,\underline{A} &:\equiv (\underline{\gamma} : \underline{\Gamma}) \to \Gamma\,\underline{\gamma} \to \llbracket\underline{A}\rrbracket_{\boldsymbol{A}}\,\underline{\gamma} \to \mathsf{Set}_{j+2}\\
\mathsf{Tm}_{\mathbf{D}}\,\Gamma\,A\,\underline{t} &:\equiv (\underline{\gamma} : \underline{\Gamma})(\gamma : \Gamma\,\underline{\Gamma}) \to A\,\underline{\gamma}\,\gamma \to (\llbracket\underline{t}\rrbracket_{\boldsymbol{A}}\,\underline{\gamma})
\end{aligned}$$

For $\Gamma : \mathsf{SynSig}_j$, the type of displayed Γ-algebras over some $\underline{\gamma} : \llbracket\Gamma\rrbracket_{\boldsymbol{A}}$ is given by $\llbracket\Gamma\rrbracket_{\mathbf{D}}\,\underline{\gamma}$. In other words, $\llbracket\Gamma\rrbracket_{\mathbf{D}}$ yields the notion of types in the flCwF of Γ-algebras given by $\llbracket\Gamma\rrbracket_{\mathbf{M}}$.

Definition 6.6 (Displayed algebra section model of ToS)**.** Analogously to $\boldsymbol{A}$ and $\mathbf{D}$, we define $\mathbf{S}$ as a displayed ToS model over the total model of $\mathbf{D}$, which is given by restricting $\mathbf{M}$ to the fourth components, corresponding to interpretations of terms and actions on terms.

For $\underline{\gamma} : \llbracket\Gamma\rrbracket_{\boldsymbol{A}}$ and $\gamma : \llbracket\Gamma\rrbracket_{\mathbf{D}}\,\underline{\gamma}$, the type of Γ-sections is computed as $\llbracket\Gamma\rrbracket_{\mathbf{S}}\,\underline{\gamma}\,\gamma$.

6.3 Term Algebras

The basic idea is that initial algebras can be built from the terms of syn_j. For example, consider the syntactic signature for natural numbers:

$$\mathsf{NatSig} :\equiv \bullet \triangleright (N : \mathsf{U}) \triangleright (\mathit{zero} : \mathsf{El}\,N) \triangleright (\mathit{suc} : N \Rightarrow \mathsf{El}\,N)$$

The type $\mathsf{Tm}_{\mathsf{syn}}\,\mathsf{NatSig}\,(\mathsf{El}_{\mathsf{syn}}\,N)$ is isomorphic to the usual type of natural numbers, since, intuitively, such terms can only be built from iterated usage of *zero* and *suc*. We will build a term algebra for each signature in this manner. First, we describe the interpretation of syn in the Set model.

Definition 6.7 (Term algebra construction)**.** For each syntactic signature $\underline{\Omega} : \mathsf{SynSig}_j$, we define a displayed ToS model over syn_j, named $\mathbf{T}_{\underline{\Omega}}$. The underlying sets are as follows:

$$\begin{aligned}
\mathsf{Con}_{\mathbf{T}_{\underline{\Omega}}}\,\underline{\Gamma} &:\equiv \mathsf{Sub}\,\underline{\Omega}\,\underline{\Gamma} \to \llbracket\underline{\Gamma}\rrbracket_{\boldsymbol{A}}\\
\mathsf{Sub}_{\mathbf{T}_{\underline{\Omega}}}\,\Gamma\,\Delta\,\underline{\sigma} &:\equiv (\underline{\nu} : \mathsf{Sub}\,\underline{\Omega}\,\underline{\Gamma}) \to \Delta\,(\underline{\sigma} \circ \nu) \simeq \llbracket\underline{\sigma}\rrbracket_{\boldsymbol{A}}\,(\Gamma\,\underline{\nu})\\
\mathsf{Ty}_{\mathbf{T}_{\underline{\Omega}}}\,\Gamma\,\underline{A} &:\equiv (\underline{\nu} : \mathsf{Sub}\,\underline{\Omega}\,\underline{\Gamma}) \to \mathsf{Tm}\,\underline{\Omega}\,(\underline{A}[\underline{\nu}]) \to \llbracket\underline{A}\rrbracket_{\boldsymbol{A}}\,(\Gamma\,\underline{\nu})\\
\mathsf{Tm}_{\mathbf{T}_{\underline{\Omega}}}\,\Gamma\,A\,\underline{t} &:\equiv (\underline{\nu} : \mathsf{Sub}\,\underline{\Omega}\,\underline{\Gamma}) \to \llbracket\underline{A}\rrbracket_{\boldsymbol{A}}\,\underline{\nu}\,(\underline{t}[\underline{\nu}]) \simeq_{\mathsf{id}} \llbracket\underline{t}\rrbracket_{\boldsymbol{A}}(\Gamma\,\underline{\nu})
\end{aligned}$$

Above, the $\simeq$ in the definition of $\mathsf{Sub}_{\mathbf{T}_{\underline{\Omega}}}$ is a context isomorphism in $\llbracket\underline{\Delta}\rrbracket_{\mathbf{M}}$, which is the flCwF of $\underline{\Delta}$-algebras. The $\simeq_{\mathsf{id}}$ in $\mathsf{Ty}_{\mathbf{T}_{\underline{\Omega}}}$ is a vertical context isomorphism in the displayed flCwF given by $\llbracket\underline{A}\rrbracket_{\mathbf{M}}$.

So far, the underlying sets in $\mathbf{T}_{\underline{\Omega}}$ are similar to what was given in [24] in the construction of term algebras, but there is an important difference: in ibid. strict equalities are used instead of isomorphisms. In our case, isomorphisms are necessary once again because of infinitary functions types and our identity type; we shall see this shortly. The universe is interpreted as follows:

$$\begin{aligned}
&\mathsf{U}_{\mathbf{T}_{\underline{\Omega}}} : (\underline{\nu} : \mathsf{Sub}\,\underline{\Omega}\,\underline{\Gamma})(\underline{t} : \mathsf{Tm}\,\underline{\Omega}\,\mathsf{U}) \to \mathsf{Set}_{j+1}\\
&\mathsf{U}_{\mathbf{T}_{\underline{\Omega}}}\,\underline{\nu}\,\underline{t} :\equiv \mathsf{Tm}\,\underline{\Omega}\,(\mathsf{El}\,\underline{t})\\
&\mathsf{El}_{\mathbf{T}_{\underline{\Omega}}}\,a : (\underline{\nu} : \mathsf{Sub}\,\underline{\Omega}\,\underline{\Gamma})(\underline{t} : \mathsf{Tm}\,\underline{\Omega}\,(\mathsf{El}\,(\underline{a}[\underline{\nu}]))) \to \llbracket\underline{a}\rrbracket_{\boldsymbol{A}}\,(\Gamma\,\underline{\nu})\\
&\mathsf{El}_{\mathbf{T}_{\underline{\Omega}}}\,a\,\underline{\nu}\,\underline{t} :\equiv (a\,\underline{\nu})\,\underline{t}
\end{aligned}$$

Hence, a syntactic $\underline{t} : \mathsf{Tm}\,\underline{\Omega}\,\mathsf{U}$ is interpreted as a set of terms with type $\mathsf{El}\,\underline{t}$. In the interpretation of El, note that

$$a\,\underline{\nu} : \mathsf{U}_{\mathbf{T}_{\underline{\Omega}}}\,\underline{\nu}\,(\underline{a}[\underline{\nu}]) \simeq_{\mathsf{id}} \llbracket\underline{a}\rrbracket_{\boldsymbol{A}}\,(\Gamma\,\underline{\nu})$$

hence

$$a\,\underline{\nu} : \mathsf{Tm}\,\underline{\Omega}\,(\mathsf{El}\,(\underline{a}[\underline{\nu}])) \simeq_{\mathsf{id}} \llbracket\underline{a}\rrbracket_{\boldsymbol{A}}\,(\Gamma\,\underline{\nu})$$

The $\simeq_{\mathsf{id}}$ above is just an isomorphism of sets, since it lives in $\llbracket\mathsf{U}\rrbracket_{\mathbf{M}}$ which was given as the flCwF of sets in Section 4.11. This above isomorphism is a good summary of the construction: the interpretation of a $\underline{a} : \mathsf{Tm}\,\underline{\Omega}\,\mathsf{U}$ in the term algebra is isomorphic to a set of terms.

Inductive functions are interpreted by transport along such isomorphism:

$$\Pi_{\mathbf{T}_{\underline{\Omega}}}\,a\,B\,\underline{\nu}\,\underline{t} :\equiv \lambda\,\alpha.\,B\,(\underline{\nu},\,(a\,\underline{\nu})^{-1}\,\alpha)\,(\underline{t}\;@\;((a\,\underline{\nu})^{-1}\,\alpha))$$

For the infinitary function space, we need the following, where $\simeq_{\mathsf{id}}$ is again set isomorphism.

$$\begin{aligned}
\Pi^{\mathsf{inf}}_{\mathbf{T}_{\underline{\Omega}}}\,A\,b\,\underline{\nu} : \mathsf{Tm}\,\underline{\Omega}\,(\mathsf{El}\,(\Pi^{\mathsf{inf}}\,A\,(\lambda\,\alpha.\,(b\,\alpha)[\underline{\nu}])))\\
\simeq_{\mathsf{id}}\,((\alpha : A) \to \llbracket\alpha\rrbracket_{\boldsymbol{A}}\,(\Gamma\,\underline{\nu}))
\end{aligned}$$

This can be given using the natural isomorphism consisting of $\mathsf{app}^{\mathsf{inf}}$ and $\mathsf{lam}^{\mathsf{inf}}$. However, the sides are not strictly equal. For the identity type, we build the following isomorphism using equality reflection.

$$\begin{aligned}
\mathsf{Id}_{\mathbf{T}_{\underline{\Omega}}}\,a\,t\,u\,\underline{\nu} : \mathsf{Tm}\,\underline{\Omega}\,(\mathsf{El}\,(\mathsf{Id}\,(\underline{a}[\underline{\nu}])\,(\underline{t}[\underline{\nu}])\,(\underline{u}[\underline{\nu}])))\\
\simeq_{\mathsf{id}}\,(\llbracket\underline{t}\rrbracket_{\boldsymbol{A}}\,(\Gamma\,\underline{\nu}) = \llbracket\underline{u}\rrbracket_{\boldsymbol{A}}\,(\Gamma\,\underline{\nu}))
\end{aligned}$$

We omit the rest of the definition of $\mathbf{T}_{\underline{\Omega}}$. The interpretations of equations in the CwF and fl-structure parts become fairly technical, and we also need to utilize iso-cleaving to interpret type substitution and substitution laws. However, the basic shape of the model remains similar to [24].

Now, we can build the term algebra for $\underline{\Omega}$ by taking $\llbracket\underline{\Omega}\rrbracket_{\mathbf{T}_{\underline{\Omega}}}\,\mathsf{id}$, which has type $\llbracket\underline{\Omega}\rrbracket_{\boldsymbol{A}}$.

Remark. If we start with a syntactic signature at level j, then the underlying sets in the term algebra are all in Set_{j+1}. Hence, the term algebra for $\mathsf{NatSig} : \mathsf{SynSig}_0$ has an underlying set in Set_{j+1}. This is a bit inconvenient, since normally we would have natural number in Set_0. Our current term model construction cannot avoid this level bump, since syn_j is necessarily a large type, and we do not have a way to construct a small set from a large set of terms. Perhaps this would be possible with a *resizing rule* [32]. Also, if we only consider closed finitary QIITs, with no possibility of referring to external types in signatures, then we can modify the current term model construction so that we always build

sets in Set_0. This would cover natural numbers and most dependent type theories.

6.4 Term Algebras Support Induction

7 Related Work

Cartmell [12]. Displayed categories Ahrens [1].

Paolo paper [2]. Finitary case [24].

Direct construction of concrete QIIT: Streicher [29], Brunerie [10], Ambroise [25].

Higher inductive types: HIT descriptions [6, 23], HITs in cubical type theories[13, 16], semantics for subclasses of HITs [5, 19, 26, 28, 31] Construction of example IITs without UIP: Jasper [20].

8 Conclusion

References

[1] Benedikt Ahrens and Peter LeFanu Lumsdaine. 2019. Displayed Categories. *Logical Methods in Computer Science* Volume 15, Issue 1 (March 2019). https://doi.org/10.23638/LMCS-15(1:20)2019

[2] Thorsten Altenkirch, Paolo Capriotti, Gabe Dijkstra, Nicolai Kraus, and Fredrik Nordvall Forsberg. 2018. Quotient Inductive-Inductive Types. In *Foundations of Software Science and Computation Structures*, Christel Baier and Ugo Dal Lago (Eds.). Springer International Publishing, Cham, 293–310.

[3] Carlo Angiuli, Robert Harper, and Todd Wilson. 2016. Computational higher type theory I: Abstract cubical realizability. *arXiv preprint arXiv:1604.08873* (2016).

[4] Carlo Angiuli, Kuen-Bang Favonia Hou, and Robert Harper. 2018. Cartesian Cubical Computational Type Theory: Constructive Reasoning with Paths and Equalities. *Computer Science Logic 2018* (2018).

[5] Steve Awodey, Jonas Frey, and Sam Speight. 2018. Impredicative Encodings of (Higher) Inductive Types. In *Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science (LICS '18)*. Association for Computing Machinery, New York, NY, USA, 76–85. https://doi.org/10.1145/3209108.3209130

[6] Henning Basold, Herman Geuvers, and Niels van der Weide. 2017. Higher Inductive Types in Programming. *Journal of Universal Computer Science* 23, 1 (jan 2017), 63–88.

[7] Jean-Philippe Bernardy, Patrik Jansson, and Ross Paterson. 2012. Proofs for Free — Parametricity for Dependent Types. *Journal of Functional Programming* 22, 02 (2012), 107–152. https://doi.org/10.1017/S0956796812000056

[8] Marc Bezem, Thierry Coquand, and Simon Huber. 2014. A model of type theory in cubical sets. In *19th International Conference on Types for Proofs and Programs (TYPES 2013)*, Vol. 26. 107–128.

[9] Lars Birkedal, Ranald Clouston, Bassel Mannaa, Rasmus Ejlers Møgelberg, Andrew M Pitts, and Bas Spitters. 2018. Modal dependent type theory and dependent right adjoints. *arXiv preprint arXiv:1804.05236* (2018).

[10] Guillaume Brunerie. 2019. A formalization of the initiality conjecture in Agda. (August 2019). https://guillaumebrunerie.github.io/pdf/initiality.pdf Slides of a talk at the Homotopy Type Theory 2019 Conference, Carnegie Mellon University, Pittsburgh, Pennsylvania.

[11] Jacques Carette, Oleg Kiselyov, and Chung-Chieh Shan. 2007. Finally tagless, partially evaluated. In *Asian Symposium on Programming Languages and Systems*. Springer, 222–238.

[12] John Cartmell. 1986. Generalised algebraic theories and contextual categories. *Annals of Pure and Applied Logic* 32 (1986), 209–243.

[13] Evan Cavallo and Robert Harper. 2019. Higher Inductive Types in Cubical Computational Type Theory. *Proc. ACM Program. Lang.* 3, POPL, Article Article 1 (Jan. 2019), 27 pages. https://doi.org/10.1145/3290314

[14] Pierre Clairambault and Peter Dybjer. 2014. The biequivalence of locally cartesian closed categories and Martin-Löf type theories. *Mathematical Structures in Computer Science* 24, 6 (2014).

[15] Cyril Cohen, Thierry Coquand, Simon Huber, and Anders Mörtberg. 2016. Cubical type theory: a constructive interpretation of the univalence axiom. *arXiv preprint arXiv:1611.02108* (2016).

[16] Thierry Coquand, Simon Huber, and Anders Mörtberg. 2018. On Higher Inductive Types in Cubical Type Theory. *LICS '18: Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science* (2018).

[17] Gabe Dijkstra. 2017. *Quotient inductive-inductive definitions*. Ph.D. Dissertation. University of Nottingham.

[18] Peter Dybjer. 1996. Internal Type Theory. In *Lecture Notes in Computer Science*. Springer, 120–134.

[19] Peter Dybjer and Hugo Moeneclaey. 2018. Finitary Higher Inductive Types in the Groupoid Model. *Electronic Notes in Theoretical Computer Science* 336 (2018), 119 – 134. https://doi.org/10.1016/j.entcs.2018.03.019 The Thirty-third Conference on the Mathematical Foundations of Programming Semantics (MFPS XXXIII).

[20] Jasper Hugunin. 2019. Constructing Inductive-Inductive Types in Cubical Type Theory. In *Foundations of Software Science and Computation Structures*, Mikołaj Bojańczyk and Alex Simpson (Eds.). Springer International Publishing, Cham, 295–312.

[21] Bart Jacobs. 1993. Comprehension categories and the semantics of type dependency. *Theoretical Computer Science* 107, 2 (1993), 169–207.

[22] Peter T Johnstone. 2002. *Sketches of an elephant: A topos theory compendium*. Vol. 2. Oxford University Press.

[23] Ambrus Kaposi and András Kovács. 2018. A Syntax for Higher Inductive-Inductive Types. In *3rd International Conference on Formal Structures for Computation and Deduction (FSCD 2018) (Leibniz International Proceedings in Informatics (LIPIcs))*, Hélène Kirchner (Ed.), Vol. 108. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany, 20:1–20:18. https://doi.org/10.4230/LIPIcs.FSCD.2018.20

[24] Ambrus Kaposi, András Kovács, and Thorsten Altenkirch. 2019. Constructing quotient inductive-inductive types. *Proceedings of the ACM on Programming Languages* 3, POPL (2019), 2.

[25] Ambrus Kaposi, András Kovács, and Lafont Ambroise. 2019. For Induction-Induction, Induction is Enough. *Submitted to TYPES 2019 post-proceedings* (2019).

[26] PETER LEFANU LUMSDAINE and MICHAEL SHULMAN. [n. d.]. Semantics of higher inductive types. *Mathematical Proceedings of the Cambridge Philosophical Society* ([n. d.]), 1–50. https://doi.org/10.1017/S030500411900015X

[27] The Univalent Foundations Program. 2013. *Homotopy type theory: Univalent foundations of mathematics*. Technical Report. Institute for Advanced Study.

[28] Kristina Sojakova. 2015. Higher Inductive Types As Homotopy-Initial Algebras. In *Proceedings of the 42Nd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL '15)*. ACM, New York, NY, USA, 31–42. https://doi.org/10.1145/2676726.2676983

[29] Thomas Streicher. 2012. *Semantics of type theory: correctness, completeness and independence results*. Springer Science & Business Media.

[30] Amin Timany and Matthieu Sozeau. 2018. Cumulative inductive types in Coq. *LIPIcs: Leibniz International Proceedings in Informatics* (2018).

[31] Niels van der Weide. 2016. *Higher Inductive Types*. Master's thesis. Radboud University, Nijmegen.

[32] Vladimir Voevodsky. 2011. Resizing rules, slides from a talk at TYPES2011. *At author's webpage* (2011).