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, quotient inductive-inductive types

1 Introduction

P and Q in cwf, level lifts, think about revising cETT or compressing

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 good example. 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 [**?**] and comprehension categories [**?**] to categories with families [**?**], but all of these are categories with extra structure.

The main motivation of the current paper is to generalize previously formalized [4] QIITs so that they accommodate more algebraic formulations of type theories, and more aspects of their practical metatheory. As a side effect of fulfilling this goal, infinitary QIITs such as Cauchy real numbers [**?**] are covered as well.

1.1 Contributions

We add the following features to QIITs:

1. **Large constructors, large elimination** and algebras at different universe levels. Large elimination is routinely used in the metatheory of type theory, but it has not been presented explicitly in previous works about QIITs.
2. **Infinitary constructors**. This allows infinitely branching trees. Also, 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 [4], where only finitary QIITs were described. We exploit 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 [3] 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

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

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. In a nutshell, the new semantics must be given mutually with a form of a "structure identity principle"[**?**], which says that signature extension respects algebra isomorphisms.

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 **??**, we describe a cumulative variant of extensional type theory which is used as metatheory in the rest of the paper. In Section 3, we introduce the theory of QIIT signatures. TODO

2 Metatheory

In this paper we consider QII algebras at arbitrary finite levels, along with large eliminations, where the initial algebra can be at a different level than the target algebra. For a simple example, consider natural number algebras at level i, given as the Σ-type $NatAlg_i :\equiv (Nat : \mathrm{Set}_i) \times Nat \times (Nat \to Nat)$. The initial such algebra is the set of natural numbers, which is at level 0, but in type theory we often want to eliminate into larger Set_i, for example when computing a Nat-indexed family of types. To support convenient reasoning about levels in this paper, we need two features: cumulativity and a way to quantify over finite levels. The former is essential to reduce bureaucratic overhead, while the latter is required to talk about arbitrary levels.

2.1 Cumulativity

In its most basic form, cumulativity requires that whenever $A : \mathrm{Set}_i$, and $i < j$ then also $A : \mathrm{Set}_j$. We also need cumulativity for Σ and function types, in order to have cumulativity for QII algebras in general. For example, we want to have $\gamma : NatAlg_j$ whenever $\gamma : NatAlg_i$ and $i < j$.

We use Sterling's cumulative algebraic type theory [5] as basis for our metatheory.

REASONS

We extend the base theory with an extensional identity type, Σ-types and the unit type. In [5] a proof of canonicity is provided for the base theory, which also includes a standard set-theoretic model. It is straightforward to extend the canonicity proof to cover our additional type formers.

From now on, we refer to this theory as cETT (cumulative extensional type theory). When working in cETT, we use the following notation. We have Russell-style 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 also use subscripts as a field projection notation for iterated pairs. For example, for $t : (A : \mathrm{Set}_i) \times (B : \mathrm{Set}_i) \times (f : A \to B)$, we use B_t to denote the projection of the second component. Sometimes we omit the subscript if it is clear from context.

$$\textsc{universe formation}\quad \frac{i < j}{\Gamma \vdash \mathrm{Set}_i : \mathrm{Set}_j} \qquad \textsc{function formation}\quad \frac{\Gamma \vdash A : \mathrm{U}_i \qquad \Gamma, x : A \vdash B : \mathrm{U}_j}{\Gamma \vdash (x : A) \to B : \mathrm{U}_{\max(i,j)}}$$

$$\Sigma\ \textsc{formation}\quad \frac{\Gamma \vdash A : \mathrm{U}_i \qquad \Gamma, x : A \vdash B : \mathrm{U}_j}{\Gamma \vdash (x : A) \times B : \mathrm{U}_{\max(i,j)}} \qquad \textsc{unit formation}\quad \frac{}{\Gamma \vdash \top : \mathrm{Set}_i}$$

$$\textsc{equality formation}\quad \frac{\Gamma \vdash A : \mathrm{Set}_i \qquad \Gamma \vdash t : A \qquad \Gamma \vdash u : A}{\Gamma \vdash t = u : \mathrm{Set}_i}$$

$$\textsc{type lifting}\quad \frac{\Gamma \vdash A : \mathrm{Set}_i \qquad i \le j}{\Gamma \vdash \Uparrow_j A : \mathrm{Set}_j} \qquad \textsc{lift composition}\quad \Uparrow_j (\Uparrow_i A) \equiv \Uparrow_j A$$

$$\textsc{identity lift}\quad \frac{\Gamma \vdash A : \mathrm{Set}_i}{(\Uparrow_i A) \equiv A} \qquad \textsc{term lifting}\quad \{t \mid \Gamma \vdash t : A\} \equiv \{t \mid \Gamma \vdash t : \Uparrow_i A\}$$

$$\textsc{context lifting}\quad (\Gamma, x : A) \equiv (\Gamma, x : \Uparrow_i A) \qquad \textsc{universe lifting}\quad \Uparrow_j \mathrm{Set}_i \equiv \mathrm{Set}_i$$

$$\textsc{function lifting}\quad \Uparrow_i ((x : A) \to B) \equiv (x : \Uparrow_i A) \to \Uparrow_i B$$

$$\textsc{pair lifting}\quad \Uparrow_i ((x : A) \times B) \equiv (x : \Uparrow_i A) \times \Uparrow_i B \qquad \textsc{unit lifting}\quad \Uparrow_i \top \equiv \top$$

$$\textsc{equality lifting}\quad \Uparrow_i (t = u) \equiv (t = u)$$

Figure 1. Some of the rules for lifting.

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

We also have a lifting operation on types, which introduces cumulativity. In Figure 1, we include an excerpt of cETT's rules. We change presentation slightly from Sterling: we use a $\mathrm{U}_{\max(i,j)}$ return type in functions and pairs, instead of having both A and B types in the same Set_i, and we define type lifting with $i \le j$ instead of $i < j$. These changed rules are all derivable from the original ones. In general, we can expect that lifting never impedes constructions, because it appropriately computes out of the way.

Of special note is the *term lifting rule*. It is a sort equation, an equation between sets of terms, expressing that lifted types have exactly the same terms as unlifted ones. This allows us to have $t : \Uparrow_j A$ whenever $t : A$. Together with the

universe lifting rule, this implies $A : \mathrm{Set}_{i+j}$ whenever $A : \mathrm{Set}_i$. Similarly, the lifting rules for functions and pairs give us cumulativity for *NatAlg*. We derive a notion of subtyping:

Definition 2.1 (Cumulative subtyping)**.** For $A : \mathrm{Set}_i$, $i \leq j$ and $B : \mathrm{Set}_j$, we define $A \leq B : \mathrm{Set}_j$ as $B = \Uparrow_j A$. It follows from term lifting and equality reflection that whenever $A \leq B$ and $t : A$, then also $t : B$.

2.2 Universe Polymorphism

For the sake of simplicity, we do not add this to cETT directly. Instead, we quantify over levels in an unspecified metatheory *outside* cETT. Hence, a universe polymorphic cETT term is understood as a $\mathbb{N}$-indexed family of terms. We reuse the notation of cETT functions for universe polymorphism, e.g. as in the following function:

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

In this paper, we do not need to internalize level-polymorphic constructions, so this setup is sufficient.

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. Π

Definition 3.1 (Notion of model for the theory of signatures)**.** For levels i and j, $\mathsf{ToS}_{i,j} : \mathrm{Set}_{\max(i,\,j)+1}$ 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_j. Following notation in [4], we denote these respectively as $\mathsf{Con} : \mathrm{Set}_j$, $\mathsf{Sub} : \mathsf{Con} \to \mathsf{Con} \to \mathrm{Set}_j$, $\mathsf{Ty} : \mathsf{Con} \to \mathrm{Set}_j$ and $\mathsf{Tm} : (\Gamma : \mathsf{Con}) \to \mathsf{Ty}\,\Gamma \to \mathrm{Set}_j$. 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, with $\Pi : (a : \mathsf{Tm}\,\Gamma\,\mathsf{U}) \to \mathsf{Ty}\,(\Gamma \triangleright \mathsf{El}\,a) \to \mathsf{Ty}\,\Gamma$ and 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 with Set_i domain: $\hat{\Pi} : (A : \mathrm{Set}_i) \to (A \to \mathsf{Ty}\,\Gamma) \to \mathsf{Ty}\,\Gamma$, with $\hat{\mathsf{app}} : \mathsf{Tm}\,\Gamma\,(\hat{\Pi}\,A\,B) \to ((x : A) \to \mathsf{Tm}\,\Gamma\,(B\,x))$ and its inverse $\hat{\mathsf{lam}}$.

Infinitary function space, with $\tilde{\Pi} : (A : \mathrm{Set}_i) \to (A \to \mathsf{Tm}\,\Gamma\,\mathsf{U}) \to \mathsf{Tm}\,\Gamma\,\mathsf{U}$, $\tilde{\mathsf{app}} : \mathsf{Tm}\,\Gamma\,(\mathsf{El}\,(\tilde{\Pi}\,A\,b)) \to ((x : A) \to \mathsf{Tm}\,\Gamma\,(\mathsf{El}\,(b\,x)))$ and $\tilde{\mathsf{lam}}$.

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 $\mathsf{Refl} : (T : \mathsf{Tm}\,\Gamma\,(\mathsf{El}\,a)) \to \mathsf{Id}\,a\,t\,t$, 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:

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

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

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

Here, i refers to the level of external types appearing in the signature, in the domains of $\hat{\Pi}$ and $\tilde{\Pi}$ functions, while the quantified j level is required to allow interpreting a signature in arbitrary-sized ToS models. Note that Sig_i 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.

$$\begin{aligned}
&\mathit{NatSig} : \mathsf{Sig}_0\\
&\mathit{NatSig} :\equiv \lambda(j : \mathbb{N})(M : \mathsf{ToS}_{0,j}).\\
&\quad (\bullet_M \triangleright_M (N : \mathsf{U}_M) \triangleright_M (\mathit{zero} : \mathsf{El}_M\,N)\\
&\qquad \triangleright_M (\mathit{suc} : \Pi_M\,N\,(\mathsf{El}_M\,N)))
\end{aligned}$$

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_i 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"[2], and it is used for defining and interpreting embedded languages.

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

$$\mathit{TreeSig} :\equiv \bullet \triangleright (\mathit{Tree} : \mathsf{U}) \triangleright (\mathit{node} : \hat{\Pi}\,\mathrm{Set}_0\,\lambda A.\,\Pi\,(\tilde{\Pi}\,A\,\lambda\_.\,\mathit{Tree})\,(\mathsf{El}\,\mathit{Tree}))$$

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

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

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

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 [4], 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.[1]. 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 to model notions of induction. The reason here is that CwFs allow us to define induction in strictly the same way as one would write it 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 isomorphisms. 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, we fix i and j levels, and define a model $M : \mathsf{ToS}_{i,1+\max(i,j+1)}$, such than Con_M is a type of structured categories (of algebras). The level i marks the level of all external sets in function domains, and j marks the level of all internal sorts in an algebra. Hence, every algebra has level $\max(i, j+1)$. The bump is only needed for j, since algebras only contain elements of $T : \mathsf{Set}_j$ types, while inductive sets are themselves elements of Set_i. For example, $NatAlg_i : \mathsf{Set}_{\max(0,i+1)}$.

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

4.2 Contexts

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

Definition 4.1 (Finite limit CwFs)**.** For a level i, there is a cETT type $\mathsf{flCwF}_i : \mathsf{Set}_{i+1}$, which is an iterated Σ-type with the following components:

1. A CwF with underlying sets all in Set_i. We reuse the notation from definition 3.1.
2. Σ-type $\Sigma : (A : \mathsf{Ty}\,\Gamma) \to \mathsf{Ty}\,(\Gamma \triangleright A) \to \mathsf{Ty}\,\Gamma$, with term formers proj1, proj2 and $-,-$.
3. Identity type $\mathsf{Id} : (A : \mathsf{Ty}\,\Gamma) \to \mathsf{Tm}\,\Gamma\,A \to \mathsf{Tm}\,\Gamma\,A \to \mathsf{Ty}\,\Gamma$, with equality reflection and UIP.
4. Constant families. This includes a type former $\mathsf{K} : \mathsf{Con} \to \mathsf{Ty}\,\Gamma$, where Γ is implicitly quantified, and an isomorphism between $\mathsf{Sub}\,\Gamma\,\Delta$ and $\mathsf{Tm}\,\Gamma\,(\mathsf{K}\,\Delta)$ which is natural in Γ. We have $\mathsf{lam}^{\mathsf{K}} : \mathsf{Sub}\,\Gamma\,\Delta \to \mathsf{Tm}\,\Gamma\,(\mathsf{K}\,\Delta)$ and $\mathsf{app}^{\mathsf{K}} : \mathsf{Tm}\,\Gamma\,(\mathsf{K}\,\Delta) \to \mathsf{Sub}\,\Gamma\,\Delta$ as its components. 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 [3].

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 $\Gamma : \mathsf{flCwF}_i$, we have the following predicate on contexts:

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

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)**.** *An object* $\Gamma : \mathsf{Con}_\Gamma$ *supports induction if and only if it is initial. Moreover, induction and initiality are both proof irrelevant predicates.*

Proof. By brief internal flCwF reasoning in [4]. □

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 (or "lex") categories [3]. It can be also seen that the categorical product of Γ and Δ in an flCwF can be given as $\Gamma \triangleright \mathsf{K}\,\Delta$, and equalizers can be given using the identity type.

segue text?

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

Definition 4.6 (Type categories, c.f. [3])**.** For each $\Gamma : \mathsf{Con}$, there is a category whose objects are types $A : \mathsf{Ty}\,\Gamma$, and morphisms from A to B are terms $t : \mathsf{Tm}\,(\Gamma \triangleright A)\,B[\mathsf{p}]$. Identity morphisms are given by $\mathsf{q} : \mathsf{Tm}\,(\Gamma \triangleright A)\,A[\mathsf{p}]$, and composition $t \circ u$ by $t[\mathsf{p}, u]$. The assignment of type categories to contexts extends to an indexed category. For each $\sigma : \mathsf{Sub}\,\Gamma\,\Delta$, there is a functor from $\mathsf{Ty}\,\Delta$ to $\mathsf{Ty}\,\Gamma$, which sends A to $A[\sigma]$ and $t : \mathsf{Tm}\,(\Gamma \triangleright A)\,B[\mathsf{p}]$ 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

we have a level lift here, decide how to handle

We define $\mathbf{Sub}\,\mathbf{\Gamma}\,\mathbf{\Delta} : \mathsf{Set}_{1+\max(i,j+1)}$ as the type of weak flCwF morphisms from $\mathbf{\Gamma}$ to $\mathbf{\Delta}$.

Definition 4.8. A weak flCwF morphism $\boldsymbol{\sigma} : \mathbf{Sub}\,\mathbf{\Gamma}\,\mathbf{\Delta}$ 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 $\epsilon : \mathsf{Sub}\,(\boldsymbol{\sigma}\,\bullet)\,\bullet$ has a retraction.
4. Each $(\boldsymbol{\sigma}\,\mathsf{p}, \boldsymbol{\sigma}\,\mathsf{q}) : \mathsf{Sub}\,(\boldsymbol{\sigma}\,(\Gamma \triangleright A))\,(\boldsymbol{\sigma}\,\Gamma \triangleright \boldsymbol{\sigma}\,A)$ 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 $\boldsymbol{\sigma}_\triangleright : \boldsymbol{\sigma}\,(\Gamma \triangleright A) \simeq \boldsymbol{\sigma}\,\Gamma \triangleright \boldsymbol{\sigma}\,A$.

Theorem 4.9. *Every $\boldsymbol{\sigma} : \mathbf{Sub}\,\mathbf{\Gamma}\,\mathbf{\Delta}$ preserves fl-structure up to type isomorphism. That is, we have isomorphisms*

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

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

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 $\epsilon : \mathsf{Sub}\,\Gamma\,\bullet$ 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 that $\boldsymbol{\sigma}$ preserves term and substitution formers. □

4.4 Identity and Composition

References

[1] 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.

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

[3] 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).

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

[5] Jonathan Sterling. 2019. Algebraic type theory and universe hierarchies. *arXiv preprint arXiv:1902.08848* (2019).