

# Generalizations of Quotient Inductive-Inductive Types<sup>\*</sup>

Ambrus Kaposi and András Kovács

Eötvös Loránd University, Budapest, Hungary  
{akaposi|kovacsandras}@inf.elte.hu

**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, infinitary constructors and equations between sorts. 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. We use self-description to bootstrap a model theory for the theory of signatures without using preterms or assuming a pre-existing internal syntax for a type theory. We give initial algebra semantics for described QIITs, and we show the equivalence of initiality and induction. We present two extensions of a previous term model construction. The first one constructs all large infinitary QIITs without sort equations from the QIIT of the theory of signatures. The second one constructs all large finitary QIITs with sort equations, from the same syntax. This separation is required because handling infinitary constructors in the term model requires showing a strong form of invariance under algebra isomorphisms, which is violated by sort equations.

## 1 Introduction

### Citations

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

---

<sup>\*</sup> This work was supported by EFOP-3.6.3-VEKOP-16-2017-00002 grant and COST Action EUTypes CA15123.

both features in order to write down the signature of categories:

$$\begin{aligned}
Obj & : \text{Set} \\
Mor & : Obj \rightarrow Obj \rightarrow \text{Set} \\
id & : Mor\ i\ j \\
- \circ - & : Mor\ j\ k \rightarrow Mor\ i\ j \rightarrow Mor\ i\ k \\
idl & : id \circ f = f \\
idr & : f \circ id = f \\
ass & : f \circ (g \circ h) = (f \circ g) \circ h
\end{aligned}$$

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. A number of different notions 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 extend QIITs so that it accommodates all algebraic notions of type theories which have been used in previous works. As a beneficial side effect of fulfilling this goal, infinitary QIITs such as Cauchy real numbers will be covered as well.

We generalize previous notions of QIITs in the following ways:

1. **Large constructors, large elimination** and models at different universe levels. This feature is routinely used in the metatheory of type theory, but it has not been presented explicitly in previous works about QIITs. In order to handle universe levels gracefully, we work in an extensional type theory extended with cumulative universes.
2. **Infinitary constructors.** This enables defining Cauchy real numbers and surreal numbers as QIITs. 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. We use this 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.
4. **Sort equations**, or equations between type constructors. They have been used recently to conveniently represent type-theoretic universes, for example Russell-style universes or cumulative hierarchies.

We also develop semantics. We show that for each signature, there is a cwf (category with families) of algebras, extended with  $\Sigma$ -types, extensional identity, and constant families. This additional structure corresponds to a type-theoretic flavor of finite limits, and it was shown in [?] that the category of such cwfs is biequivalent to the category of finitely complete categories.

As to the existence of initial algebras, we present two different term model constructions, yielding the following results:

1. All large infinitary QIITs with recursive equations, but *without sort equations*, are reducible to the theory of signatures, i.e. their initial algebras are constructible from the initial algebra for the theory of signatures.
2. All large finitary QIITs with sort equations, but no recursive equations, are reducible to the theory of signatures.

The reason for the two separate constructions is that sort equations are not as well-behaved as the other extensions: they are modelled as strict equalities of sets in algebras, hence they do not respect isomorphism of sets. We make essential use of invariance under set isomorphism in the first term model construction, so we cannot throw sort equations into that mix.

In Section 2 we set up cumulative universes which we use in later sections. In Section 3 we introduce the theory of QIIT signatures. In Section 4 we develop initial algebra semantics, also covering the theory of signatures itself. In Sections 5-6 we present the two term model constructions. We discuss related work and conclusions in Sections 7-8.

## 2 Cumulative Extensional Type Theory

In the following sections, we will consider QIIT 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 a given level  $i$ , given as the  $\Sigma$ -type  $NatAlg_i := (Nat : Set_i) \times Nat \times (Nat \rightarrow Nat)$ . 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.

Universe levels are usually viewed as a bureaucratic and tedious part of type theory, and are often omitted or mentioned only in passing. Here we intend to model universe levels in a precise way, because large eliminations are an essential part of the practical usage of inductive types. Hence, we need a way to conveniently reason about levels without too much administrative burden. Unfortunately, simple universe setups are insufficient for this purpose. We need two features: cumulativity and a way to quantify over finite levels.

### 2.1 Cumulativity

Cumulativity makes handling of algebras and constructions at different levels much easier. For example, given an algebra  $\gamma : NatAlg_i$ , cumulativity allows us

to also have  $\gamma : \text{NatAlg}_{i+j}$ . In a non-cumulative setting such as Agda, we may only have for each type  $A : \text{Set}_i$  a type  $\text{Lift}_{i,j} A : \text{Set}_{i+j}$  which is isomorphic to  $A$  up to definitional equality.

In the most basic form, cumulativity requires that whenever  $A : \text{Set}_i$ , then also  $A : \text{Set}_{i+j+1}$ . However, the *NatAlg* example above also required a form of cumulativity for  $\Sigma$  and  $\Pi$  types, which we also need in general for QIIT algebras.

We use Sterling’s cumulative algebraic type theory [?] as the general setting for the rest of the paper. The reason for this is twofold. First, it supports the right kind of cumulativity for our purposes. Second, it is itself a finitary QIIT with sort equations, hence this paper provides a model theory for it<sup>1</sup>.

We extend the base theory with an extensional identity type,  $\Sigma$ -types and the unity type. In [?] 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  $\text{Set}_i$  indexed by natural numbers, dependent functions as  $(x : A) \rightarrow B$ , and dependent pairs as  $(x : A) \times B$  with projections  $\text{proj1}$  and  $\text{proj2}$ . We also use a field projection notation for iterated pairs. For example, for  $t : (A : \text{Set}_i) \times (B : \text{Set}_i) \times (f : A \rightarrow B)$ , we use  $t.B$  to denote the projection of the second component.

We write propositional equality as  $t = u$ , with  $\text{refl}_t$  for reflexivity. We have equality reflection for  $- = -$ . The unit type is  $\top$ , with inhabitant  $\text{tt}$ .

We also have a lifting operation on types, which introduces cumulativity.

UNIVERSE FORMATION	TYPE LIFTING	UNIVERSE LIFTING
$\frac{}{\Gamma \vdash \text{Set}_i : \text{Set}_{i+j+1}}$	$\frac{\Gamma \vdash A : \text{Set}_i}{\Gamma \vdash \uparrow_j A : \text{Set}_{i+j+1}}$	$\uparrow_j \text{Set}_i = \text{Set}_{i+j}$
TERM LIFTING	CONTEXT LIFTING	
$\{t \mid \Gamma \vdash t : A\} = \{t \mid \Gamma \vdash t : \uparrow_j A\}$	$(\Gamma, x : A) = (\Gamma, x : \uparrow_j A)$	
FUNCTION LIFTING	PAIR LIFTING	
$\uparrow_j ((x : A) \rightarrow B) = (x : \uparrow_j A) \rightarrow \uparrow_j B$	$\uparrow_j ((x : A) \times B) = (x : \uparrow_j A) \times \uparrow_j B$	

**Fig. 1.** Some of the rules for lifting.

In Figure 1, we include an excerpt of Sterling’s rules, with a slight change of presentation. We lift from  $i$  to  $i + j + 1$  levels instead of assuming  $i < j$  and lifting to  $j$ . We also omit rules for naturality and substitution of lifting. In general, we can expect that lifting never impedes constructions, because it appropriately computes out of the way.

<sup>1</sup> Which is, of course, somewhat circular, but not more circular than e.g. the study of models of ZFC in ZFC.

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  $A : \text{Set}_{i+j+1}$  whenever  $A : \text{Set}_i$ . Similarly, the lifting rules for functions and pairs give us cumulativity for *NatAlg*. We derive a notion of subtyping:

**Definition 1.** Cumulative subtyping. For  $A : \text{Set}_i$  and  $B : \text{Set}_{i+j+1}$ , we define  $A \prec B : \text{Set}_{i+j+1}$  as  $B = \uparrow_j A$ . It follows from term lifting and equality reflection that whenever  $A \prec B$  and  $t : A$ , then also  $t : B$ .

## 2.2 Universe Polymorphism

We also need to quantify over universe levels. cETT does not support this, and we leave it like that. Instead, for the sake of simplicity, 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 use the  $\forall$  symbol for this, e.g. as  $\forall i. \text{Set}_i \rightarrow \text{Set}_i$ . This is sufficient, because we do not need to internalize level-polymorphic constructions in cETT.

## 3 Signatures

In this section, we define signatures for QIITs which support all the mentioned generalizations. Signatures are given as contexts in a certain type theory, called 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 2.** Notion of model for ToS. For levels  $i$  and  $j$ , there is a cETT type  $\text{ToS}_{i,j} : \text{Set}_{\max(i,j)+1}$ , whose elements are ToS models (or ToS-algebras).  $\text{ToS}_{i,j}$  is an iterated  $\Sigma$ -type, containing all of the following components.

1. A category with families (cwf), where all four underlying sets (of objects, morphisms, types and terms) are in  $\text{Set}_i$ . Following notation in [?], we denote these respectively as  $\text{Con} : \text{Set}_i$ ,  $\text{Sub} : \text{Con} \rightarrow \text{Con} \rightarrow \text{Set}_i$ ,  $\text{Ty} : \text{Con} \rightarrow \text{Set}_i$  and  $\text{Tm} : (\Gamma : \text{Con}) \rightarrow \text{Ty } \Gamma \rightarrow \text{Set}_i$ . We denote the empty context as  $\bullet : \text{Con}$ , context extension as  $- \triangleright - : (\Gamma : \text{Con}) \rightarrow \text{Ty } \Gamma \rightarrow \text{Con}$ . Substitution on types and terms is written as  $-[-]$ .
2. A universe  $\text{U} : \text{Ty } \Gamma$  with decoding  $\text{El} : (a : \text{Tm } \Gamma \text{ U}) \rightarrow \text{Ty } \Gamma$ .
3. Inductive function space, with  $\Pi : (a : \text{Tm } \Gamma \text{ U}) \rightarrow \text{Ty } (\Gamma \triangleright \text{El } a) \rightarrow \text{Ty } \Gamma$ , application as  $\text{app} : \text{Tm } \Gamma (\Pi a B) \rightarrow \text{Tm } (\Gamma \triangleright \text{El } a) B$ , and its inverse  $\text{lam}$ .
4. External function space with  $\text{Set}_j$  domain:  $\hat{\Pi} : (A : \text{Set}_j) \rightarrow (A \rightarrow \text{Ty } \Gamma) \rightarrow \text{Ty } \Gamma$ , with  $\text{app } \hat{\Pi} : \text{Tm } \Gamma (\hat{\Pi} A B) \rightarrow (x : A) \rightarrow \text{Tm } \Gamma (B x)$ , and likewise  $\text{lam } \hat{\Pi}$  as inverse.

5. Infinitary function space, with  $\tilde{\Pi} : (A : \text{Set}_j) \rightarrow (A \rightarrow \text{Tm } \Gamma \text{ U}) \rightarrow \text{Tm } \Gamma \text{ U}$ ,  $\text{app} : \text{Tm } \Gamma (\text{El } (\hat{\Pi} A b)) \rightarrow (x : A) \rightarrow \text{Tm } \Gamma (\text{El } (b x))$ , and  $\text{lam}$ .
6. An identity type  $\text{Id} : (a : \text{Tm } \Gamma \text{ U}) \rightarrow \text{Tm } \Gamma (\text{El } a) \rightarrow \text{Tm } \Gamma (\text{El } a) \rightarrow \text{Tm } \Gamma \text{ U}$ , with  $\text{Refl} : (T : \text{Tm } \Gamma (\text{El } a)) \rightarrow \text{Id } a \text{ T T}$  and equality reflection.
7. An identity type  $\text{IdU} : \text{Tm } \Gamma \text{ U} \rightarrow \text{Tm } \Gamma \text{ U} \rightarrow \text{Ty } \Gamma$ , with evident  $\text{ReflU}$  and equality reflection.

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

*Notational conventions.* Cwf components by default support de Bruijn indices, which are not easily readable. We use instead a nameful notation for binders in context extension,  $\Pi$  and  $\text{lam}$ , e.g. as  $(\bullet \triangleright a : U \triangleright t : \text{El } a)$ . We also define a type-theoretic version of  $\text{app}$  for convenience:

$$\begin{aligned} - @ - &: \text{Tm } \Gamma (\Pi a B) \rightarrow (u : \text{Tm } \Gamma (\text{El } a)) \rightarrow \text{Tm } \Gamma (B[id, u]) \\ t @ u &:= (\text{app } t)[id, u] \end{aligned}$$

**Definition 3.** Notion of signature. A QIIT signature at level  $j$  is a context in an arbitrary  $M : \text{ToS}_{i,j}$  model. We define the type of such signatures as follows:

$$\text{Sig}_j := \forall i. (M : \text{ToS}_{i,j}) \rightarrow M.\text{Con}$$

Note that  $\text{Sig}_j$  is defined as a universe-polymorphic type, so it does not have a type internally to cETT. A concrete signature  $\text{sig}$  at level  $j$  has type  $\forall i. (M : \text{ToS}_{i,j}) \rightarrow M.\text{Con}$ , so it builds up a context from the components of an arbitrary  $\text{ToS}$  model.

*Example 1.* The signature for natural numbers is the following, where  $\bullet$ ,  $- \triangleright -$ ,  $\text{U}$ ,  $\text{El}$  and  $\Pi$  all refer to components of  $M$ :

$$\begin{aligned} \text{NatSig} &: \text{Sig}_0 \\ \text{NatSig} &:= \lambda(M : \text{ToS}_{i,0}). (\bullet \triangleright N : \text{U} \triangleright \text{zero} : \text{El } N \triangleright \text{suc} : \Pi (n : N) (\text{El } N)) \end{aligned}$$

The  $j$  level in  $\text{Sig}_j$  specifies the level of the external types appearing in a signature, since the  $\hat{\Pi}$  and  $\tilde{\Pi}$  function spaces have domains in  $\text{Set}_j$ . In the case of natural numbers, no external types appear, so the level of  $\text{NatSig}$  can be chosen as 0.

With this, we are able to specify QIITs, and we can also interpret each signature in an arbitrary  $\text{ToS}$  model, just by applying a signature to a model.  $\text{Sig}_j$  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  $\text{ToS}$  components. In functional programming, this encoding is sometimes called “finally tagless” [?], and it is used for defining and interpreting embedded languages.

### 3.1 Example Signatures

Compared to signatures in the previous work [?], we additionally support  $\tilde{\Pi}$  and  $\text{IdU}$ , and  $\text{Id}$  has a more liberal type which allows recursive equations. Let us see

examples for these three in order. For brevity, we shall only write contexts, and assume that all ToS operations come from an arbitrary model.

*Example 2.* Infinitary constructors. The universe  $\mathbf{U}$  is closed under the  $\tilde{\Pi}$  function type, which makes it possible to write such functions in the domains of  $\Pi$  types. This allows a signature for  $W$ -types, given arbitrary  $S : \mathbf{Set}_i$  and  $P : S \rightarrow \mathbf{Set}_i$  types for shapes and positions.

$$WSig \equiv \bullet \triangleright W : \mathbf{U} \triangleright sup : \hat{\Pi} S (\lambda s. \Pi (\tilde{\Pi} (P s) \lambda_. W) (\mathbf{E} W))$$

*Example 3.* Sort equations. An interesting example is the term lifting rule for cETT (omitting everything except term lifting):

$$\begin{aligned} cETTSig &\equiv \\ \dots \triangleright Tm\uparrow : \Pi (\Gamma : Con) (\Pi (A : Ty @ \Gamma) ((\mathbf{Id} \mathbf{U} (Tm @ \Gamma @ A) (Tm @ \Gamma @ (\uparrow_j @ A)))))) \end{aligned}$$

*Example 4.* Recursive equations. Our definition of  $\mathbf{Id}$  returns in  $\mathbf{U}$ , which, similarly to the case of infinitary functions, allows us to write equations in  $\Pi$  domains. A minimal example:

$$RecEqSig \equiv \bullet \triangleright A : \mathbf{U} \triangleright a : \mathbf{E} A \triangleright f : \Pi (x : A) (\Pi (\mathbf{Id} A x a) (\mathbf{E} A))$$

There more complicated and interesting examples, e.g. boundary conditions in various cubical type theories[?]. Note that our  $\mathbf{Id}$  allows iterated equations as well, but these are all made trivial in the semantics, where we assume uniqueness of identity proofs.

### 3.2 A Signature for the Theory of Signatures

put this in infinitary term models beginning

Starting from Section 5, we need to do proofs and constructions by induction on signatures. However, the current signatures do not support induction, analogously to how induction is not available for Church encodings. We would like to have two things: a notion of signature induction, and a model of ToS which supports induction. Then, we can use contexts in the syntax of ToS (i.e. in the inductive model) as signatures. We provide a notion of induction on signatures by

1. Showing that there is a signature which describes ToS.
2. Building categorical semantics, which for each signature provides a notion of induction.

On the other hand, in this paper we will only *assume* the existence of a syntax for ToS starting from Section 5. We do not provide a construction of the syntax from simpler type formers, although we do show that lower-level ToS syntaxes are constructible from higher-level ones.

- 4 Categories with Families of Algebras
- 5 Infinitary Term Models
  - 5.1 Preservation of Isomorphisms
- 6 Term Models for Sort Equations
- 7 Related Work
- 8 Conclusion