Generalizations of Quotient Inductive-Inductive Types*

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

ISSUE. We actually need flCwF pseudomorphisms for Sub in infinitary semantics. $a : \mathsf{Tm}\,\Gamma\,\mathsf{U}$ is *not* a discrete displayed flCwF, only $\mathsf{El}\,a$ is! $\tilde{\Pi}\,A\,b : \mathsf{Tm}\,\Gamma,\mathsf{U}$ does not preserve (displayed) object in Γ strictly.

- New semantics with pseudomorphisms for Sub and Tm. Is it enough to have weak preservation of comprehension, and we get all other preservation for free, see "gluing for type theory"?
- Check term model and eliminators to see if change needed.
- Check if the isomorphism for K in flCwF can be made strict, i.e. $\mathsf{Sub}\,\Gamma\,\Delta = \mathsf{Tm}\,\Gamma\,(\mathsf{K}\,\Delta)$.

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

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:

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

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 the notion of QIIT so that it accommodates all algebraic notions of type theories which have been used in previous works. As a side effect of fulfilling this goal, infinitary QIITs such as Cauchy real numbers are 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.
2. **Infinitary constructors**. This allows infinitely branching trees. Also, the theory of QIIT signatures is itself large and infinitary, thus now it can "eat itself", i.e. include its own signature and provide its own metatheory. This was previously not possible in [**?**], where only finitary QIITs were described. 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 Σ-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 **??**-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 level i, given as the Σ-type $NatAlg_i :\equiv (Nat : \mathsf{Set}_i)\times Nat\times(Nat \to Nat)$. 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.

2.1 Cumulativity

We need cumulativity to reduce bureaucratic overhead. In its most basic form, cumulativity requires that whenever $A : \mathsf{Set}_i$, and $i < j$ then also $A : \mathsf{Set}_j$. We also need cumulativity for Σ and function types, in order to have cumulativity

for algebras in general. For example, we want to have $\gamma : NatAlg_j$ whenever $\gamma : NatAlg_i$ and $i < j$. We have found that non-cumulative algebras induce excessive "lifting" noise, which we would like to avoid.

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[1].

We extend the base theory with an extensional identity type, Σ-types and the unit 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 Set_i indexed by natural numbers, dependent functions as $(x : A) \to B$, and dependent pairs as $(x : A) \times B$ with projections $\mathsf{proj1}$ and $\mathsf{proj2}$. 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.

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.

UNIVERSE FORMATION
$$\frac{i < j}{\Gamma \vdash \mathsf{Set}_i : \mathsf{Set}_j}$$

FUNCTION FORMATION
$$\frac{\Gamma \vdash A : \mathsf{U}_i \qquad \Gamma, x : A \vdash B : \mathsf{U}_j}{\Gamma \vdash (x : A) \to B : \mathsf{U}_{\max(i,j)}}$$

Σ FORMATION
$$\frac{\Gamma \vdash A : \mathsf{U}_i \qquad \Gamma, x : A \vdash B : \mathsf{U}_j}{\Gamma \vdash (x : A) \times B : \mathsf{U}_{\max(i,j)}}$$

UNIT FORMATION
$$\frac{}{\Gamma \vdash \top : \mathsf{Set}_i}$$

TYPE LIFTING
$$\frac{\Gamma \vdash A : \mathsf{Set}_i \qquad i < j}{\Gamma \vdash \Uparrow_j A : \mathsf{Set}_j}$$

LIFT COMPOSITION
$$\Uparrow_j (\Uparrow_i A) = \Uparrow_j A$$

TERM LIFTING
$$\{t \mid \Gamma \vdash t : A\} = \{t \mid \Gamma \vdash t : \Uparrow_i A\}$$

CONTEXT LIFTING
$$(\Gamma, x : A) = (\Gamma, x : \Uparrow_i A)$$

UNIVERSE LIFTING
$$\Uparrow_j \mathsf{Set}_i = \mathsf{Set}_i$$

FUNCTION LIFTING
$$\Uparrow_i ((x : A) \to B) = (x : \Uparrow_i A) \to \Uparrow_i B$$

PAIR LIFTING
$$\Uparrow_i ((x : A) \times B) = (x : \Uparrow_i A) \times \Uparrow_i B$$

UNIT LIFTING
$$\Uparrow_i \top = \top$$

Fig. 1. Some of the rules for lifting.

[1] Which is, of course, somewhat circular, but not more circular than e.g. the study of models of ZFC in ZFC.

In Figure 1, we include an excerpt of cETT's rules. We change presentation slightly from Sterling: we use a $\mathsf{U}_{\max(i,j)}$ return type in functions and pairs, instead of having both A and B types in the same Set_i. 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 $A : Set_{i+j+1}$ whenever $A : Set_i$. Similarly, the lifting rules for functions and pairs give us cumulativity for $NatAlg$. We derive a notion of subtyping:

Definition 1 (Cumulative subtyping). For $A : \mathsf{Set}_i$ and $B : \mathsf{Set}_{i+j+1}$, we define $A \prec B : \mathsf{Set}_{i+j+1}$ 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 reuse the notation of cETT functions for level-polymorphism, e.g. as in the following function:

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

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

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 the theory of signatures). For levels i and j, there is a cETT type $\mathsf{ToS}_{i,j} : \mathsf{Set}_{\max(i,\,j)+1}$, whose elements are ToS models (or ToS-algebras). $\mathsf{ToS}_{i,j}$ is an iterated Σ-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 Set_j. Following notation in [**?**], we denote these respectively as $\mathsf{Con} : \mathsf{Set}_j$, $\mathsf{Sub} : \mathsf{Con} \to \mathsf{Con} \to \mathsf{Set}_j$, $\mathsf{Ty} : \mathsf{Con} \to \mathsf{Set}_j$

and $\mathsf{Tm} : (\Gamma : \mathsf{Con}) \to \mathsf{Ty}\,\Gamma \to \mathsf{Set}_j$. We denote the empty context as $\bullet : \mathsf{Con}$, context extension as $- \triangleright - : (\Gamma : \mathsf{Con}) \to \mathsf{Ty}\,\Gamma \to \mathsf{Con}$. Substitution on types and terms is written as $-[-]$, with id and $- \circ -$ for identity and composition, and $-,- : (\sigma : \mathsf{Sub}\,\Gamma\,\Delta) \to \mathsf{Tm}\,\Gamma\,A[\sigma] \to \mathsf{Sub}\,\Gamma\,(\Delta \triangleright A)$ for substitution extension.
2. A universe $\mathsf{U} : \mathsf{Ty}\,\Gamma$ with decoding $\mathsf{El} : (a : \mathsf{Tm}\,\Gamma\,\mathsf{U}) \to \mathsf{Ty}\,\Gamma$.
3. 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.
4. External function space with Set_i domain: $\hat{\Pi} : (A : \mathsf{Set}_i) \to (A \to \mathsf{Ty}\,\Gamma) \to \mathsf{Ty}\,\Gamma$, with $\mathsf{app}\hat{\Pi} : \mathsf{Tm}\,\Gamma\,(\hat{\Pi}\,A\,B) \to ((x : A) \to \mathsf{Tm}\,\Gamma\,(B\,x))$ and its inverse $\mathsf{lam}\hat{\Pi}$.
5. Infinitary function space, with $\tilde{\Pi} : (A : \mathsf{Set}_i) \to (A \to \mathsf{Tm}\,\Gamma\,\mathsf{U}) \to \mathsf{Tm}\,\Gamma\,\mathsf{U}$, $\mathsf{app}\tilde{\Pi} : \mathsf{Tm}\,\Gamma\,(\mathsf{El}\,(\tilde{\Pi}\,A\,b)) \to ((x : A) \to \mathsf{Tm}\,\Gamma\,(\mathsf{El}\,(b\,x)))$ and $\mathsf{lam}\tilde{\Pi}$.
6. 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.
7. An identity type $\mathsf{IdU} : \mathsf{Tm}\,\Gamma\,\mathsf{U} \to \mathsf{Tm}\,\Gamma\,\mathsf{U} \to \mathsf{Ty}\,\Gamma$, with reflexivity as ReflU, equality reflection and UIP.

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 $\mathsf{ToS}_{i,j}$.

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, Π and lam, e.g. as $(\bullet \triangleright a : U \triangleright t : \mathsf{El}\,a)$. We also define a type-theoretic version of app for convenience:

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

Definition 3 (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:

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

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 does not have a type internally to cETT.

Example 1. Signature for natural numbers. Here, no external types appear, so the level of $NatSig$ can be chosen as 0.

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

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 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 IdU, and 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 U is closed under the $\tilde{\Pi}$ function type, which makes it possible to write such functions 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.

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

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

$$\begin{aligned} &cETTSig :\equiv \\ &\ldots \triangleright Tm{\Uparrow} : \Pi\, (\Gamma : Con)\, (\Pi\, (A : Ty@\Gamma)((\mathsf{IdU}\, (Tm@\Gamma@A)\, (Tm@\Gamma@(\Uparrow_j @A)))) \end{aligned}$$

Example 4. Recursive equations. Our definition of Id returns in U, which, similarly to the case of infinitary functions, 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 made trivial in the semantics, where we assume UIP.

4 Categorical Semantics

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 [**?**], we do this by creating a model of ToS, where contexts are categories supporting the above requirements and substitutions are strictly structure-preserving functors. Then, each signature can applied to this model, yielding an interpretation of the signature as a structured category of algebras. First, we fix our notion of such structured category.

Definition 4 (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.
2. Σ-type $\mathsf{\Sigma} : (A : \mathsf{Ty}\,\Gamma) \to \mathsf{Ty}\,(\Gamma \triangleright A) \to \mathsf{Ty}\,\Gamma$, with usual projections and pairing.
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. The idea is that $\mathsf{K}\,\Delta$ is a representation of Δ as a type in any context. Dybjer and Clairambault called constant families "democracy" in [**?**].

We use Ty in an flCwF to represent "dependent objects", and Tm for "dependent morphisms". We use these to define induction.

Definition 5 (Notion of induction in an flCwF). Given $C : \mathsf{flCwF}_i$, we have the following predicate on objects:

$$\begin{aligned} &\mathit{Inductive} : \mathsf{Con}_C \to \mathsf{Set}_i \\ &\mathit{Inductive}\,\Gamma :\equiv (A : \mathsf{Ty}_C\,\Gamma) \to \mathsf{Tm}_C\,\Gamma\,A \end{aligned}$$

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

Proof. Shown by brief internal flCwF reasoning in [**?**]. □

The reason for the "finite limit CwF" naming is the following: Dybjer and Clairambault showed[**?**] that the 2-category of flCwFs is biequivalent to the 2-category of finitely complete (or "lex") categories. 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.

So, why do we use flCwFs instead of lex categories? After all, the former has more components and has more immediate complexity than the latter. The reason is that flCwFs 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 what we want 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.

Definition 6 (flCwF model for ToS). We have for each i and j levels an $M : \mathsf{ToS}_{i,1+\max(i,j+1)}$ such that $\mathsf{Con}_M = \mathsf{flCwF}_{\max(i,j+1)}$. This model is an extension of the CwF model of signatures in [**?**]. The new additions are: Σ-types in the semantic CwF, the new type formers in signatures, and the treatment of universe levels. We summarize the model below.

As noted, Con is given as $\mathsf{flCwF}_{\max(i,j+1)}$. 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, i.e. object of the flCwF, 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, $NatAlg_i : \mathsf{Set}_{\max(0,i+1)}$.

$\sigma : \mathsf{Sub}\,\Gamma\,\Delta$ substitutions are functors preserving all flCwF structure strictly.

$A : \mathsf{Ty}\,\Gamma$ types are displayed flCwFs over Γ. This is an flCwF where every underlying set is indexed over underlying sets in Γ, and it is an extension of the displayed categories of Ahrens and Lumsdaine[**?**]. It contains equivalent data to an flCwF slice $\delta : \mathsf{Sub}\,\Delta\,\Gamma$. We prefer it to slices because it yields expected notions of induction strictly.

$t : \mathsf{Tm}\,\Gamma\,A$ terms are sections of the displayed flCwF A. This can be viewed as a dependent functor, which maps structure in Γ to corresponding displayed structure in A, e.g. it maps objects to displayed objects, morphisms to displayed morphisms, and so on.

$\bullet : \mathsf{Con}$ is the terminal flCwF, whose underlying category is the terminal category with one object.

$\Gamma \triangleright A$ is the total flCwF of the A displayed flCwF, where every underlying set is given by packing up indexed underlying sets of A in Σ-s.

$\mathsf{U} : \mathsf{Ty}\,\Gamma$ is a displayed flCwF which is constantly (i.e. it does not depend on Γ) the flCwF of sets in Set_j, whose underlying category is the usual category of sets, and which has evident finite limit structure (Σ, Id, K from Definition 4). Hence, elements of $\mathsf{Tm}\,\Gamma\,\mathsf{U}$ are functors from Γ to $\mathbf{Set}$. $\mathsf{El} : \mathsf{Tm}\,\Gamma\,\mathsf{U} \to \mathsf{Ty}\,\Gamma$ interprets such functors as discrete displayed flCwFs, where all morphisms are identities.

$\Pi\,a\,B : \mathsf{Ty}\,\Gamma$ is a displayed flCwF where objects are dependent functions from the set of objects of a to the set of objects of B, and morphisms are pointwise morphisms. This is possible to build because the $a : \mathsf{Tm}\,\Gamma\,\mathsf{U}$ domain is discrete. A general Π type is not possible because of variance issues; this follows from the fact that the category of categories does not support Π types[**?**]. Finite limit structure is given pointwise, and is inherited from the domain B.

$\hat{\Pi}\,A\,B : \mathsf{Ty}\,\Gamma$ is the A-indexed direct product of the family of displayed flCwFs given by $B : A \to \mathsf{Ty}\,\Gamma$.

$\tilde{\Pi}\,A\,b$: $\mathsf{Tm}\,\Gamma\,\mathsf{U}$ is likewise the A-indexed direct product of the family of discrete displayed flCwFs given by $b : A \to \mathsf{Tm}\,\Gamma\,\mathsf{U}$.

$\mathsf{Id}\,a\,t\,u$: $\mathsf{Tm}\,\Gamma\,\mathsf{U}$ maps every object γ of Γ to the set $t\,\gamma = u\,\gamma$, and every morphism $\sigma : \mathsf{Sub}_\Gamma\,\gamma\,\gamma'$ to a function with type $t\,\gamma = u\,\gamma \to t\,\gamma' = u\,\gamma'$, which can be given by the functoriality of t and u. In short, $\mathsf{Id}\,a\,t\,u$ expresses pointwise equality of t and u dependent functors.

$\mathsf{IdU}\, a\, b : \mathsf{Ty}\, \Gamma$ is a displayed flCwF where the set of objects over $\gamma : \mathsf{Con}_\Gamma$ is $a\,\gamma = b\,\gamma$, that is, a strict equation of types in Set_j. Displayed morphisms are trivial (i.e. constantly $\top$) because of UIP in cETT.

Example 5. Algebras for pointed sets.

5 Infinitary Term Models

5.1 Induction on Signatures

sync

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.

5.2 Preservation of Isomorphisms

6 Term Models with Sort Equations

7 Related Work

8 Conclusion