For Finitary Induction-Induction, Induction Is Enough

Ambrus Kaposi
Eötvös Loránd University, Budapest, Hungary
akaposi@inf.elte.hu

András Kovács
Eötvös Loránd University, Budapest, Hungary
kovacsandras@inf.elte.hu

Ambroise Lafont
IMT Atlantique, Inria, LS2N CNRS, Nantes, France
ambroise.lafont@gmail.com

Abstract

Inductive-inductive types (IITs) are a generalisation of inductive types in type theory. They allow the mutual definition of types with multiple sorts where later sorts can be indexed by previous ones. An example is the Chapman-style syntax of type theory with conversion relations for each sort where e.g. the sort of types is indexed by contexts. In this paper we show that if a model of extensional type theory (ETT) supports indexed W-types, then it supports finitely branching IITs. We use a small internal type theory called the theory of signatures to specify IITs. We show that if a model of ETT supports the syntax for the theory of signatures, then it supports all IITs. We construct this syntax from indexed W-types using preterms and typing relations and prove its initiality following Streicher. The construction of the syntax and its initiality proof were formalised in Agda.

2012 ACM Subject Classification Theory of computation → Type theory

Keywords and phrases type theory, inductive types, inductive-inductive types

Digital Object Identifier 10.4230/LIPIcs.TYPES.2019.6

Supplementary Material The contents of Section 4 were formalised in Agda, the formalisation is available at `https://github.com/amblafont/UniversalII`.

Funding *Ambrus Kaposi*: this author was supported by the National Research, Development and Innovation Fund of Hungary, financed under the Thematic Excellence Programme funding scheme, Project no. ED_18-1-2019-0030, by the New National Excellence Program of the Ministry for Innovation and Technology, Project no. ÚNKP-19-4-ELTE-874, and by the Bolyai Fellowship of the Hungarian Academy of Sciences, Project no. BO/00659/19/3.
András Kovács: this author was supported by the European Union, co-financed by the European Social Fund (EFOP-3.6.3-VEKOP-16-2017-00002).
Ambroise Lafont: this author was supported by the CoqHoTT ERC Grant 637339.

Acknowledgements The authors would like to thank Thorsten Altenkirch, Rafaël Bocquet, Simon Boulier, Fredrik Nordvall-Forsberg and Jakob von Raumer for discussions on the topics of this paper. We also thank the anonymous reviewers for their helpful comments and suggestions.

1 Introduction

Many mutual inductive types can be reduced to indexed inductive types, where the index disambiguates different sorts. For example, consider the mutual inductive datatype with two sorts isEven and isOdd, defined by the following constructors.

$$\begin{aligned}
&\mathsf{isEven} &&: \mathbb{N} \to \mathsf{Set}\\
&\mathsf{isOdd} &&: \mathbb{N} \to \mathsf{Set}\\
&\mathsf{zeroEven} &&: \mathsf{isEven}\ \mathsf{zero}
\end{aligned}$$

$$\begin{aligned}&\mathsf{sucEven} &&: (n : \mathbb{N}) \to \mathsf{isOdd}\, n \to \mathsf{isEven}\, (\mathsf{suc}\, n)\\ &\mathsf{sucOdd} &&: (n : \mathbb{N}) \to \mathsf{isEven}\, n \to \mathsf{isOdd}\, (\mathsf{suc}\, n)\end{aligned}$$

This can be reduced to the following single inductive family where $\mathsf{isEven?}\,\mathsf{true}$ represents isEven and $\mathsf{isEven?}\,\mathsf{false}$ represent isOdd.

$$\begin{aligned}&\mathsf{isEven?} &&: \mathsf{Bool} \to \mathbb{N} \to \mathsf{Set}\\ &\mathsf{zeroEven} &&: \mathsf{isEven?}\,\mathsf{true}\,\mathsf{zero}\\ &\mathsf{sucEven} &&: (n : \mathbb{N}) \to \mathsf{isEven?}\,\mathsf{false}\, n \to \mathsf{isEven?}\,\mathsf{true}\,(\mathsf{suc}\, n)\\ &\mathsf{sucOdd} &&: (n : \mathbb{N}) \to \mathsf{isEven?}\,\mathsf{true}\, n \to \mathsf{isEven?}\,\mathsf{false}\,(\mathsf{suc}\, n)\end{aligned}$$

Inductive-inductive types (IITs [26]) allow the mutual definition of a type and a family of types over the first one. IITs were originally introduced to represent the well-typed syntax of type theory itself, and a prominent example is still Chapman's [13] syntax for a type theory. A minimised version is the IIT of contexts and types given by the following constructors.

$$\begin{aligned}&\mathsf{Con} &&: \mathsf{Set}\\ &\mathsf{Ty} &&: \mathsf{Con} \to \mathsf{Set}\\ &\mathsf{empty} &&: \mathsf{Con}\\ &\mathsf{ext} &&: (\mathit{\Gamma} : \mathsf{Con}) \to \mathsf{Ty}\,\mathit{\Gamma} \to \mathsf{Con}\\ &\mathsf{U} &&: (\mathit{\Gamma} : \mathsf{Con}) \to \mathsf{Ty}\,\mathit{\Gamma}\\ &\mathsf{El} &&: (\mathit{\Gamma} : \mathsf{Con}) \to \mathsf{Ty}\,(\mathsf{ext}\,\mathit{\Gamma}\,(\mathsf{U}\,\mathit{\Gamma}))\end{aligned}$$

This type has two sorts, Con and Ty. The ext constructor of Con refers to Ty and the Ty-constructor U refers to Con, hence the two sorts have to be defined simultaneously. Moreover, Ty is indexed over Con. This precludes a reduction analogous to the reduction of isEven–isOdd, as we would get a type indexed over itself. Another unique feature of IITs (which also holds for higher inductive types [29]) is that later constructors can refer to previous constructors: in our case, El mentions ext.

The elimination principle for the above IIT has the following two motives (one for each sort) and four methods (one for each constructor).

$$\begin{aligned}&\mathit{Con}^D &&: \mathsf{Con} \to \mathsf{Set}\\ &\mathit{Ty}^D &&: \mathit{Con}^D\,\mathit{\Gamma} \to \mathsf{Ty}\,\mathit{\Gamma} \to \mathsf{Set}\\ &\mathit{empty}^D &&: \mathit{Con}^D\,\mathsf{empty}\\ &\mathit{ext}^D &&: (\mathit{\Gamma}^D : \mathit{Con}^D\,\mathit{\Gamma}) \to \mathit{Ty}^D\,\mathit{\Gamma}^D\,A \to \mathit{Con}^D\,(\mathsf{ext}\,\mathit{\Gamma}\,A)\\ &U^D &&: (\mathit{\Gamma}^D : \mathit{Con}^D\,\mathit{\Gamma}) \to \mathit{Ty}^D\,\mathit{\Gamma}^D\,(\mathsf{U}\,\mathit{\Gamma})\\ &\mathit{El}^D &&: (\mathit{\Gamma}^D : \mathit{Con}^D\,\mathit{\Gamma}) \to \mathit{Ty}^D\,(\mathit{ext}^D\,\mathit{\Gamma}^D\,(U^D\,\mathit{\Gamma}^D))\,(\mathsf{El}\,\mathit{\Gamma})\end{aligned}$$

Above we used implicit quantifications for $\mathit{\Gamma} : \mathsf{Con}$ and $A : \mathsf{Ty}\,\mathit{\Gamma}$ to ease readability, e.g. Ty^D has an implicit parameter $\mathit{\Gamma}$ before its explicit parameter of type $\mathit{Con}^D\,\mathit{\Gamma}$.

Given the above motives and methods the elimination principle provides two functions

$$\begin{aligned}&\mathsf{elimCon} &&: (\mathit{\Gamma} : \mathsf{Con}) \to \mathit{Con}^D\,\mathit{\Gamma}\\ &\mathsf{elimTy} &&: (A : \mathsf{Ty}\,\mathit{\Gamma}) \to \mathit{Ty}^D\,(\mathsf{elimCon}\,\mathit{\Gamma})\,A\end{aligned}$$

with the following computation rules.

$$
\begin{aligned}
&\mathsf{elimCon}\,\mathsf{empty} &&= \mathit{empty}^D \\
&\mathsf{elimCon}\,(\mathsf{ext}\,\Gamma\,A) &&= \mathit{ext}^D\,(\mathsf{elimCon}\,\Gamma)\,(\mathsf{elimTy}\,A) \\
&\mathsf{elimTy}\,(\mathsf{U}\,\Gamma) &&= U^D\,(\mathsf{elimCon}\,\Gamma) \\
&\mathsf{elimTy}\,(\mathsf{El}\,\Gamma) &&= \mathit{El}^D\,(\mathsf{elimCon}\,\Gamma)
\end{aligned}
$$

The functions elimCon and elimTy are an example of a *recursive-recursive* definition (using nomenclature from [26]). This means two mutually defined functions where the type of the second function depends on the first function. The proof assistant Agda [28] allows defining such functions (even from non-IITs) and is currently the only proof assistant supporting IITs[1].

Reducing IITs to inductive types (more precisely, to indexed W-types) is an open problem. Forsberg [26] presented a reduction in extensional type theory, however, this only provides a simpler, non-recursive-recursive elimination principle. Hugunin [19] reduced several IITs to inductive types, working inside a cubical type theory, but he also only constructed the simple eliminator. To illustrate the difference, we list the motives, methods and the simple elimination principle for the Con–Ty example. Again, we use implicit quantifications.

$$
\begin{aligned}
&\mathit{Con}^S &&: \mathsf{Con} \to \mathsf{Set} \\
&\mathit{Ty}^S &&: \mathsf{Ty}\,\Gamma \to \mathsf{Set} \\
&\mathit{empty}^S &&: \mathit{Con}^S\,\mathsf{empty} \\
&\mathit{ext}^S &&: \mathit{Con}^S\,\Gamma \to \mathit{Ty}^S\,A \to \mathit{Con}^S\,(\mathsf{ext}\,\Gamma\,A) \\
&U^S &&: \mathit{Con}^S\,\Gamma \to \mathit{Ty}^S\,(\mathsf{U}\,\Gamma) \\
&\mathit{El}^S &&: \mathit{Con}^S\,\Gamma \to \mathit{Ty}^S\,(\mathsf{El}\,\Gamma) \\
&\mathsf{selimCon} &&: (\Gamma : \mathsf{Con}) \to \mathit{Con}^S\,\Gamma \\
&\mathsf{selimTy} &&: (A : \mathsf{Ty}\,\Gamma) \to \mathit{Ty}^S\,A
\end{aligned}
$$

This simple elimination principle is not capable of defining standard (metacircular) interpretation [4] of our small syntax. Using pattern matching notation, this interpretation is the following:

$$
\begin{aligned}
&[\![-]\!] &&: \mathsf{Con} \to \mathsf{Set}_1 \\
&[\![-]\!] &&: [\![\Gamma]\!] \to \mathsf{Set}_1 \\
&[\![\mathsf{empty}]\!] &&:= \top \\
&[\![\mathsf{ext}\,\Gamma\,A]\!] &&:= (\gamma : [\![\Gamma]\!]) \times [\![A]\!]\,\gamma \\
&[\![\mathsf{U}\,\Gamma]\!]\,\gamma &&:= \mathsf{Set} \\
&[\![\mathsf{El}\,\Gamma]\!]\,(\gamma, X) &&:= X
\end{aligned}
$$

The reason that we need the general elimination principle to define $[\![-]\!]$ is that $[\![-]\!]$ for types refers to $[\![-]\!]$ for contexts, hence this function is recursive-recursive.

Kaposi, Kovács, and Altenkirch [21] introduced a small type theory, called the theory of signatures, to describe quotient inductive-inductive types (QIIT). QIITs are generalisations of IITs where equality constructors are also allowed. A QIIT signature is a context in

[1] An experimental version of Coq with IITs is also available on GitHub.

the theory of QIIT signatures, for example natural numbers are specified by the context $(Nat : \mathsf{U}, zero : Nat, suc : Nat \to Nat)$ of length three (Nat, $zero$ and suc are variable names). The theory of QIIT signatures is itself a QIIT. In ibid., it is proved that if a model of extensional type theory supports the theory of QIIT signatures, then it supports all QIITs.

By omitting the equality type former from the theory of QIIT signatures, we obtain a theory of IIT signatures and the construction is still valid. It follows that if a model of extensional type theory supports the theory of IIT signatures, it supports all IITs.

In this paper we show that any model of extensional type theory with indexed W-types supports the theory of IIT signatures, and as a consequence all IITs. The difficulty in this construction is that the theory of IIT signatures is itself a QIIT, it is both inductive-inductive and has equality constructors. However, it can be seen as the well-typed syntax of a small type theory without any computation rules. Hence we can represent the syntax of normal forms without quotienting. We construct this well-typed normal syntax using preterms and typing relations from indexed W-types. Finally, we prove the elimination principle in the style of the initiality proof of Streicher.

Streicher [30] constructs the syntactic model of type theory using well-typed preterms and then shows initiality of this model by (1) defining a partial map to any other model by induction on preterms and (2) showing that whenever this partial function receives a well-typed preterm on its input it actually gives an output. Instead of defining a partial function, we define the graph of the same function as a relation and then show that it is functional as a second step. This can be seen as an indexed variant of the construction using partial functions.

Just as [21], we only consider finitary IITs, that is, constructors can only have a finite number of recursive arguments. An example constructor for Con–Ty which is not allowed is the following:

$$\Pi_\infty : (\Gamma : \mathsf{Con}) \to (\mathbb{N} \to \mathsf{Ty}\,\Gamma) \to \mathsf{Ty}\,\Gamma$$

Structure of paper and list of contributions

We describe related work in Section 1.1, and explain our notation and Agda formalisaton in Section 1.2. Then the following three sections describe our three contributions:

- Section 2. We define what it means for a model of extensional type theory (ETT, Definition 1) to support all inductive-inductive types (IITs): Definition 12. The novel contribution here is a (predicative) Church encoding of signatures following [8].
- Section 3. In Theorem 23, we show that if a model of ETT supports the theory of IIT signatures (Definition 15), then it supports IITs. This is an adaptation of a proof in [21].
- Section 4. Our main contribution is showing that if a model of ETT supports indexed W-types, then it supports the theory of IIT signatures (Theorem 57), and hence, all IITs (Corollary 58).

We list further work in Section 5.

The contents of this paper were presented at the TYPES 2019 conference in Oslo [22].

1.1 Related Work

The current work builds heavily on the work of Kaposi et al. [21] on finitary quotient inductive-inductive types (QIITs); we reuse both QIIT syntax and semantics by restricting to IITs, and we reuse the term model construction of QIITs as well. We also make use of the extension to infinitary QIITs [24] to derive the specification of the elimination principle for the theory of IIT signatures.

IITs (although not by this name) were first used to describe the well-typed syntax of type theory [15, 13]. Agda supported these general inductive definitions even before they were named IITs and given semantics by Nordvall Forsberg and Setzer [27]. Nordvall Forsberg's thesis [26] contains a specification similar in style to Dybjer and Setzer's codes for inductive-recursive types [17]. He also develops a categorical semantics based on dialgebras and provides a reduction of IITs to indexed inductive types, however only constructs the simple elimination principle as opposed to the general one. Altenkirch et al. [2] define signatures for QIITs (thus IITs as well) and their categorical semantics, however without proving existence of initial algebras. Their notion of signature, like Nordvall Forsberg's, involves more encoding overhead than ours.

Cartmell [12] introduced generalised algebraic theories using a type-theoretic syntax. Removing equations from his signatures and only considering finite signatures, we obtain finitary IIT signatures similar to ours. He does not consider constructing initial algebras using simpler classes of inductive types.

Hugunin [19] constructs several IITs in cubical Agda from inductive types. In this setting, the lack of UIP makes constructions significantly more involved, and essentially involves coinductive-coinductive well-formedness predicates defined as homotopy limits. Hugunin does not consider a generic syntax of IITs and only works on specific examples (although the examples vary greatly). He also only constructs simple elimination principles.

Streicher [30] presents an interpretation of the well-formed presyntax of a type theory into a categorical model, which is an important ingredient in constructing an initial model, although he does not present details on the construction of the term model or its initiality proof. Our initiality proof can be seen as an indexed variant of his construction (see Subsection 4.2 for a comparison).

Voevodsky was interested in constructing initial models of type theories from presyntaxes. Inspired by this, Brunerie et al. [10] formalised Streicher's proof in Agda for a type theory with Π, Σ, $\mathbb{N}$, identity types and an infinite hierarchy of universes. They used UIP, function extensionality and quotient types in the formalisation. In this paper we construct a type theory without computation rules, hence we avoid using quotients.

Intrinsic (well-typed) syntaxes for type theories were constructed using IITs [13], inductive-recursive types [15, 6] and QIITs [4]. In this paper we avoid using such general classes of inductive types as our goal is to reduce IITs to indexed inductive types.

Reducing general classes of inductive types to simpler classes has a long tradition in type theory. Indexed W-types were reduced to W-types [3] (using the essentially Streicher's idea of preterms and a typing predicate), small inductive-recursive types to indexed W-types [25], mutual inductive types to indexed W-types [23], W-types to natural numbers and quotients [1]. (Q)IITs can be reduced to quotient inductive types using the reduction of generalised algebraic theories to essentially algebraic theories [12]. Using the same reduction as mutual inductive types to indexed inductive types, (Q)IITs with more than two sorts can be reduced to (Q)IITs with only two sorts [20].

Awodey, Frey and Speight [8] construct inductive types using a restricted Church encoding in a type theory with an impredicative universe. We use the predicative version of their encoding to define IIT signatures.

Our reduction of IITs to indexed inductive types goes through two steps: first we construct a concrete QIIT using inductive types, then we construct all IITs from this particular QIIT. A more direct approach is proposed by [5]: here the initial algebra would be constructed directly for any IIT signature without going through an intermediate step.

1.2 Notation and Formalisation

▶ **Definition 1** (Model of extensional type theory (ETT)). *By a model of ETT we mean a category with families (CwF) [16, 18] with a countable predicative hierarchy of universes closed under the following type formers: Π, Σ, $\top$ and an identity type with uniqueness of identity proofs and equality reflection.*

We will use Agda-like type theoretic syntax to work in the internal language of models of ETT:

- Universes are written Set_i. We usually omit level indices in this paper.
- Π types are notated as $(x : A) \to B$, or as $A \to B$ when non-dependent. We sometimes omit function arguments, by implicitly generalising over variables.
- Σ-types, notated either as $(x : A) \times B$, or as $\sum_x B$ when we want to leave the type of the first projection implicit. Projections are either named or given by proj_1 and proj_2. We use $A \times B$ for non-dependent pairs.
- The unit type $\top$ has the constructor tt which is definitionally equal to all elements of $\top$.
- The equality (identity) type is written $t = u$, it has a constructor $\mathsf{refl} : t = t$, and equality reflection, hence we use the same $=$ sign for definitional equality. We occasionally indicate by ${}_{e_1,\dots,e_n}\#t$ that t is well-typed thanks to the equalities $e_1,\dots,e_n$. To construct proofs, sometimes we write equational reasoning, e.g. $f\,a \overset{e}{=} f\,b$ where $e : a = b$. We also have uniqueness of identity proofs (UIP), expressing $(e : t = t) \to e = \mathsf{refl}$. Note that function extensionality, expressing $((x : A) \to f\,x = g\,x) \to f = g$ is derivable.

The contents of Section 4 were formalised in Agda, the formalisation is available at `https://github.com/amblafont/UniversalII`. Agda's pattern matching mechanism implies uniqueness of identity proofs, we assumed function extensionality as an axiom and used rewrite rules [14] to obtain limited equality reflection.

2 A Definition of Inductive-Inductive Types

In this section we specify what it means that a model of ETT supports IITs. We first define the notion of IIT signature. Signatures for algebraic theories are usually given by inductive definitions. On the one hand, we take this even further: our notion of signature is given by a small type theory tailor-made to describe signatures, which we call the *theory of IIT signatures*. On the other hand we would like to avoid using a complicated inductive definition (a type theory is a quotient inductive-inductive type [4]) to describe a simpler class of inductive types. Hence we use a Church encoding [8] of the theory of IIT signatures, thereby avoiding the need for pre-existing inductive definitions. Another feature of our signatures is that they can include types from the model of ETT (such as $\mathbb{N}$ in the isEven–isOdd). This is why signatures are specified internally to the particular model of ETT.[2]

We define the theory of IIT signatures by saying what its algebras (models) are. We call the *theory of IIT signatures algebras* simply *signature algebras*. The theory of signatures is a small type theory consisting of a (1) a substitution calculus (category with families, CwF [16]) equipped with (2) a universe, (3) a function space where the domain is in the universe and (4) another function space with external domain. We explain the usage of these type formers through examples after the definition.

[2] There is another method inspired by Capriotti [11] which allows stating what it means that any CwF $\mathcal{C}$ (not necessarily a model of ETT) supports IITs with definitional computation rules. In this method, signatures are described in the internal language of $\hat{\mathcal{C}}$, the presheaf model over $\mathcal{C}$. We do not use this approach because it is more technical, and it would not strengthen our main result Corollary 58 as the proof of Theorem 57 needs $\mathcal{C}$ to be a model of ETT.

▶ **Definition 2** (Signature algebra, SignAlg)**.** *In a model of ETT, a signature algebra is an iterated Σ type consisting of the following four (families of) sets, 17 operations and 18 equalities.*

(1) Substitution calculus

Con	$: \mathsf{Set}$
Ty	$: \mathsf{Con} \to \mathsf{Set}$
Sub	$: \mathsf{Con} \to \mathsf{Con} \to \mathsf{Set}$
Tm	$: (\Gamma : \mathsf{Con}) \to \mathsf{Ty}\,\Gamma \to \mathsf{Set}$
id	$: \mathsf{Sub}\,\Gamma\,\Gamma$
$- \circ -$	$: \mathsf{Sub}\,\Theta\,\Delta \to \mathsf{Sub}\,\Gamma\,\Theta \to \mathsf{Sub}\,\Gamma\,\Delta$
ass	$: (\sigma \circ \delta) \circ \nu = \sigma \circ (\delta \circ \nu)$
idl	$: \mathsf{id} \circ \sigma = \sigma$
idr	$: \sigma \circ \mathsf{id} = \sigma$
$-[-]$	$: \mathsf{Ty}\,\Delta \to \mathsf{Sub}\,\Gamma\,\Delta \to \mathsf{Ty}\,\Gamma$
$-[-]$	$: \mathsf{Tm}\,\Delta\,A \to (\sigma : \mathsf{Sub}\,\Gamma\,\Delta) \to \mathsf{Tm}\,\Gamma\,(A[\sigma])$
$[\mathsf{id}]$	$: A[\mathsf{id}] = A$
$[\circ]$	$: A[\sigma \circ \delta] = A[\sigma][\delta]$
$[\mathsf{id}]$	$: t[\mathsf{id}] = t$
$[\circ]$	$: t[\sigma \circ \delta] = t[\sigma][\delta]$
$\cdot$	$: \mathsf{Con}$
ϵ	$: \mathsf{Sub}\,\Gamma\,\cdot$
$\cdot\eta$	$: (\sigma : \mathsf{Sub}\,\Gamma\,\cdot) \to \sigma = \epsilon$
$- \rhd -$	$: (\Gamma : \mathsf{Con}) \to \mathsf{Ty}\,\Gamma \to \mathsf{Con}$
$-, -$	$: (\sigma : \mathsf{Sub}\,\Gamma\,\Delta) \to \mathsf{Tm}\,\Gamma\,(A[\sigma]) \to \mathsf{Sub}\,\Gamma\,(\Delta \rhd A)$
π_1	$: \mathsf{Sub}\,\Gamma\,(\Delta \rhd A) \to \mathsf{Sub}\,\Gamma\,\Delta$
π_2	$: (\sigma : \mathsf{Sub}\,\Gamma\,(\Delta \rhd A)) \to \mathsf{Tm}\,\Gamma\,(A[\pi_1 \sigma])$
$\pi_1\beta$	$: \pi_1(\sigma, t) = \sigma$
$\pi_2\beta$	$: \pi_2(\sigma, t) = t$
$\pi\eta$	$: (\pi_1\,\sigma, \pi_2\,\sigma) = \sigma$
$,\circ$	$: (\sigma, t) \circ \delta = (\sigma \circ \delta, t[\delta])$

(2) Universe

U	$: \mathsf{Ty}\,\Gamma$
El	$: \mathsf{Tm}\,\Gamma\,\mathsf{U} \to \mathsf{Ty}\,\Gamma$
$\mathsf{U}[]$	$: \mathsf{U}[\sigma] = \mathsf{U}$
$\mathsf{El}[]$	$: (\mathsf{El}\,a)[\sigma] = \mathsf{El}\,(a[\sigma])$

(3) Inductive parameters

Π	$: (a : \mathsf{Tm}\,\Gamma\,\mathsf{U}) \to \mathsf{Ty}\,(\Gamma \rhd \mathsf{El}\,a) \to \mathsf{Ty}\,\Gamma$
$- @ -$	$: \mathsf{Tm}\,\Gamma\,(\Pi\,a\,B) \to (u : \mathsf{Tm}\,\Gamma\,(\mathsf{El}\,a)) \to \mathsf{Tm}\,\Gamma\,(\mathsf{El}\,(B[\mathsf{id}, u]))$
$\Pi[]$	$: (\Pi\,a\,B)[\sigma] = \Pi\,(a[\sigma])\,(B[\sigma \circ \mathsf{p}, \mathsf{q}])$
$@[]$	$: (t \,@\, \alpha)[\sigma] = (t[\sigma])\,@(\alpha[\sigma])$

(4) External parameters

$$\begin{aligned}
&\hat{\Pi} &&: (T : \mathsf{Set}) \to (T \to \mathsf{Ty}\,\Gamma) \to \mathsf{Ty}\,\Gamma \\
&-\hat{@}- &&: \mathsf{Tm}\,\Gamma\,(\hat{\Pi}\,T\,B) \to (\alpha : T) \to \mathsf{Tm}\,\Gamma\,(B\,\alpha) \\
&\hat{\Pi}[] &&: (\hat{\Pi}\,T\,B)[\sigma] = \hat{\Pi}\,T\,(\lambda\alpha.(B\,\alpha)[\sigma]) \\
&\hat{@}[] &&: (t\,\hat{@}\,\alpha)[\sigma] = (t[\sigma])\,\hat{@}\,\alpha
\end{aligned}$$

Given an $M : \mathsf{SignAlg}$*, we denote its components by* Con^M, Ty^M, Sub^M, Tm^M, id^M*, and so on. We omit the indices if there is only one signature algebra in scope (e.g. in Definition 3 and Example 4).*

▶ **Definition 3** (Abbreviations). *For a signature algebra, we use* $\mathsf{wk} : \mathsf{Sub}\,(\Gamma \rhd A)\,\Gamma$ *to mean* $\pi_1\,\mathsf{id}$*. We recover de Bruijn indices by setting* $0 := \pi_2\,\mathsf{id}$ *and* $1 + n := n[\mathsf{wk}]$*.* $\Pi\,a\,(B[\mathsf{wk}])$ *is abbreviated by* $a \Rightarrow B$*,* $\hat{\Pi}\,T\,(\lambda\_.B)$ *by* $T \hat{\Rightarrow} B$*.*

▶ **Example 4** (Example contexts in a signature algebra). Given a signature algebra, we can define a context which specifies natural numbers. For readability, an informal version of the same context is displayed on the right using variable names.

$$\cdot \rhd \mathsf{U} \rhd z : \mathsf{El}\,0 \rhd s : 1 \Rightarrow \mathsf{El}\,1 \qquad\qquad \cdot \rhd N : \mathsf{U} \rhd z : \mathsf{El}\,N \rhd s : N \Rightarrow \mathsf{El}\,N$$

We start with the empty context $\cdot$, then we declare a sort U, then we declare an operator producing an element of the sort denoted by $\mathsf{El}\,0$ where 0 is the de Bruijn index referring to the sort. Finally, we declare an operator which takes as input an element of the sort (now it became de Bruijn index 1) and produces an element of the same sort. Note the asymmetry of the function type $\Rightarrow$: the domain needs to be an element of U, while the codomain can be any type (including another function type). This ensures strict positivity of the operators.

Lists with elements of a given $T : \mathsf{Set}$ type are given by the following context. Here we use the function space with external domain $\hat{\Rightarrow}$ to include a T in the signature. For readability, we omit the λ and the superscripts and we do not write the compatibility condition. On the right we list the same signature with variable names.

$$\cdot \rhd \mathsf{U} \rhd \mathsf{El}\,0 \rhd T \hat{\Rightarrow} 1 \Rightarrow \mathsf{El}\,1 \qquad\qquad \cdot \rhd L : \mathsf{U} \rhd \mathit{nil} : \mathsf{El}\,L \rhd \mathit{cons} : T \hat{\Rightarrow} L \Rightarrow \mathsf{El}\,L$$

The Con–Ty example from Section 1 is given by the following context.

$$\begin{array}{lll}
\cdot \rhd & \cdot \rhd & \\
\mathsf{U} \rhd & \mathit{Con} & : \mathsf{U} \rhd \\
0 \Rightarrow \mathsf{U} \rhd & \mathit{Ty} & : \mathit{Con} \Rightarrow \mathsf{U} \rhd \\
\mathsf{El}\,1 \rhd & \mathit{empty} & : \mathsf{El}\,\mathit{Con} \rhd \\
\Pi\,2\,(2 \mathbin{@} 0 \Rightarrow \mathsf{El}\,3) \rhd & \mathit{ext} & : \Pi\,(\Gamma : \mathit{Con})\,(\mathit{Ty} \mathbin{@} \Gamma \Rightarrow \mathsf{El}\,\mathit{Con}) \rhd \\
\Pi\,3\,(\mathsf{El}\,(3 \mathbin{@} 0)) \rhd & U & : \Pi\,(\Gamma : \mathit{Con})\,(\mathsf{El}\,(\mathit{Ty} \mathbin{@} \Gamma)) \rhd \\
\Pi\,4\,(\mathsf{El}\,(4 \mathbin{@} (2 \mathbin{@} 0 \mathbin{@} (1 \mathbin{@} 0)))) & \mathit{El} & : \Pi\,(\Gamma : \mathit{Con})\,(\mathsf{El}\,(\mathit{Ty} \mathbin{@} (\mathit{ext} \mathbin{@} \Gamma \mathbin{@} (U \mathbin{@} \Gamma))))
\end{array}$$

The above examples are contexts in any signature algebra, and we could take this as a definition of signature: $(M : \mathsf{SignAlg}) \to \mathsf{Con}^M$ is the usual Church-encoding of contexts. However (as we will see in Remark 24) the notion of constructor for such signatures would be too strong. Another approach would be to assume that there is a syntax for signature algebras (an initial signature algebra), and then a signature would be a context in this signature algebra. We will define syntactic signatures using this approach in the next section (Definition 16), but for now we do not want to assume the existence of any inductive type. Instead, we will use a restricted Church encoding. This requires the notion of morphism of signatures.

The notion of morphism is determined by the notion of algebra [24], but we include it here for completeness.

▶ **Definition 5** (Signature morphism, SignMor). *A morphism from signature algebras M to N denoted* $\mathsf{SignMor}\,M\,N$ *consists of four functions and 17 equalities expressing that the functions preserve the operations of the two algebras. We use the same naming as in Definition 2 and use superscripts to denote which algebra is meant.*

(1) Substitution calculus

$$\begin{array}{lll}
\mathsf{Con} & : \mathsf{Con}^M & \to \mathsf{Con}^N \\
\mathsf{Ty} & : \mathsf{Ty}^M\,\Gamma & \to \mathsf{Ty}^N\,(\mathsf{Con}\,\Gamma) \\
\mathsf{Sub} & : \mathsf{Sub}^M\,\Gamma\,\Delta & \to \mathsf{Sub}^N\,(\mathsf{Con}\,\Gamma)\,(\mathsf{Con}\,\Delta) \\
\mathsf{Tm} & : \mathsf{Tm}^M\,\Gamma\,A & \to \mathsf{Tm}^N\,(\mathsf{Con}\,\Gamma)\,(\mathsf{Ty}\,A) \\
\mathsf{id} & : \mathsf{Sub}\,\mathsf{id}^M & = \mathsf{id}^N \\
\circ & : \sigma \circ^M \delta & = \mathsf{Sub}\,\sigma \circ^N \mathsf{Sub}\,\delta \\
[] & : A[\sigma]^M & = \mathsf{Ty}\,A[\mathsf{Sub}\,\sigma]^N \\
[] & : t[\sigma]^M & = \mathsf{Tm}\,t[\mathsf{Sub}\,\sigma]^N \\
\cdot & : \mathsf{Con}\,\cdot^M & = \cdot^N \\
\epsilon & : \mathsf{Sub}\,\epsilon^M & = \epsilon^N \\
\rhd & : \mathsf{Con}\,(\Gamma \rhd^M A) & = \mathsf{Con}\,\Gamma \rhd^N \mathsf{Ty}\,A \\
, & : \mathsf{Sub}\,(\sigma,^M t) & = \mathsf{Sub}\,\sigma,^N \mathsf{Tm}\,t \\
\pi_1 & : \mathsf{Sub}\,(\pi_1{}^M\,\sigma) & = \pi_1{}^N\,(\mathsf{Sub}\,\sigma) \\
\pi_2 & : \mathsf{Tm}\,(\pi_2{}^M\,\sigma) & = \pi_2{}^N\,(\mathsf{Sub}\,\sigma)
\end{array}$$

(2) Universe

$$\begin{array}{lll}
\mathsf{U} & : \mathsf{Ty}\,\mathsf{U}^M & = \mathsf{U}^N \\
\mathsf{El} & : \mathsf{Ty}\,(\mathsf{El}^M\,a) & = \mathsf{El}^N\,(\mathsf{Tm}\,a)
\end{array}$$

(3) Inductive parameters

$$\begin{array}{lll}
\Pi & : \mathsf{Ty}\,(\Pi^M\,a\,B) & = \Pi^N\,(\mathsf{Tm}\,a)\,(\mathsf{Ty}\,B) \\
@ & : \mathsf{Tm}\,(t\,@^M\,u) & = \mathsf{Tm}\,t\,@^N\,\mathsf{Tm}\,u
\end{array}$$

(4) External parameters

$$\begin{array}{lll}
\hat{\Pi} & : \mathsf{Ty}\,(\hat{\Pi}^M\,T\,B) & = \hat{\Pi}^N\,T\,(\lambda\alpha.\mathsf{Ty}\,(B\,\alpha)) \\
\hat{@} & : \mathsf{Tm}\,(t\,\hat{@}^M\,\alpha) & = \mathsf{Tm}\,t\,\hat{@}^N\,\alpha
\end{array}$$

Given an $f : \mathsf{SignMor}\,M\,N$*, we denote its first four components just by* f_{Con}, f_{Ty}, f_{Sub}, f_{Tm} *or just write f if it is clear which one is meant.*

We define IIT signatures using the Church encoding introduced by Awodey, Frey and Speight [8]. A difference is that we avoid impredicativity. This restricts the possible eliminations on signatures: we can only eliminate into a universe which is smaller than the level of signatures. However, this still covers all eliminations in this paper, and it is also not an issue for us that signatures do not live in the smallest universe.

▶ **Definition 6** (IIT signature). *An IIT signature is a context in an arbitrary signature algebra, which is also compatible with morphisms:*

$$\begin{aligned}\mathsf{Sign} := \big(&sig : (M : \mathsf{SignAlg}) \to \mathsf{Con}^M\big)\times \\ &\big((M\,N : \mathsf{SignAlg})(f : \mathsf{SignMor}\,M\,N) \to f_{\mathsf{Con}}\,(sig\,M) = sig\,N\big).\end{aligned}$$

The compatibility condition says that if we obtain an M-context using sig at signature algebra M and then we transport it to N using f, we get the same N-context as directly applying sig to N.

The lack of impredicativity implies that our notion of signatures do not form a signature algebra.

▶ **Lemma 7.** *There is no* $M : \mathsf{SignAlg}$*, in which* $\mathsf{Con}^M = \mathsf{Sign}$.

Proof. If the Con component in $\mathsf{SignAlg}$ is Set_i, then $\mathsf{SignAlg}$ is in Set_{i+1}, but as Sign is defined as $(\mathsf{SignAlg} \to \dots) \times \dots$, it is at least in Set_{i+1}, so we can't choose $\mathsf{Con}^M : \mathsf{Set}_i$ to be $\mathsf{Sign} : \mathsf{Set}_{i+1}$. ◀

Note that the notion of IIT signature is relative to a model of ETT: it is expressed as a term (of a function type) in the model. This is necessary because of the function space $\hat\Pi$, which has as domain an arbitrary type in the model. We make use of $\hat\Pi$ in signatures with external parameters, like the type of the elements in lists.

▶ **Example 8** (Example signature). Now we can formally describe the contexts given in Example 4 as signatures. For natural numbers, we have the following pair of functions. The second function returns an equality proof which we describe using equational reasoning.

$$\begin{aligned}&(nat, natc) := \\ &\big(\lambda M.(\cdot^M \rhd^M \mathsf{U}^M \rhd^M \mathsf{El}^M\,0^M \rhd^M 1^M \Rightarrow^M \mathsf{El}^M\,1^M), \\ &\quad \lambda M\,N\,f\,.\,f_{\mathsf{Con}}\,(\cdot^M \rhd^M \mathsf{U}^M \rhd^M \mathsf{El}^M\,0^M \rhd^M 1^M \Rightarrow^M \mathsf{El}^M\,1^M) = \\ &\qquad\quad f_{\mathsf{Con}}\,(\cdot^M \rhd^M \mathsf{U}^M \rhd^M \mathsf{El}^M\,0^M) \rhd^N f_{\mathsf{Ty}}\,(1^N \Rightarrow^N \mathsf{El}^N\,1^N) = \\ &\qquad\quad f_{\mathsf{Con}}\,(\cdot^M \rhd^M \mathsf{U}^M) \rhd^N f_{\mathsf{Ty}}\,(\mathsf{El}^M\,0^M) \rhd^N f_{\mathsf{Tm}}\,1^N \Rightarrow^M f_{\mathsf{Ty}}\,(\mathsf{El}^N\,1^N) = \\ &\qquad\quad f_{\mathsf{Con}}\,\cdot^M \rhd^N f_{\mathsf{Ty}}\,\mathsf{U}^M \rhd^N \mathsf{El}^N\,(f_{\mathsf{Tm}}\,0^M) \rhd^N 1^M \Rightarrow^M \mathsf{El}^M\,(f_{\mathsf{Tm}}\,1^N) = \\ &\qquad\quad \cdot^N \rhd^N \mathsf{U}^N \rhd^N \mathsf{El}^N\,0^N \rhd^N 1^N \Rightarrow^N \mathsf{El}^N\,1^N\big)\end{aligned}$$

The first component builds the context describing natural numbers in M, the second one uses the fact that f is a morphism, that is, it preserves all operations.

The signatures for lists and Con–Ty can be given analogously.

Given a model of ETT and an IIT signature in it, we would like to say what it means that the model supports the given IIT. For this we define the signature algebra ADS which will provide notions of algebras, displayed algebras and sections for each signature. This is the same as the $-^{\mathsf{A}}$, $-^{\mathsf{D}}$ and $-^{\mathsf{S}}$ operations in [21]. Before defining ADS, we illustrate its usage by an example.

▶ **Example 9** (Algebras, displayed algebras and sections for natural numbers). For the signature of natural numbers as given in Example 8, algebras are given by the Σ-type $(N : \mathsf{Set}) \times N \times (N \to N)$. A displayed algebra over (N, z, s) is given by the Σ-type

$$(N^D : N \to \mathsf{Set}) \times N^D\,z \times ((n : N) \to N^D\,n \to N^D\,(s\,n)),$$

and a section of a displayed algebra (N^D, z^D, s^D) over (N, z, s) is given by the Σ-type

$$(N^S : (n : N) \to N^D\,n) \times (N^S\,z = z^D) \times ((n : N) \to N^S\,(s\,n) = s^D\,n\,(N^S\,n)).$$

Displayed algebras over the initial algebra are called motives and methods of the eliminator, while a section of a displayed algebra over the initial algebra is the eliminator together with its computation rules.

▶ **Definition 10** (The signature algebra ADS). *We define an element of* SignAlg *by listing all its components* Con, Ty, Sub, *and so on, one per row. Each such component has three parts denoted by* $^{\mathsf{A}}$, $^{\mathsf{D}}$ *and* $^{\mathsf{S}}$, *respectively. The equality components of* SignAlg *are omitted as they are all reflexivity.*

$$\begin{array}{lll}
(\Gamma^{\mathsf{A}} : \mathsf{Set}) & \times(\Gamma^{\mathsf{D}} : \Gamma^{\mathsf{A}} \to \mathsf{Set}) & \times(\Gamma^{\mathsf{S}} : (\gamma : \Gamma^{\mathsf{A}}) \to \Gamma^{\mathsf{D}}\,\gamma \to \mathsf{Set}) \\
(A^{\mathsf{A}} : \Gamma^{\mathsf{A}} \to \mathsf{Set}) & \times(A^{\mathsf{D}} : \Gamma^{\mathsf{D}}\,\gamma \to A^{\mathsf{A}}\,\gamma \to \mathsf{Set}) & \times(A^{\mathsf{S}} : \Gamma^{\mathsf{S}}\,\gamma\,\gamma^{D} \to (\alpha : A^{\mathsf{A}}\,\gamma) \to \\
 & & \quad A^{\mathsf{D}}\,\gamma^{D}\,\alpha \to \mathsf{Set}) \\
(\sigma^{\mathsf{A}} : \Gamma^{\mathsf{A}} \to \Delta^{\mathsf{A}}) & \times(\sigma^{\mathsf{D}} : \Gamma^{\mathsf{D}}\,\gamma \to \Delta^{\mathsf{D}}\,(\sigma^{\mathsf{A}}\,\gamma)) & \times(\sigma^{\mathsf{S}} : \Gamma^{\mathsf{S}}\,\gamma\,\gamma^{D} \to \\
 & & \quad \Delta^{\mathsf{S}}\,(\sigma^{\mathsf{A}}\,\gamma)\,(\sigma^{\mathsf{D}}\,\gamma^{D})) \\
(t^{\mathsf{A}} : (\gamma : \Gamma^{\mathsf{A}}) \to A^{\mathsf{A}}\,\gamma) & \times(t^{\mathsf{D}} : (\gamma^{D} : \Gamma^{\mathsf{D}}\,\gamma) \to & \times(t^{\mathsf{S}} : (\gamma^{S} : \Gamma^{\mathsf{S}}\,\gamma\,\gamma^{D}) \to \\
 & \quad A^{\mathsf{D}}\,\gamma^{D}\,(t^{\mathsf{A}}\,\gamma)) & \quad A^{\mathsf{S}}\,(t^{\mathsf{A}}\,\gamma)\,(t^{\mathsf{D}}\,\gamma^{D})) \\
\mathsf{id}^{\mathsf{A}}\,\gamma := \gamma & \mathsf{id}^{\mathsf{D}}\,\gamma^{D} := \gamma^{D} & \mathsf{id}^{\mathsf{S}}\,\gamma^{S} := \gamma^{S} \\
(\sigma \circ \delta)^{\mathsf{A}}\,\gamma := \sigma^{\mathsf{A}}\,(\delta^{\mathsf{A}}\,\gamma) & (\sigma \circ \delta)^{\mathsf{D}}\,\gamma^{D} := \sigma^{\mathsf{D}}\,(\delta^{\mathsf{D}}\,\gamma^{D}) & (\sigma \circ \delta)^{\mathsf{S}}\,\gamma^{S} := \sigma^{\mathsf{S}}\,(\delta^{\mathsf{S}}\,\gamma^{S}) \\
(A[\sigma])^{\mathsf{A}}\,\gamma := A^{\mathsf{A}}\,(\sigma^{\mathsf{A}}\,\gamma) & (A[\sigma])^{\mathsf{D}}\,\gamma^{D} := A^{\mathsf{D}}\,(\sigma^{\mathsf{D}}\,\gamma^{D}) & (A[\sigma])^{\mathsf{S}}\,\gamma^{S} := A^{\mathsf{S}}\,(\sigma^{\mathsf{S}}\,\gamma^{S}) \\
(t[\sigma])^{\mathsf{A}}\,\gamma := t^{\mathsf{A}}\,(\sigma^{\mathsf{A}}\,\gamma) & (t[\sigma])^{\mathsf{D}}\,\gamma^{D} := t^{\mathsf{D}}\,(\sigma^{\mathsf{D}}\,\gamma^{D}) & (t[\sigma])^{\mathsf{S}}\,\gamma^{S} := t^{\mathsf{S}}\,(\sigma^{\mathsf{S}}\,\gamma^{S}) \\
\cdot^{\mathsf{A}} := \top & \cdot^{\mathsf{D}}\,_ := \top & \cdot^{\mathsf{S}}\,_\,_ := \top \\
\epsilon^{\mathsf{A}}\,_ := \mathsf{tt} & \epsilon^{\mathsf{D}}\,_ := \mathsf{tt} & \epsilon^{\mathsf{S}}\,_ := \mathsf{tt} \\
(\Gamma \rhd A)^{\mathsf{A}} := & (\Gamma \rhd A)^{\mathsf{D}}\,(\gamma, \alpha) := & (\Gamma \rhd A)^{\mathsf{S}}\,(\gamma, \alpha)\,(\gamma^{D}, \alpha^{D}) := \\
\quad (\gamma : \Gamma^{\mathsf{A}}) \times A^{\mathsf{A}}\,\gamma & \quad (\gamma^{D} : \Gamma^{\mathsf{D}}\,\gamma) \times A^{\mathsf{D}}\,\gamma^{D}\,\alpha & \quad (\gamma^{S} : \Gamma^{\mathsf{S}}\,\gamma\,\gamma^{D}) \times A^{\mathsf{S}}\,\gamma^{S}\,\alpha\,\alpha^{D} \\
(\sigma, t)^{\mathsf{A}}\,\gamma := (\sigma^{\mathsf{A}}\,\gamma, t^{\mathsf{A}}\,\gamma) & (\sigma, t)^{\mathsf{D}}\,\gamma^{D} := (\sigma^{\mathsf{D}}\,\gamma^{D}, t^{\mathsf{D}}\,\gamma^{D}) & (\sigma, t)^{\mathsf{S}}\,\gamma^{S} := (\sigma^{\mathsf{S}}\,\gamma^{S}, t^{\mathsf{S}}\,\gamma^{S}) \\
(\pi_1\,\sigma)^{\mathsf{A}}\,\gamma := \mathsf{proj}_1\,(\sigma^{\mathsf{A}}\,\gamma) & (\pi_1\,\sigma)^{\mathsf{D}}\,\gamma^{D} := \mathsf{proj}_1\,(\sigma^{\mathsf{D}}\,\gamma^{D}) & (\pi_1\,\sigma)^{\mathsf{S}}\,\gamma^{S} := \mathsf{proj}_1\,(\sigma^{\mathsf{S}}\,\gamma^{S}) \\
(\pi_2\,\sigma)^{\mathsf{A}}\,\gamma := \mathsf{proj}_2\,(\sigma^{\mathsf{A}}\,\gamma) & (\pi_2\,\sigma)^{\mathsf{D}}\,\gamma^{D} := \mathsf{proj}_2\,(\sigma^{\mathsf{D}}\,\gamma^{D}) & (\pi_2\,\sigma)^{\mathsf{S}}\,\gamma^{S} := \mathsf{proj}_2\,(\sigma^{\mathsf{S}}\,\gamma^{S}) \\
\mathsf{U}^{\mathsf{A}}\,\gamma := \mathsf{Set} & \mathsf{U}^{\mathsf{D}}\,\gamma^{D}\,T := T \to \mathsf{Set} & \mathsf{U}^{\mathsf{S}}\,\gamma^{S}\,T\,T^{D} := (\alpha : T) \to T^{D}\,\alpha \\
(\mathsf{El}\,a)^{\mathsf{A}}\,\gamma := a^{\mathsf{A}}\,\gamma & (\mathsf{El}\,a)^{\mathsf{D}}\,\gamma^{D}\,\alpha := a^{\mathsf{D}}\,\gamma^{D}\,\alpha & (\mathsf{El}\,a)^{\mathsf{S}}\,\gamma^{S}\,\alpha\,\alpha^{D} := (a^{\mathsf{S}}\,\gamma^{S}\,\alpha = \alpha^{D}) \\
(\Pi\,a\,B)^{\mathsf{A}}\,\gamma := & (\Pi\,a\,B)^{\mathsf{D}}\,\gamma^{D}\,f := & (\Pi\,a\,B)^{\mathsf{S}}\,\gamma^{S}\,f\,f^{D} := (\alpha : a^{\mathsf{A}}\,\gamma) \to \\
\quad (\alpha : a^{\mathsf{A}}\,\gamma) \to B^{\mathsf{A}}\,(\gamma, \alpha) & \quad (\alpha^{D} : a^{\mathsf{D}}\,\gamma^{D}\,\alpha) \to & \quad B^{\mathsf{S}}\,(\gamma^{S}, \mathsf{refl}_{a^{\mathsf{S}}\,\gamma^{S}\,\alpha})\,(f\,\alpha) \\
 & \quad B^{\mathsf{D}}\,(\gamma^{D}, \alpha^{D})\,(f\,\alpha) & \qquad (f^{D}\,(a^{\mathsf{S}}\,\gamma^{S}\,\alpha)) \\
(t \mathbin{@} u)^{\mathsf{A}}\,\gamma := t^{\mathsf{A}}\,\gamma\,(u^{\mathsf{A}}\,\gamma) & (t \mathbin{@} u)^{\mathsf{D}}\,\gamma^{D} := t^{\mathsf{D}}\,\gamma^{D}\,(u^{\mathsf{D}}\,\gamma^{D}) & (t \mathbin{@} u)^{\mathsf{S}}\,\gamma^{S} :=_{u^{\mathsf{S}}\,\gamma^{S}\,\#} t^{\mathsf{S}}\,\gamma^{\mathsf{S}}\,(u^{\mathsf{A}}\,\gamma) \\
(\hat{\Pi}\,T\,B)^{\mathsf{A}}\,\gamma := & (\hat{\Pi}\,T\,B)^{\mathsf{D}}\,\gamma^{D}\,f := & (\hat{\Pi}\,T\,B)^{\mathsf{S}}\,\gamma^{S}\,f\,f^{D} := (\alpha : T) \to \\
\quad (\alpha : T) \to (B\,\alpha)^{\mathsf{A}}\,\gamma & \quad (\alpha : T) \to (B\,\alpha)^{\mathsf{D}}\,\gamma^{D}\,(f\,\alpha) & \quad (B\,\alpha)^{\mathsf{S}}\,\gamma^{S}\,(f\,\alpha)\,(f^{D}\,\alpha) \\
(t \mathbin{\hat{@}} \alpha)^{\mathsf{A}}\,\gamma := t^{\mathsf{A}}\,\gamma\,\alpha & (t \mathbin{\hat{@}} \alpha)^{\mathsf{D}}\,\gamma^{D} := t^{\mathsf{D}}\,\gamma^{D}\,\alpha & (t \mathbin{\hat{@}} \alpha)^{\mathsf{S}}\,\gamma^{S} := t^{\mathsf{S}}\,\gamma^{S}\,\alpha
\end{array}$$

Definition 10 can be explained by columns (see [21, Sections 4 and 6] for more details) or by rows (see [21, Section 7.4]).

We first explain it by columns: the first column ($^{\mathsf{A}}$ components) corresponds to the standard model (set model, metacircular interpretation [4]): contexts are sets, types are families, terms are functions, the universe U is given by Set, function spaces are given by the external function space. The $^{\mathsf{D}}$ column is a logical predicate interpretation, $^{\mathsf{A}}$ and $^{\mathsf{D}}$ together are a unary version of the parametric model for dependent types [7]. Contexts are predicates, types are families of predicates, terms say that the $^{\mathsf{A}}$ interpretation respects the predicates (this is ususally called fundamental lemma of the logical predicate). U is given by predicate space, the predicate at a Π type holds for a function if it respects the predicates. For $\hat{\Pi}$, the predicate is defined pointwise. The last column $^{\mathsf{S}}$ is a modified dependent logical relation which refers to both ^{A} and $^{\mathsf{D}}$. Contexts are binary relations where the second parameter depends on the first one, types are dependent variants of this, terms say that the relation is respected by $^{\mathsf{A}}$ and $^{\mathsf{D}}$, respectively. U is however not relation space, but a function and $(\mathsf{El}\, a)^{\mathsf{S}}$ is the graph of the function a^{S}. Π^{S} for a function again says that the function respects the relation, however we do not simply say

$$(\Pi\, a\, B)^{\mathsf{S}}\, \gamma^S\, f\, f^D := (\alpha : a^{\mathsf{A}}\, \gamma)(\alpha^D : a^{\mathsf{D}}\, \gamma^D\, \alpha)(\alpha^{\mathsf{S}} : (\mathsf{El}\, a)^{\mathsf{S}}\, \gamma^S\, \alpha\, \alpha^D) \to B^{\mathsf{S}}\, \dots,$$

as $(\mathsf{El}\, a)^{\mathsf{S}}\, \gamma^S\, \alpha\, \alpha^D$ is just an equality $a^{\mathsf{S}}\, \gamma^S\, \alpha = \alpha^D$ which we can singleton contract. So we omit α^D and this equality as an input and replace α^D by $a^{\mathsf{S}}\, \gamma^S\, \alpha$ in the definition.

When viewing ADS by rows, we can see that it is a part of the CwF model of type theory [21, Section 7.4]. In the CwF model, a context is given by a CwF. Now, from the category part of the CwF, we only have objects (Γ^{A}), and from the families, we have the families for types Γ^{D} and terms Γ^{S}. Types are the corresponding parts of displayed CwFs, substitutions are parts of CwF morphisms, terms are parts of CwF sections. U is part of the CwF of sets, $\mathsf{El}\, a$ is the part of the discrete displayed CwF coming from a (which is a CwF-morphism from Γ to the CwF of sets). Π is given by a dependent product of displayed CwFs where it is essential that the domain is discrete, $\hat{\Pi}$ is the pointwise direct product.

▶ **Definition 11** (The set signature algebra A). $\mathsf{A} : \mathsf{SignAlg}$ *is given by the first* $^{\mathsf{A}}$ *components of* ADS *(Definition 10), that is,* $\mathsf{Con}^{\mathsf{A}} := \mathsf{Set}$*,* $\mathsf{Ty}^{\mathsf{A}}\, \Gamma := \Gamma \to \mathsf{Set}$*,* $\mathsf{Sub}^{\mathsf{A}}\, \Gamma\, \Delta := \Gamma \to \Delta$*, and so on. There is a morphism from* ADS *to* A *defined by* $-^{\mathsf{A}}$ *at each component, which we also denote by* $-^{\mathsf{A}} : \mathsf{SignMor}\, \mathsf{ADS}\, \mathsf{A}$*.*

▶ **Definition 12** (A model of ETT supports IITs). *A model of ETT supports IITs if for any signature* $(sig, sigc) : \mathsf{Sign}$ *there is a*

$$\mathsf{con}_{sig} : (sig\, \mathsf{ADS})^{\mathsf{A}}$$

and an

$$\mathsf{elim}_{sig} : (\gamma^D : (sig\, \mathsf{ADS})^{\mathsf{D}}\, \mathsf{con}_{sig}) \to (sig\, \mathsf{ADS})^{\mathsf{S}}\, \mathsf{con}_{sig}\, \gamma^D.$$

In other words, for any signature, we have an algebra called con (constructors) and for any displayed algebra over the constructors, we have a section (called the eliminator).

One can check that Definition 12 gives the right notion of constructors and elimination principle for the signatures in Example 8.

▶ **Example 13** (A model of ETT supports natural numbers). For the signature $(nat, natc)$ of natural numbers in Example 8, the type of con_{nat} is

$$\begin{aligned}
&(nat\, \mathsf{ADS})^{\mathsf{A}} = \\
&(\cdot^{\mathsf{ADS}} \rhd^{\mathsf{ADS}} \mathsf{U}^{\mathsf{ADS}} \rhd^{\mathsf{ADS}} \mathsf{El}^{\mathsf{ADS}}\, 0^{\mathsf{ADS}} \rhd^{\mathsf{ADS}} 1^{\mathsf{ADS}} \Rightarrow^{\mathsf{ADS}} \mathsf{El}^{\mathsf{ADS}}\, 1^{\mathsf{ADS}})^{\mathsf{A}} = \\
&\Big(((\cdot \rhd \mathsf{U}) \rhd \mathsf{El}\, (\pi_2\, \mathsf{id})) \rhd (\pi_2\, (\pi_1\, \mathsf{id})) \Rightarrow \mathsf{El}\, (\pi_2\, (\pi_1\, \mathsf{id}))\Big)^{\mathsf{A}} = \\
&\Big(\gamma'' : \big(\gamma' : ((\gamma : \cdot^{\mathsf{A}}) \times \mathsf{U}^{\mathsf{A}}\, \gamma)\big) \times (\mathsf{El}\, (\pi_2\, \mathsf{id}))^{\mathsf{A}}\, \gamma'\Big) \times \Big(\Pi\, (\pi_2\, (\pi_1\, \mathsf{id}))\, (\pi_2\, (\pi_1\, (\pi_1\, \mathsf{id})))\Big)^{\mathsf{A}}\, \gamma'' = \\
&\Big(\gamma'' : \big(\gamma' : ((\gamma : \top) \times \mathsf{Set})\big) \times (\mathsf{proj}_2\, \gamma')\Big) \times \big(\mathsf{proj}_2\, (\mathsf{proj}_1\, \gamma'') \to \mathsf{proj}_2\, (\mathsf{proj}_1\, \gamma'')\big),
\end{aligned}$$

which is a left-nested Σ type isomorphic to its right-nested counterpart

$$(N : \mathsf{Set}) \times \big(N \times (N \to N)\big).$$

Writing $(((\mathsf{tt}, \mathsf{Nat}), \mathsf{zero}), \mathsf{suc})$ for con_{nat}, the type of elim_{nat} computes as follows.

$$\begin{aligned}
&(\gamma^D : (nat\,\mathsf{ADS})^{\mathsf{D}}\,\mathsf{con}_{nat}) \to (nat\,\mathsf{ADS})^{\mathsf{S}}\,\mathsf{con}_{nat}\,\gamma^D = \\
&\Big(\gamma^D : \Big(\big((\cdot \rhd \mathsf{U}) \rhd \mathsf{El}\,(\pi_2\,\mathsf{id})\big) \rhd \big(\pi_2\,(\pi_1\,\mathsf{id})\big) \Rightarrow \mathsf{El}\,\big(\pi_2\,(\pi_1\,\mathsf{id})\big)\Big)^{\mathsf{D}}\,\mathsf{con}_{nat}\Big) \to \\
&\Big(\big((\cdot \rhd \mathsf{U}) \rhd \mathsf{El}\,(\pi_2\,\mathsf{id})\big) \rhd \big(\pi_2\,(\pi_1\,\mathsf{id})\big) \Rightarrow \mathsf{El}\,\big(\pi_2\,(\pi_1\,\mathsf{id})\big)\Big)^{\mathsf{S}}\,\mathsf{con}_{nat}\,\gamma^D = \\
&\Big(\gamma^D : \Big(\big((\cdot \rhd \mathsf{U}) \rhd \mathsf{El}\,(\pi_2\,\mathsf{id})\big) \rhd \big(\pi_2\,(\pi_1\,\mathsf{id})\big) \Rightarrow \mathsf{El}\,\big(\pi_2\,(\pi_1\,\mathsf{id})\big)\Big)^{\mathsf{D}} \\
&\quad \big(((\mathsf{tt}, \mathsf{Nat}), \mathsf{zero}), \mathsf{suc}\big)\Big) \to \\
&\Big(\big((\cdot \rhd \mathsf{U}) \rhd \mathsf{El}\,(\pi_2\,\mathsf{id})\big) \rhd \big(\pi_2\,(\pi_1\,\mathsf{id})\big) \Rightarrow \mathsf{El}\,\big(\pi_2\,(\pi_1\,\mathsf{id})\big)\Big)^{\mathsf{S}}\,\big(((\mathsf{tt}, \mathsf{Nat}), \mathsf{zero}), \mathsf{suc}\big)\,\gamma^D = \\
&\Big(\big(((\mathsf{tt}, N^D), z^D), s^D\big) : \Big(\gamma^{D\prime\prime} : \big(\gamma^{D\prime} : ((\gamma^D : \cdot^{\mathsf{D}}\,\mathsf{tt}) \times \mathsf{U}^{\mathsf{D}}\,\gamma^D\,\mathsf{Nat})\big) \times (\mathsf{El}\,(\pi_2\,\mathsf{id}))^{\mathsf{D}}\,\gamma^{D\prime}\,\mathsf{zero}\Big) \times \\
&\qquad\qquad \Big(\Pi\,\big(\pi_2\,(\pi_1\,\mathsf{id})\big)\,\big(\pi_2\,(\pi_1\,(\pi_1\,\mathsf{id}))\big)\Big)^{\mathsf{D}}\,\gamma^{D\prime\prime}\,\mathsf{suc}\Big) \to \\
&\Big(\big((\cdot \rhd \mathsf{U}) \rhd \mathsf{El}\,(\pi_2\,\mathsf{id})\big) \rhd \big(\pi_2\,(\pi_1\,\mathsf{id})\big) \Rightarrow \mathsf{El}\,\big(\pi_2\,(\pi_1\,\mathsf{id})\big)\Big)^{\mathsf{S}}\,\big(((\mathsf{tt}, \mathsf{Nat}), \mathsf{zero}), \mathsf{suc}\big) \\
&\quad \big(((\mathsf{tt}, N^D), z^D), s^D\big) = \\
&\Big(\big(((\mathsf{tt}, N^D), z^D), s^D\big) : \Big(\gamma^{D\prime\prime} : \big(\gamma^{D\prime} : ((\gamma^D : \cdot^{\mathsf{D}}\,\mathsf{tt}) \times \mathsf{U}^{\mathsf{D}}\,\gamma^D\,\mathsf{Nat})\big) \times (\mathsf{El}\,(\pi_2\,\mathsf{id}))^{\mathsf{D}}\,\gamma^{D\prime}\,\mathsf{zero}\Big) \times \\
&\qquad\qquad \Big(\Pi\,\big(\pi_2\,(\pi_1\,\mathsf{id})\big)\,\big(\pi_2\,(\pi_1\,(\pi_1\,\mathsf{id}))\big)\Big)^{\mathsf{D}}\,\gamma^{D\prime\prime}\,\mathsf{suc}\Big) \to \\
&\Big(\gamma^{S\prime\prime} : \big(\gamma^{S\prime} : ((\gamma^S : \cdot^{\mathsf{S}}\,\mathsf{tt}\,\mathsf{tt}) \times \mathsf{U}^{\mathsf{S}}\,\gamma^S\,\mathsf{Nat}\,N^D)\big) \times (\mathsf{El}\,(\pi_2\,\mathsf{id}))^{\mathsf{S}}\,\gamma^{S\prime}\,\mathsf{zero}\,z^D\Big) \times \\
&\Big(\Pi\,\big(\pi_2\,(\pi_1\,\mathsf{id})\big)\,\big(\pi_2\,(\pi_1\,(\pi_1\,\mathsf{id}))\big)\Big)^{\mathsf{S}}\,\gamma^{S\prime\prime}\,\mathsf{suc}\,s^D = \\
&\Big(\big(((\mathsf{tt}, N^D), z^D), s^D\big) : \Big(\gamma^{D\prime\prime} : \big(\gamma^{D\prime} : ((\gamma^D : \top) \times (\mathsf{Nat} \to \mathsf{Set}))\big) \times \mathsf{proj}_2\,\gamma^{D\prime}\,\mathsf{zero}\Big) \times \\
&\qquad\qquad \big(\mathsf{proj}_2\,(\mathsf{proj}_1\,\gamma^{D\prime\prime})\,n \to \mathsf{proj}_2\,(\mathsf{proj}_1\,\gamma^{D\prime\prime})\,(\mathsf{suc}\,n)\big)\Big) \to \\
&\Big(\gamma^{S\prime\prime} : \big(\gamma^{S\prime} : ((\gamma^S : \top) \times ((n : \mathsf{Nat}) \to N^D\,n))\big) \times \mathsf{proj}_2\,\gamma^{S\prime}\,\mathsf{zero} = z^D\Big) \times \\
&\Big(\big(n : \mathsf{Nat}\big) \to \mathsf{proj}_2\,(\mathsf{proj}_1\,(\mathsf{proj}_1\,\gamma^{S\prime\prime}))\,(\mathsf{suc}\,n) = s^D\,\big(\mathsf{proj}_2\,(\mathsf{proj}_1\,(\mathsf{proj}_1\,\gamma^{S\prime\prime}))\,n\big)\Big)
\end{aligned}$$

This is again a left-nested version of the expected elimination principle

$$\begin{aligned}
&(N^D : \mathsf{Nat} \to \mathsf{Set})(z^D : N^D\,\mathsf{zero})\big(s^D : (n : \mathsf{Nat}) \to N^D\,n \to N^D\,(\mathsf{suc}\,n)\big) \to \\
&\big(N^S : (n : \mathsf{Nat}) \to N^D\,n\big) \times (N^S\,\mathsf{zero} = z^D) \times \big((n : \mathsf{Nat}) \to N^S\,(\mathsf{suc}\,n) = s^D\,(N^S\,n)\big)
\end{aligned}$$

▶ Remark 14. The computation rules of the elimination principle are only expected up to the internal equality type, but as we work with a model of ETT, we also get them as definitional equalities by equality reflection.

3 Constructing all IITs from the Theory of IIT Signatures

In the previous section, using the notions of signature algebras and signature morphisms, we defined IIT signatures and what it means for a model of ETT to support all IITs. In this section we show that if a model of ETT supports the theory of IIT signatures, then it supports all IITs. Using the Church encoding of Definition 6, every model of ETT can describe ITT signatures. In contrast, in Definition 15, we will require existence of an initial signature algebra.

The contents of this section are an adjustment of [21, Sections 4 and 6] to our setting.

▶ **Definition 15.** *A model of ETT supports the theory of IIT signatures if there is a signature algebra* $\mathsf{I} : \mathsf{SignAlg}$ *equipped with a unique morphism* $[\![-]\!]_M : \mathsf{SignMor}\,\mathsf{I}\,M$ *into any algebra* M*. Sometimes we omit the subscript* $_M$*. We call* I *the syntax or initial algebra, the morphism* $[\![-]\!]$ *is called recursor.*

▶ **Definition 16** (Syntactic signatures)**.** *In a model of ETT supporting the theory of ITT signatures, we call elements of* $\mathsf{Con}^{\mathsf{I}}$ *syntactic signatures.*

One may wonder what is the relationship between the two notion of signatures.

▶ **Lemma 17.** *In a model of ETT supporting the theory of ITT signatures, signatures and syntactic signatures are isomorphic.*

Proof. We can turn a $(sig, sigc) : \mathsf{Sign}$ into $\mathsf{Con}^{\mathsf{I}}$ by $sig\,\mathsf{I}$ and an $\Omega : \mathsf{Con}^{\mathsf{I}}$ into a Sign by $\big(\lambda M.[\![\Omega]\!]_M, \lambda M\,N\,f.\big(f\,[\![\Omega]\!]_M = (f \circ [\![-]\!]_M)\,\Omega = [\![\Omega]\!]_N\big)\big)$ where the equality proof in the second component comes from uniqueness of the recursor (we have to define composition of morphisms $\circ$ for this). The compositions of these two maps are the identities: $(sig, sigc)$ is mapped to $(\lambda M.[\![sig\,I]\!]_M, \dots) = (\lambda M.[\![-]\!]_M\,(sig\,I), \dots)$ which is equal to $(\lambda M.sig\,M, \dots)$ because of $sigc$; Ω is mapped to $[\![\Omega]\!]_{\mathsf{I}} = \Omega$ by uniqueness of $[\![-]\!]$. ◀

We will define the term signature algebra by which we obtain the constructors con for any IIT signature. Then we will define another signature algebra which provides the eliminator. Before doing these, we illustrate the idea of both constructions on natural numbers.

▶ **Example 18.** For natural numbers, we will define the constructors con as the following natural number algebra $(\mathsf{Nat}, \mathsf{zero}, \mathsf{suc})$. We write variable names instead of de Bruijn indices for readability.

$$\begin{aligned}\mathsf{Nat} &:= \mathsf{Tm}^{\mathsf{I}}\,(\cdot \rhd N : \mathsf{U} \rhd z : \mathsf{El}\,N \rhd s : N \Rightarrow \mathsf{El}\,N)\,(\mathsf{El}\,N)\\ \mathsf{zero} &:= z\\ \mathsf{suc} &:= \lambda t.(s\,@\,t)\end{aligned}$$

Natural numbers are simply I-terms of type $\mathsf{El}\,N$ in the context which is the syntactic signature for natural numbers. In this context, the only way to define a term of type $\mathsf{El}\,N$ is to use z and s, corresponding to the zero and suc constructors.

To define the action of the eliminator on a natural number $n : \mathsf{Nat}$, let's look at the type of the displayed algebra interpretation of the number:

$$[\![n]\!]_{\mathsf{ADS}}{}^{\mathsf{D}} : (\gamma^D : [\![\cdot \rhd N : \mathsf{U} \rhd z : \mathsf{El}\,N \rhd s : N \Rightarrow \mathsf{El}\,N]\!]^{\mathsf{D}}\,\mathsf{con}) \to [\![\mathsf{El}\,N]\!]^{\mathsf{D}}\,([\![n]\!]^{\mathsf{A}}\,\mathsf{con})$$

This says that for a displayed algebra $\gamma^D = (N^D, z^D, s^D)$ over con (i.e. the motives and methods of the eliminator), we get a witness of the predicate $[\![\mathsf{El}\,N]\!]^{\mathsf{D}} = N^D$ at the algebra interpretation of n. This is not yet good, as we would like to get $N^D\,n$ instead of $N^D\,([\![n]\!]^{\mathsf{A}}\,\mathsf{con})$ as a result. However, interpretation into the term signature algebra will imply that $n = [\![n]\!]^{\mathsf{A}}\,\mathsf{con}$.

▶ **Definition 19** (Term signature algebra $\mathsf{IC}_{\_}$). *For an $\Omega : \mathsf{Con}^{I}$, we define $\mathsf{IC}_{\Omega} : \mathsf{SignAlg}$ which we call the term signature algebra. It is equipped with a morphism $\_^{\mathsf{I}} : \mathsf{SignMor}\,(\mathsf{IC}_{\Omega})\,\mathsf{I}$. We define IC_{Ω} by listing its components Con, Ty, Sub, and so on, one per row. Each component has two parts denoted by $^{\mathsf{I}}$ and $^{\mathsf{C}}$. The $^{\mathsf{I}}$ part just reuses the corresponding components from I, and thus the morphism $\_^{\mathsf{I}}$ is defined as the obvious projection. We omit the equality components, as they come from UIP or are trivial. We also omit the components for terms and substitutions as their $^{\mathsf{C}}$ parts consist of uninformative equational reasoning.*

$$\begin{array}{ll}
\Gamma^{\mathsf{I}} : \mathsf{Con}^{\mathsf{I}} & \Gamma^{\mathsf{C}} : \mathsf{Sub}^{\mathsf{I}}\,\Omega\,\Gamma^{\mathsf{I}} \to [\![\Gamma]\!]_{\mathsf{A}} \\
A^{\mathsf{I}} : \mathsf{Ty}^{\mathsf{I}}\,\Gamma^{\mathsf{I}} & A^{\mathsf{C}} : (\nu : \mathsf{Sub}^{\mathsf{I}}\,\Omega\,\Gamma^{\mathsf{I}}) \to \mathsf{Tm}^{\mathsf{I}}\,\Omega\,(A^{\mathsf{I}}[\nu]) \to [\![A]\!]_{\mathsf{A}}\,(\Gamma^{\mathsf{C}}\,\nu) \\
\sigma^{\mathsf{I}} : \mathsf{Sub}^{\mathsf{I}}\,\Gamma^{\mathsf{I}}\,\Delta^{\mathsf{I}} & \sigma^{\mathsf{C}} : \Delta^{\mathsf{C}}\,(\sigma^{\mathsf{I}} \circ \nu) = [\![\sigma]\!]_{\mathsf{A}}\,(\Gamma^{\mathsf{C}}\,\nu) \\
t^{\mathsf{I}} : \mathsf{Tm}^{\mathsf{I}}\,\Gamma^{\mathsf{I}}\,A^{\mathsf{I}} & t^{\mathsf{C}} : A^{\mathsf{C}}\,\nu\,(t^{\mathsf{I}}[\nu]) = [\![t]\!]_{\mathsf{A}}\,(\Gamma^{\mathsf{C}}\,\nu) \\
(A[\sigma])^{\mathsf{I}} := A^{\mathsf{I}}[\sigma^{\mathsf{I}}]^{\mathsf{I}} & (A[\sigma])^{\mathsf{C}}\,\nu\,t := A^{\mathsf{C}}\,(\sigma^{\mathsf{I}} \circ \nu)\,t \\
\cdot^{\mathsf{I}} := \cdot^{\mathsf{I}} & \cdot^{\mathsf{C}}\;\nu := \mathsf{tt} \\
(\Gamma \rhd A)^{\mathsf{I}} := \Gamma^{\mathsf{I}} \rhd^{\mathsf{I}} A^{\mathsf{I}} & (\Gamma \rhd A)^{\mathsf{C}}\,\nu := (\Gamma^{\mathsf{C}}\,(\pi_1\,\nu), A^{\mathsf{C}}\,(\pi_1\,\nu)\,(\pi_2\,\nu)) \\
\mathsf{U}^{\mathsf{I}} := \mathsf{U}^{\mathsf{I}} & \mathsf{U}^{\mathsf{C}}\nu\,a := \mathsf{Tm}^{\mathsf{I}}\,\Omega\,(\mathsf{El}^{\mathsf{I}}\,a) \\
(\mathsf{El}\,a)^{\mathsf{I}} := \mathsf{El}^{\mathsf{I}}\,a^{\mathsf{I}} & (\mathsf{El}\,a)^{\mathsf{C}}\,\nu\,t :=_{a^{\mathsf{C}}\,\nu\#}\,t \\
(\Pi\,a\,B)^{\mathsf{I}} := \Pi^{\mathsf{I}}\,a^{\mathsf{I}}\,B^{\mathsf{I}} & (\Pi\,a\,B)^{\mathsf{C}}\,\nu\,t := \lambda\alpha.B^{\mathsf{C}}\,(\nu,_{a^{\mathsf{C}}\,\nu\#}\,\alpha)\,(t\,@\,_{a^{\mathsf{C}}\,\nu\#}\alpha) \\
(\hat{\Pi}\,T\,B)^{\mathsf{I}} := \hat{\Pi}^{\mathsf{I}}\,T\,B^{\mathsf{I}} & (\hat{\Pi}\,T\,B)^{\mathsf{C}}\,\nu\,t := \lambda\alpha.(B\,\alpha)^{\mathsf{C}}\,\nu\,(t\,\hat{@}\,\alpha)
\end{array}$$

▶ **Example 20.** Now, given a syntactic signature $\Omega : \mathsf{Con}^{\mathsf{I}}$, we get the constructors as an Ω-algebra by $\omega := ([\![\Omega]\!]_{\mathsf{IC}_{\Omega}})^{\mathsf{C}}\,\mathsf{id}^{\mathsf{I}} : [\![\Omega]\!]_{\mathsf{A}}$. If Ω is the syntactic signature for natural numbers, we get the constructors as in Example 18.

An $a : \mathsf{Tm}^{\mathsf{I}}\,\Omega\,\mathsf{U}$ is a sort term for the syntactic signature Ω. If Ω is the syntactic signature for natural numbers, a can only be N (1 as a de Bruijn index). If Ω is the syntactic signature for Con–Ty (Example 4), a can be Con, $Ty\,@\,empty$, $Ty\,@(ext\,@\,empty\,@(U\,@\,empty))$, and so on. In any case, for such an a, we obtain $([\![a]\!]_{\mathsf{IC}_{\Omega}})^{\mathsf{C}}\,\mathsf{id}^{\mathsf{I}} : \mathsf{Tm}^{\mathsf{I}}\,\Omega\,(\mathsf{El}\,a) = [\![a]\!]_{\mathsf{A}}\,\omega$. That is, the algebra interpretation of a sort term at the constructors is equal to terms of that sort.

A $t : \mathsf{Tm}^{\mathsf{I}}\,\Omega\,(\mathsf{El}\,a)$ is a term of a sort type a constructed using the constructors in Ω. For natural numbers, such a t can only be s applied iteratively to z. For such a t, we obtain $([\![t]\!]_{\mathsf{IC}_{\Omega}})^{\mathsf{C}}\,\mathsf{id}^{\mathsf{I}} : (t = [\![t]\!]_{\mathsf{A}}\,\omega)$. That is, a constructor term is equal to its algebra interpretation at the constructors. This is exactly the equation needed at the end of Example 18.

▶ **Definition 21** (Eliminator signature algebra $\mathsf{IE}_{\_}$). *Given an $\Omega : \mathsf{Con}^{\mathsf{I}}$, we use the abbreviation $\omega := [\![\Omega]\!]_{\mathsf{IC}_{\Omega}}\,\mathsf{id}^{\mathsf{I}}$ as in Example 20. Assuming an $\omega^{D} : ([\![\Omega]\!]_{\mathsf{ADS}})^{\mathsf{D}}\,\omega$, we define the signature algebra $\mathsf{IE}_{\omega^{\mathsf{D}}}$. It is equipped with a morphism $\_^{\mathsf{I}} : \mathsf{SignMor}\,\mathsf{IE}_{\omega^{D}}\,\mathsf{I}$. We define $\mathsf{IE}_{\omega^{D}}$ by listing its components Con, Ty, Sub, and so on, one per row. Each component has two parts denoted by $^{\mathsf{I}}$ and $^{\mathsf{E}}$. The $^{\mathsf{I}}$ part just reuses the corresponding components of I, thus the morphism $\_^{\mathsf{I}}$ is defined as the obvious projection. We omit the equality components, as they come from UIP or are trivial. We also omit the components for terms and substitutions as their $^{\mathsf{E}}$ parts are uninformative equational reasonings.*

$$\begin{array}{ll}
\Gamma^{\mathsf{I}} : \mathsf{Con}^{\mathsf{I}} & \Gamma^{\mathsf{E}} : (\nu : \mathsf{Sub}^{\mathsf{I}}\,\Omega\,\Gamma^{\mathsf{I}}) \to [\![\Gamma]\!]^{\mathsf{S}}\,([\![\nu]\!]^{\mathsf{A}}\,\omega)\,([\![\nu]\!]^{\mathsf{D}}\,\omega^{D}) \\
A^{\mathsf{I}} : \mathsf{Ty}^{\mathsf{I}}\,\Gamma^{\mathsf{I}} & A^{\mathsf{E}} : (\nu : \mathsf{Sub}^{\mathsf{I}}\,\Omega\,\Gamma^{\mathsf{I}})(t : \mathsf{Tm}^{\mathsf{I}}\,\Omega\,(A^{\mathsf{I}}[\nu])) \to \\
& \quad [\![A]\!]^{\mathsf{S}}\,(\Gamma^{\mathsf{E}}\,\nu)\,([\![t]\!]^{\mathsf{A}}\,\omega)\,([\![t]\!]^{\mathsf{D}}\,\omega^{D}) \\
\sigma^{\mathsf{I}} : \mathsf{Sub}^{\mathsf{I}}\,\Gamma^{\mathsf{I}}\,\Delta^{\mathsf{I}} & \sigma^{\mathsf{E}} : \Delta^{\mathsf{E}}\,(\sigma^{\mathsf{I}} \circ \nu) = [\![\sigma]\!]^{\mathsf{S}}\,(\Gamma^{\mathsf{E}}\,\nu)
\end{array}$$

$$\begin{array}{ll}
t^{\mathsf{I}} : \mathsf{Tm}^{\mathsf{I}}\,\mathit{\Gamma}^{\mathsf{I}}\,A^{\mathsf{I}} & t^{\mathsf{E}} : A^{\mathsf{E}}\,\nu\,(t^{\mathsf{I}}[\nu]) = \llbracket t \rrbracket^{\mathsf{S}}\,(\mathit{\Gamma}^{\mathsf{E}}\,\nu) \\
(A[\sigma])^{\mathsf{I}} := A^{\mathsf{I}}[\sigma^{\mathsf{I}}]^{\mathsf{I}} & (A[\sigma])^{\mathsf{E}}\,\nu\,t := A^{\mathsf{E}}\,(\sigma^{\mathsf{I}} \circ \nu)\,t \\
\cdot^{\mathsf{I}} := \cdot^{\mathsf{I}} & \cdot^{\mathsf{E}}\ \nu := \mathsf{tt} \\
(\mathit{\Gamma} \rhd A)^{\mathsf{I}} := \mathit{\Gamma}^{\mathsf{I}} \rhd^{\mathsf{I}} A^{\mathsf{I}} & (\mathit{\Gamma} \rhd A)^{\mathsf{E}}\,\nu := (\mathit{\Gamma}^{\mathsf{E}}\,(\pi_1\,\nu), A^{\mathsf{E}}\,(\pi_1\,\nu)\,(\pi_2\,\nu)) \\
\mathsf{U}^{\mathsf{I}} := \mathsf{U}^{\mathsf{I}} & \mathsf{U}^{\mathsf{E}}\nu\,a := \lambda\alpha._{\llbracket\alpha\rrbracket^{\mathsf{C}}\,\mathsf{id}\#}\big(\llbracket_{\llbracket\alpha\rrbracket^{\mathsf{C}}\,\mathsf{id}\#}\alpha\rrbracket^{\mathsf{D}}\,\omega^{D}\big) \\
(\mathsf{El}\,a)^{\mathsf{I}} := \mathsf{El}^{\mathsf{I}}\,a^{\mathsf{I}} & (\mathsf{El}\,a)^{\mathsf{E}}\,\nu\,t := \big(\llbracket a\rrbracket^{\mathsf{S}}\,(\mathit{\Gamma}^{\mathsf{E}}\,\nu)\,(\llbracket t\rrbracket^{\mathsf{A}}\,\omega) \overset{\llbracket t\rrbracket^{\mathsf{C}}\,\mathsf{id}}{=} \llbracket a\rrbracket^{\mathsf{S}}\,(\mathit{\Gamma}^{\mathsf{E}}\,\nu)\,t \overset{a^{\mathsf{E}}\,\nu}{=} \llbracket t\rrbracket^{\mathsf{D}}\,\omega^{D}\big) \\
(\Pi\,a\,B)^{\mathsf{I}} := \Pi^{\mathsf{I}}\,a^{\mathsf{I}}\,B^{\mathsf{I}} & (\Pi\,a\,B)^{\mathsf{E}}\,\nu\,t := \\
 & \quad \lambda\alpha._{\llbracket\alpha\rrbracket^{\mathsf{C}}\,\mathsf{id}\#}\big(B^{\mathsf{E}}\,(\nu,_{\llbracket a\rrbracket^{\mathsf{C}}\,\mathsf{id},\llbracket\nu\rrbracket^{\mathsf{C}}\,\mathsf{id}\#}\,\alpha)\,(t\,@_{\llbracket a\rrbracket^{\mathsf{C}}\,\mathsf{id},\llbracket\nu\rrbracket^{\mathsf{C}}\,\mathsf{id}\#}u)\big) \\
(\hat{\Pi}\,T\,B)^{\mathsf{I}} := \hat{\Pi}^{\mathsf{I}}\,T\,B^{\mathsf{I}} & (\hat{\Pi}\,T\,B)^{\mathsf{E}}\,\nu\,t := \lambda\alpha.(B\,\alpha)^{\mathsf{E}}\,\nu\,(t\,\hat{@}\,\alpha)
\end{array}$$

▶ **Example 22.** Given the assumptions $\mathit{\Omega}$, ω^{D} of IE, we obtain the eliminator by $\llbracket\mathit{\Omega}\rrbracket_{\mathsf{IE}_{\omega^{\mathsf{D}}}}\,\mathsf{id}^{\mathsf{I}} : \llbracket\mathit{\Omega}\rrbracket^{\mathsf{S}}\,\omega\,\omega^{D}$. The eliminator is a section of the displayed algebra ω^{D}, that is, a dependent function together with equalities witnessing that all the operations are preserved. If $\mathit{\Omega}$ is the syntactic signature for natural numbers, we get the eliminator of Example 18.

For a sort term $a : \mathsf{Tm}^{\mathsf{I}}\,\mathit{\Omega}\,\mathsf{U}$, the interpretation $(\llbracket a\rrbracket_{\mathsf{IE}_{\omega^{D}}})^{\mathsf{E}}\,\mathsf{id}$ says that $(\lambda\alpha.\llbracket\alpha\rrbracket^{\mathsf{D}}\,\omega^{D}) = \llbracket a\rrbracket^{\mathsf{S}}\,(\llbracket\mathit{\Omega}\rrbracket^{\mathsf{E}}\,\mathsf{id})$, that is, the function for the sort a in the eliminator section is the displayed algebra interpretation at ω^{D} (motives and methods). For natural numbers, this is the same as $\big(\lambda n.\llbracket n\rrbracket^{D}\,(N^{D}, z^{D}, s^{D})\big) = \big(\lambda n.\mathsf{elimNat}\,(N^{D}, z^{D}, s^{D})\,n)\big)$.

The interpretation of a constructor term $t : \mathsf{Tm}^{\mathsf{I}}\,\mathit{\Omega}\,(\mathsf{El}\,a)$ is uninteresting as it provides an equality between two different equality proofs of the computation (β) rule for t.

▶ **Theorem 23.** *If a model of ETT supports the theory of IIT signatures, then it supports all IITs.*

Proof. For a signature $(\mathit{sig}, \mathit{sigc})$, we define constructors as

$$\mathsf{con}_{\mathit{sig}} := (\llbracket \mathit{sig}\,\mathsf{I}\rrbracket_{\mathsf{IC}_{\mathit{sig}\,\mathsf{I}}})^{\mathsf{C}}\,\mathsf{id}^{\mathsf{I}} : (\mathit{sig}\,\mathsf{ADS})^{\mathsf{A}}$$

This typechecks as $\llbracket \mathit{sig}\,\mathsf{I}\rrbracket_{\mathsf{A}} = \llbracket - \rrbracket_{\mathsf{A}}\,(\mathit{sig}\,\mathsf{I}) \overset{\mathit{sigc}}{=} \mathit{sig}\,\mathsf{A} = (\mathit{sig}\,\mathsf{ADS})^{\mathsf{A}}$. We define the eliminator by and an

$$\mathsf{elim}_{\mathit{sig}}\,\gamma^{D} := (\llbracket \mathit{sig}\,\mathsf{I}\rrbracket_{\mathsf{IE}_{\gamma^{D}}})^{\mathsf{E}}\,\mathsf{id}^{\mathsf{I}} : (\mathit{sig}\,\mathsf{ADS})^{\mathsf{S}}\,\mathsf{con}_{\mathit{sig}}\,\gamma^{D}.$$

This typechecks firstly because the type of γ^{D} matches the type of the parameter of IE:

$$(\mathit{sig}\,\mathsf{ADS})^{\mathsf{D}}\,\mathsf{con}_{\mathit{sig}} \overset{\mathit{sigc}}{=} (\llbracket - \rrbracket_{\mathsf{ADS}}\,(\mathit{sig}\,\mathsf{I}))^{\mathsf{D}}\,\mathsf{con}_{\mathit{sig}} = (\llbracket \mathit{sig}\,\mathsf{I}\rrbracket_{\mathsf{ADS}})^{\mathsf{D}}\,\mathsf{con}_{\mathit{sig}},$$

and the result also has the correct type:

$$\llbracket \mathit{sig}\,\mathsf{I}\rrbracket^{\mathsf{S}}\,\mathsf{con}_{\mathit{sig}}\,\gamma^{D} = (\llbracket - \rrbracket_{\mathsf{ADS}}\,(\mathit{sig}\,\mathsf{I}))^{\mathsf{S}}\,\mathsf{con}_{\mathit{sig}}\,\gamma^{D} \overset{\mathit{sigc}}{=} (\mathit{sig}\,\mathsf{ADS})^{\mathsf{S}}\,\mathsf{con}_{\mathit{sig}}\,\gamma^{D}. \qquad ◀$$

▶ Remark 24. In the above proof, we crucially relied on the *sigc* property to define the constructors (and the eliminator). This is why the simple Church encoding of signatures is not sufficient.

4 Constructing the Theory of IIT Signatures

In this section we show that any model of ETT which supports indexed W-types also supports the theory of signatures, and as a consequence of Theorem 23, all IITs. For this, we work in the internal language of a model of ETT supporting indexed W-types [3]. Indexed

W-types correspond to the usual notion of (possibly mutual) indexed inductive types. We use Agda-style notation to define such inductive families: we list the sorts and constructors and use pattern matching when eliminating from them. For an encoding of mutual inductive families as indexed W-types, see e.g. [23].

We construct the theory of IIT signatures in the following steps:

1. We view the theory of signatures as a type theory, and we define its untyped syntax as mutual inductive types together with typing judgments given by inductive relations on the untyped syntax. Then the syntax $\mathsf{I} : \mathsf{SignAlg}$ is constructed using those untyped terms for which the typing relation holds.
2. We construct $[\![-]\!] : \mathsf{SignMor}\,\mathsf{I}\,M$ for arbitrary $M : \mathsf{SignAlg}$, by:
   a. defining a relation $- \sim -$ between the well-typed syntax and a given signature algebra. The idea is that given a syntactic context Γ and a semantic context Γ^M of the signature algebra M, we have $\Gamma \sim \Gamma^M$ if and only if $[\![\Gamma]\!] = \Gamma^M$, and similarly for types, terms, and substitutions;
   b. showing that this relation is functional and thus obtaining a morphism.
3. Proving the uniqueness of this morphism by showing that any morphism $f : \mathsf{SignMor}\,\mathsf{I}\,M$ satisfies the relation. For example, for any syntactic context Γ we have $\Gamma \sim f\,\Gamma$.

The next sections detail each of these steps.

4.1 Syntax

The goal is to define the syntactic signature algebra where contexts are pairs of a precontext together with a well-formedness proof, and similarly for types, terms and substitutions.

Crucially, we do not have conversion relations for typed syntax, nor do we need to use quotients when constructing the syntax. This is possible because there are no β-rules in the theory of signatures. Hence, we consider only normal terms in the untyped syntax, and define weakening and substitution by recursion. Avoiding quotients is important for two reasons. First, it greatly simplifies formalisation. Second, we aim to reduce the theory of signatures only to inductive types, thus making Theorem 57 stronger.

Now we present the definition of the untyped syntax and the associated typing judgments.

4.1.1 Untyped Syntax and its Properties

▶ **Definition 25** (Untyped syntax). *The untyped syntax is defined as the following inductive datatype.*

(1) Substitution calculus

$\mathsf{Con}^\mathsf{p} \quad : \mathsf{Set}$

$\mathsf{Ty}^\mathsf{p} \quad : \mathsf{Set}$

$\mathsf{Sub}^\mathsf{p} \quad : \mathsf{Set}$

$\mathsf{Tm}^\mathsf{p} \quad : \mathsf{Set}$

$\cdot^\mathsf{p} \quad : \mathsf{Con}^\mathsf{p}$

$\epsilon^\mathsf{p} \quad : \mathsf{Sub}^\mathsf{p}$

$- \rhd^\mathsf{p} - : \mathsf{Con}^\mathsf{p} \to \mathsf{Ty}^\mathsf{p} \to \mathsf{Con}^\mathsf{p}$

$- ,^\mathsf{p} - \quad : \mathsf{Sub}^\mathsf{p} \to \mathsf{Tm}^\mathsf{p} \to \mathsf{Sub}^\mathsf{p}$

$\mathsf{var}^\mathsf{p} \quad : \mathbb{N} \to \mathsf{Tm}^\mathsf{p}$

(2) Universe

$$\begin{aligned}
&\mathsf{U}^{\mathsf{p}} && : \mathsf{Ty}^{\mathsf{p}} \\
&\mathsf{El}^{\mathsf{p}} && : \mathsf{Tm}^{\mathsf{p}} \to \mathsf{Ty}^{\mathsf{p}}
\end{aligned}$$

(3) Inductive parameters

$$\begin{aligned}
&\Pi^{\mathsf{p}} && : \mathsf{Tm}^{\mathsf{p}} \to \mathsf{Ty}^{\mathsf{p}} \to \mathsf{Ty}^{\mathsf{p}} \\
&- @^{\mathsf{p}} - && : \mathsf{Tm}^{\mathsf{p}} \to \mathsf{Tm}^{\mathsf{p}} \to \mathsf{Tm}^{\mathsf{p}}
\end{aligned}$$

(4) External parameters

$$\begin{aligned}
&\hat{\Pi}^{\mathsf{p}} && : (T : \mathsf{Set}) \to (T \to \mathsf{Ty}^{\mathsf{p}}) \to \mathsf{Ty}^{\mathsf{p}} \\
&\tilde{\Pi}^{\mathsf{p}} && : (T : \mathsf{Set}) \to (T \to \mathsf{Tm}^{\mathsf{p}}) \to \mathsf{Tm}^{\mathsf{p}} \\
&- \hat{\tilde{@}} - && : \mathsf{Tm}^{\mathsf{p}} \to (\alpha : T) \to \mathsf{Tm}^{\mathsf{p}}
\end{aligned}$$

(5) Default value

$$\mathsf{err}^{\mathsf{p}} \quad : \mathsf{Tm}^{\mathsf{p}}$$

Variables are modeled as de Bruijn indices, i.e. as natural numbers pointing to a position in the context. We use the additional default constructor $\mathsf{err}^{\mathsf{p}} : \mathsf{Tm}^{\mathsf{p}}$ in case of error (ill-scoped substitution). The typing judgments will not mention $\mathsf{err}^{\mathsf{p}}$. The main interest of $\mathsf{err}^{\mathsf{p}}$ is that it behaves like a closed term (which the theory of signatures lacks), in the sense that it is invariant under substitution. This makes expected equalities about substitution true even in the ill-typed case, thus reducing the number of hypotheses for the corresponding lemmas (see Lemma 32).

We will define substitutions $-[-]$ of types and terms recursively.

Note that $(\Pi^{\mathsf{p}}\, A\, B)[\sigma]$ should be defined as $\Pi^{\mathsf{p}}\, (A[\sigma])\, (B[\mathsf{wk}_0\, \sigma\ ,^{\mathsf{p}}\ \mathsf{var}^{\mathsf{p}}\, 0])$, and thus we need to define wk_0, the weakening of substitutions. The basic idea is to increment the de Bruijn indices of all the variables. Actually, this is not so simple because of the Π^{p} type: we want to define $\mathsf{wk}_0\, (\Pi^{\mathsf{p}}\, A\, B)$ as the Π type of the weakening of A and B, but here, B must be weakened with respect to the second last variable of the context, rather than the last one. For this reason, we need to generalise the weakening as occuring anywhere in the context.

▶ **Definition 26** (Untyped weakening). *We define untyped weaking recursively on terms by the following functions.*

$$\begin{aligned}
&\mathsf{wk}_n : \mathsf{Ty}^{\mathsf{p}} \to \mathsf{Ty}^{\mathsf{p}} \\
&\mathsf{wk}_n : \mathsf{Tm}^{\mathsf{p}} \to \mathsf{Tm}^{\mathsf{p}} \\
&\mathsf{wk}_0 : \mathsf{Sub}^{\mathsf{p}} \to \mathsf{Sub}^{\mathsf{p}}
\end{aligned}$$

The natural number n specifies at which position of the context the weakening occurs. Here, wk_0 *weakens with respect to the last variable.*

Later, in Lemma 36, we show that weakening preserves typing. Stating a typing rule for this operation requires weakening at the middle of a context. This is why we define pairs of untyped contexts, which should be thought of as a splitting of a context at some position. We call the second context a telescope over the first one.

▶ **Definition 27** (Untyped telescopes). *An untyped telescope is given simply by a* $\mathsf{Con}^{\mathsf{p}}$.

▶ **Definition 28** (Merging of a context and a telescope).

$$\begin{aligned}
&-;- && : \mathsf{Con}^{\mathsf{p}} \to \mathsf{Con}^{\mathsf{p}} \to \mathsf{Con}^{\mathsf{p}} \\
&\Gamma; \cdot && := \Gamma \\
&\Gamma; (\Delta \rhd^{\mathsf{p}} A) && := (\Gamma; \Delta) \rhd^{\mathsf{p}} A
\end{aligned}$$

▶ **Definition 29** (Weakening for telescopes). *Weakening for telescopes is defined pointwise.* $\|\Gamma\|$ *denotes the length of the context* Γ*.*

$$
\begin{aligned}
&\mathsf{wk}_0 && : \mathsf{Con}^\mathsf{p} \to \mathsf{Con}^\mathsf{p}\\
&\mathsf{wk}_0\ \cdot^\mathsf{p} && := \cdot^\mathsf{p}\\
&\mathsf{wk}_0\ (\Delta\ \rhd^\mathsf{p} A) && := \mathsf{wk}_0\ \Delta\ \rhd^\mathsf{p}\ \mathsf{wk}_{\|\Delta\|}\ A
\end{aligned}
$$

This will be used to give typing rules for telescopes in Definition 35.

▶ **Definition 30** (Untyped unary substitution). *We define single substitution by recursion on the presyntax:*

$$
\begin{aligned}
&-[-:=-] : \mathsf{Ty}^\mathsf{p} \to \mathbb{N} \to \mathsf{Tm}^\mathsf{p} \to \mathsf{Ty}^\mathsf{p}\\
&-[-:=-] : \mathsf{Tm}^\mathsf{p} \to \mathbb{N} \to \mathsf{Tm}^\mathsf{p} \to \mathsf{Tm}^\mathsf{p}
\end{aligned}
$$

This is enough to state the typing judgments: indeed, the typing rule for application involves only a unary substitution.

However, to construct the syntax as a signature algebra, we need to define parallel substitutions:

▶ **Definition 31** (Untyped substitution calculus).

$$
\begin{aligned}
&-[-] && : \mathsf{Ty}^\mathsf{p} \to \mathsf{Sub}^\mathsf{p} \to \mathsf{Ty}^\mathsf{p}\\
&-[-] && : \mathsf{Tm}^\mathsf{p} \to \mathsf{Sub}^\mathsf{p} \to \mathsf{Tm}^\mathsf{p}\\
&-\circ- && : \mathsf{Sub}^\mathsf{p} \to \mathsf{Sub}^\mathsf{p} \to \mathsf{Sub}^\mathsf{p}
\end{aligned}
$$

These can be defined either by iterating unary substitutions, or by recursion on untyped syntax: the two ways yield provably equal definitions. In the following, we assume that they are defined by recursion. We also make use of the following definition:

$$
\begin{aligned}
&\mathsf{keep} : \mathsf{Sub}^\mathsf{p} \to \mathsf{Sub}^\mathsf{p}\\
&\qquad := \lambda\sigma.(\mathsf{wk}_0\,\sigma\ ,^\mathsf{p}\ \mathsf{var}^\mathsf{p}\,0)
\end{aligned}
$$

The idea is that if σ *is a substitution from* Γ *to* Δ*, then* $\mathsf{keep}\,\sigma$ *is a substitution between contexts* $\Gamma \rhd A[\sigma]$ *and* $\Delta \rhd A$ *for any type* A *where the last term is just a de Bruijn index* 0*. This occurs when defining* $(\Pi^\mathsf{p}\,A\,B)[\sigma]$ *as* $\Pi^\mathsf{p}\,(A[\sigma])\,(B[\mathsf{keep}\,\sigma])$*.*

We define the identity substitution on a context Γ *as follows, where* $\mathsf{keep}^{\|\Gamma\|}$ *is* keep *iterated* $\|\Gamma\|$ *times:*

$$
\begin{aligned}
&\mathsf{id}^\mathsf{p} : \mathsf{Con}^\mathsf{p} \to \mathsf{Sub}^\mathsf{p}\\
&\quad := \lambda\Gamma.\mathsf{keep}^{\|\Gamma\|}\epsilon^\mathsf{p}
\end{aligned}
$$

▶ **Lemma 32** (Exchange laws for weakening and substitution). *Below,* Z *denotes either a term or a type and* keep^n *denotes the* n *times iteration of* keep*.*

$$
\begin{aligned}
&\text{wk-wk} && : \mathsf{wk}_{n+p+1}(\mathsf{wk}_n\,Z) = \mathsf{wk}_n(\mathsf{wk}_{n+p}\,Z)\\
&\mathsf{wk}_n[n] && : (\mathsf{wk}_n\,Z)[n := z] = Z\\
&\mathsf{wk}_+[] && : (\mathsf{wk}_{n+p+1}\,Z)[n := \mathsf{wk}_p\,u] = \mathsf{wk}_{n+p}\,(Z[n := u])\\
&\mathsf{wk}[+] && : (\mathsf{wk}_n\,Z)[n+p+1 := u] = \mathsf{wk}_n\,(Z[n+p := u])\\
&[][+] && : Z[n := u][n+p := z] = Z[n+p+1 := z][n := (u[p := z])]
\end{aligned}
$$

$$\begin{array}{ll}
[\mathsf{keep}^n\text{-}\mathsf{wk}_0] & : Z[\mathsf{keep}^n\,(\mathsf{wk}_0\,\sigma)] = \mathsf{wk}_n(Z[\mathsf{keep}^n\,\sigma]) \\
\mathsf{wk}_n[\mathsf{keep}^n\text{-},] & : (\mathsf{wk}_n\,Z)[\mathsf{keep}^n\,(\sigma\,,^{\mathsf{p}}\,u)] = Z[\mathsf{keep}^n\sigma] \\
[:=][\mathsf{keep}] & : Z[n := u][\mathsf{keep}^n\,\sigma] = Z[\mathsf{keep}^{n+1}\,\sigma][n := u[\sigma]]
\end{array}$$

Proof. By induction on the untyped syntax. ◀

▶ **Corollary 33.** *As particular cases for $n = 0$, we get*

$$\begin{array}{ll}
\circ\mathsf{wk}_0 & : \sigma \circ (\mathsf{wk}_0\tau) = \mathsf{wk}_0(\sigma \circ \tau) \\
\mathsf{wk}_0\circ, & : \mathsf{wk}_0\,\sigma \circ (\tau\,,^{\mathsf{p}}\,t) = \sigma \circ \tau \\
[\mathsf{wk}_0] & : t[\mathsf{wk}_0\,\sigma] = \mathsf{wk}_0(t[\sigma]) \\
\mathsf{wk}_0[,] & : (\mathsf{wk}_0\,Z)[\sigma\,,^{\mathsf{p}}\,u] = Z[\sigma] \\
[0 :=][] & : Z[0 := u][\sigma] = Z[\mathsf{keep}\,\sigma][0 := u[\sigma]]
\end{array}$$

▶ **Lemma 34** (Composition functor law and associativity).

$$\begin{array}{l}
[][]\ : Z[\sigma][\tau] = Z[\sigma \circ \tau] \\
\mathsf{ass} : (\sigma \circ \delta) \circ \tau = \sigma \circ (\delta \circ \tau)
\end{array}$$

We defer laws for identity substitutions after the definition of the typing judgments, as the proofs require that some inputs are well-typed.

4.1.2 Typing Relations and Their Properties

▶ **Definition 35** (Typing relations). *The typing relations are defined as the following inductive type indexed over the untyped syntax:*

(1) Substitution calculus

$$\begin{array}{ll}
- \vdash & : \mathsf{Con}^{\mathsf{p}} \to \mathsf{Set} \\
- \vdash - & : \mathsf{Con}^{\mathsf{p}} \to \mathsf{Ty}^{\mathsf{p}} \to \mathsf{Set} \\
- \vdash - \in_{\mathbb{N}} - & : \mathsf{Con}^{\mathsf{p}} \to \mathbb{N} \to \mathsf{Ty}^{\mathsf{p}} \to \mathsf{Set} \\
- \vdash - \in - & : \mathsf{Con}^{\mathsf{p}} \to \mathsf{Tm}^{\mathsf{p}} \to \mathsf{Ty}^{\mathsf{p}} \to \mathsf{Set} \\
- \vdash - \Rightarrow - & : \mathsf{Con}^{\mathsf{p}} \to \mathsf{Sub}^{\mathsf{p}} \to \mathsf{Con}^{\mathsf{p}} \to \mathsf{Set} \\
\cdot^{\mathsf{w}} & : \cdot^{\mathsf{p}} \vdash \\
\epsilon^{\mathsf{w}} & : \Gamma \vdash \epsilon^{\mathsf{p}} \Rightarrow \cdot^{\mathsf{p}} \\
- \rhd^{\mathsf{w}} - & : (\Gamma \vdash) \to (\Gamma \vdash A) \to \Gamma \rhd^{\mathsf{p}} A \vdash \\
,^{\mathsf{w}} & : (\Delta \vdash) \to (\Gamma \vdash \sigma \Rightarrow \Delta) \to (\Delta \vdash A) \to (\Gamma \vdash t \in A[\sigma]) \to \Gamma \vdash \sigma\,,^{\mathsf{p}}\,t \Rightarrow \Delta \rhd^{\mathsf{p}} A \\
\mathsf{var}^{\mathsf{w}} & : (\Gamma \vdash n \in_{\mathbb{N}} A) \to \Gamma \vdash \mathsf{var}^{\mathsf{p}} n \in A \\
0^{\mathsf{w}} & : (\Gamma \vdash) \to (\Gamma \vdash A) \to \Gamma \rhd^{\mathsf{p}} A \vdash 0 \in_{\mathbb{N}} \mathsf{wk}^{\mathsf{p}}\,A \\
\mathsf{S}^{\mathsf{w}} & : (\Gamma \vdash) \to (\Gamma \vdash A) \to (\Gamma \vdash n \in_{\mathbb{N}} A) \to (\Gamma \vdash B) \to \Gamma \rhd^{\mathsf{p}} B \vdash \mathsf{S}\,n \in_{\mathbb{N}} \mathsf{wk}^{\mathsf{p}}\,A
\end{array}$$

(2) Universe

$$\begin{array}{ll}
\mathsf{U}^{\mathsf{w}} & : (\Gamma \vdash) \to \Gamma \vdash \mathsf{U}^{\mathsf{p}} \\
\mathsf{El}^{\mathsf{w}} & : (\Gamma \vdash) \to (\Gamma \vdash a \in \mathsf{U}^{\mathsf{p}}) \to \Gamma \vdash \mathsf{El}^{\mathsf{p}}\,a
\end{array}$$

(3) Inductive parameters

$$\begin{array}{ll}
\Pi^{\mathsf{w}} & : (\Gamma \vdash) \to (\Gamma \vdash a \in \mathsf{U}^{\mathsf{p}}) \to (\Gamma \rhd^{\mathsf{p}} \mathsf{El}^{\mathsf{p}}\,a \vdash B) \to \Gamma \vdash \Pi^{\mathsf{p}}\,a\,B \\
\mathsf{app}^{\mathsf{w}} & : (\Gamma \vdash) \to (\Gamma \vdash a \in \mathsf{U}^{\mathsf{p}}) \to (\Gamma \rhd^{\mathsf{p}} \mathsf{El}^{\mathsf{p}}\,a \vdash B) \\
& \to (\Gamma \vdash t \in \Pi^{\mathsf{p}}\,a\,B) \to (\Gamma \vdash u \in \mathsf{El}^{\mathsf{p}}\,a) \to \Gamma \vdash t \mathbin{@^{\mathsf{p}}} u \in B[0 := u]
\end{array}$$

(4) External parameters

$$\begin{aligned}
&\hat{\Pi}^{\mathsf{w}} && : (T : \mathsf{Set}) \to (A : T \to \mathsf{Ty}^{\mathsf{p}}) \to (\Gamma \vdash) \to ((t : T) \to \Gamma \vdash A\,t) \to \Gamma \vdash \hat{\Pi}^{\mathsf{p}}\,T\,A \\
&\mathsf{a\hat{p}p}^{\mathsf{w}} && : (T : \mathsf{Set}) \to (A : T \to \mathsf{Ty}^{\mathsf{p}}) \to (\Gamma \vdash) \to ((t : T) \to \Gamma \vdash A\,t) \\
& && \to (\Gamma \vdash t \in \hat{\Pi}^{\mathsf{p}}\,T\,A) \to (u : T) \to \Gamma \vdash t \hat{\hat{@}}\, u \in A\,u
\end{aligned}$$

There is possibility of redundancy in the arguments of the constructors. Here, we are “paranoid” (nomenclature from [9]), so that we get more inductive hypotheses when performing recursion.

▶ **Lemma 36** (Weakening preserves typing).

$$\begin{aligned}
\mathsf{wk_0}^{\mathsf{w}} &: (\Gamma \vdash A) \to (\Gamma; \Delta \vdash) \to \Gamma \rhd^{\mathsf{p}} A; \mathsf{wk}_0\,\Delta \vdash \\
\mathsf{wk}^{\mathsf{w}} &: (\Gamma \vdash A) \to (\Gamma; \Delta \vdash B) \to \Gamma \rhd^{\mathsf{p}} A; \mathsf{wk}_0\,\Delta \vdash \mathsf{wk}_{\|\Delta\|}\,B \\
\mathsf{wk}^{\mathsf{w}} &: (\Gamma \vdash A) \to (\Gamma; \Delta \vdash t \in B) \to \Gamma \rhd^{\mathsf{p}} A; \mathsf{wk}_0\,\Delta \vdash \mathsf{wk}_{\|\Delta\|}\,t \in \mathsf{wk}_{\|\Delta\|}\,B \\
\mathsf{wk_0}^{\mathsf{w}} &: (\Gamma \vdash A) \to (\Gamma \vdash \sigma \Rightarrow \Delta) \to \Gamma \rhd^{\mathsf{p}} A \vdash \mathsf{wk}_0\,\sigma \Rightarrow \Delta
\end{aligned}$$

Proof. By mutual induction on the typing relations. ◀

We show that judgments are stable under substitution.

▶ **Lemma 37** (Substitution preserves typing).

$$\begin{aligned}
[]^{\mathsf{w}} &: (\Gamma \vdash) \to (\Delta \vdash A) \to (\Gamma \vdash \sigma \Rightarrow \Delta) \to \Gamma \vdash A[\sigma] \\
[]^{\mathsf{w}} &: (\Gamma \vdash) \to (\Delta \vdash t \in A) \to (\Gamma \vdash \sigma \Rightarrow \Delta) \to \Gamma \vdash t[\sigma] \in A[\sigma] \\
[]^{\mathsf{w}} &: (\Delta \vdash x \in_{\mathbb{N}} A) \to (\Gamma \vdash \sigma \Rightarrow \Delta) \to \Gamma \vdash x[\sigma] \in A[\sigma] \\
\circ^{\mathsf{w}} &: (\Gamma \vdash) \to (\Gamma \vdash \sigma \Rightarrow \Delta) \to (\Delta \vdash \tau \Rightarrow E) \to \Gamma \vdash \tau \circ \sigma \Rightarrow E
\end{aligned}$$

Proof. By mutual induction on the typing relations. ◀

We show the category and functor laws involving identity substitution for well-formed types, terms and substitutions.

▶ **Lemma 38** (Identity laws).

$$\begin{aligned}
[\mathsf{id}^{\mathsf{p}}] &: (\Gamma \vdash A) \to A[\mathsf{id}^{\mathsf{p}}\,\Gamma] = A \\
[\mathsf{id}^{\mathsf{p}}] &: (\Gamma \vdash x \in_{\mathbb{N}} A) \to x[\mathsf{id}^{\mathsf{p}}\,\Gamma] = V\,x \\
[\mathsf{id}^{\mathsf{p}}] &: (\Gamma \vdash t \in A) \to t[\mathsf{id}^{\mathsf{p}}\,\Gamma] = t \\
\mathsf{idr}^{\mathsf{p}} &: (\Gamma \vdash \sigma \Rightarrow \Delta) \to \sigma \circ \mathsf{id}^{\mathsf{p}}\,\Gamma = \sigma \\
\mathsf{idl}^{\mathsf{p}} &: (\Gamma \vdash \sigma \Rightarrow \Delta) \to \mathsf{id}^{\mathsf{p}}\,\Delta \circ \sigma = \sigma
\end{aligned}$$

Finally, we show that the identity substitution itself is well-typed:

▶ **Lemma 39** (Typing for the identity substitution).

$$\mathsf{id}^{\mathsf{w}} : (\Gamma \vdash) \to \Gamma \vdash \mathsf{id}^{\mathsf{p}}\,\Gamma \Rightarrow \Gamma$$

▶ **Definition 40** (Proposition). *A type is a proposition, or proof-irrelevant, if it has at most one inhabitant.*

$$\mathsf{is\text{-}prop}\,T := (a : T) \to (a' : T) \to a = a'$$

▶ **Lemma 41** (Proof irrelevance of typing relations).

$$\begin{aligned}
&\mathsf{Con}^{\mathsf{wp}} : \text{is-prop}\,(\varGamma \vdash)\\
&\mathsf{Ty}^{\mathsf{wp}} \;\; : \text{is-prop}\,(\varGamma \vdash A)\\
&\mathsf{Var}^{\mathsf{wp}} : \text{is-prop}\,(\varGamma \vdash x \in_{\mathbb{N}} A)\\
&\mathsf{Tm}^{\mathsf{wp}} : \text{is-prop}\,(\varGamma \vdash t \in A)\\
&\mathsf{Sub}^{\mathsf{wp}} : \text{is-prop}\,(\varGamma \vdash \sigma \Rightarrow \varDelta)
\end{aligned}$$

▶ **Lemma 42** (Unicity of typing).

$$\begin{aligned}
&\mathsf{Tm}^{\mathsf{w}}{=}\mathsf{Ty} : (\varGamma \vdash t \in A) \to (\varGamma \vdash t \in B) \to A = B\\
&\mathsf{Var}^{\mathsf{w}}{=}\mathsf{Ty} : (\varGamma \vdash x \in_{\mathbb{N}} A) \to (\varGamma \vdash x \in_{\mathbb{N}} B) \to A = B
\end{aligned}$$

Let us consider for instance the application constructor $\mathsf{app}^{\mathsf{w}}$: for a codomain type B it yields an overall type $C = B[0 := u]$ for an application. Even if C is known a priori, there may be another B for which $B[0 := u] = C$, possibly leading to many proofs that $t \mathbin{@^{\mathsf{p}}} u$ has type C. Unicity of typing solves this issue, as B is then uniquely determined by the type $\Pi^{\mathsf{p}}\, A\, B$ of t.

4.1.3 The Syntax as a Signature Algebra

▶ **Definition 43** (Syntax for the theory of signatures). *We define the syntax as an element of* $\mathsf{SignAlg}$ *by pairs of untyped syntax and typing relations:*

$$\begin{aligned}
&\mathsf{Con}^{\mathsf{I}} && := \sum_{\varGamma} \varGamma \vdash\\
&\mathsf{Ty}^{\mathsf{I}}\,(\varGamma, \varGamma^{\mathsf{w}}) && := \sum_{A} \varGamma \vdash A\\
&\mathsf{Tm}^{\mathsf{I}}\,(\varGamma, \varGamma^{\mathsf{w}})(A, A^{\mathsf{w}}) && := \sum_{t} \varGamma \vdash t \in A\\
&\mathsf{Sub}^{\mathsf{I}}\,(\varGamma, \varGamma^{\mathsf{w}})(\varDelta, \varDelta^{\mathsf{w}}) && := \sum_{\sigma} \varGamma \vdash \sigma \Rightarrow \varDelta
\end{aligned}$$

The other fields are given straightforwardly. Regarding the equations, it is enough to prove them only for the untyped syntactic part: as we argued in Lemma 41, the proofs of typing judgments are automatically equal.

▶ Remark 44. Up until Definition 43, UIP is not used. Function extensionality on the other hand is necessary because the untyped metatheoretic Π takes a metatheoretic function as an argument. An example induction step that uses function extensionality is in Lemma 38, in particular in the case $(\hat{\Pi}\, T\, A)[\mathsf{id}] = \hat{\Pi}\, T\, A$. Indeed, the left hand side of this equation is equal to $\hat{\Pi}\, T\, (\lambda t.(A\, t)[\mathsf{id}])$ by definition, whereas the induction hypothesis states that $(t : T) \to (A\, t)[\mathsf{id}] = A\, t$.

4.2 Relating the Syntax to a Signature Algebra

It remains to show that the constructed syntax I is the initial signature algebra. To achieve this, we first define a relation between the syntax and any signature algebra, then show that the relation is functional, which lets us extract a signature morphism from the relation.

This approach is an alternative presentation of Streicher's method for interpreting preterms in an arbitrary model of type theory [30]. Streicher first defines a family of partial maps from the presyntax to a model, then shows that the maps are total on well-formed input. We have found that our approach is significantly easier to formalise. To see why, note that the right notion of partial map in type theory, which does not presume decidable definedness, is fairly heavyweight:

$$\mathsf{PartialMap}\,A\,B := A \to \big((P : \mathsf{Set}) \times \mathsf{is\text{-}prop}\,P \times (P \to B)\big)$$

In the above definition, we notice an opportunity for converting a fibered definition of a type family into an indexed one; if we drop the propositionality for P for the time being, we may equivalently return a family indexed over B, which is exactly just a relation $A \to B \to \mathsf{Set}$. Then, in our approach, we recover uniqueness of P through the functionality requirement on the $A \to B \to \mathsf{Set}$ relation, and totality by already assuming well-formedness of A. In type theory, using indexed families instead of display maps is a common convenience, since the former are natively supported, while the latter require carrying around auxiliary propositional equalities.

4.2.1 The Functional Relation

Given an $M : \mathsf{SignAlg}$, we define the functional relation satisfied by the $[\![-]\!] : \mathsf{SignMor}\,\mathsf{I}\,M$ by recursion on the typing judgments. If $\varGamma$ is a context in I and $\varGamma^M$ is a semantic context (i.e. a context in the signature algebra M), we want to define a type $\varGamma \sim \varGamma^M$ equivalent to $[\![\varGamma]\!] = \varGamma^M$. Of course, at this stage, $[\![-]\!]$ is not available yet since the point of defining this relation is to construct $[\![-]\!]$ in the end.

For a type A in a context $\varGamma$, we want to define a relation $A \sim A^M$ that is equivalent to $[\![A]\!] = A^M$. For this equality to make sense, the semantic type A^M must live in the semantic context $[\![\varGamma]\!]$. But again, $[\![-]\!]$ is not yet available at this stage. Exploiting the expected equivalence between $\varGamma \sim \varGamma^M$ and $[\![\varGamma]\!] = \varGamma^M$, we may consider defining $A \sim A^M$ under the hypotheses that A^M lies in a semantics context $\varGamma^M$ which is related to $\varGamma$. Then, the type of the relation for types is

$$(\varGamma : \mathsf{Con}^{\mathsf{I}}) \to (A : \mathsf{Ty}^{\mathsf{I}}\,\varGamma) \to (\varGamma^M : \mathsf{Con}^M) \to (\varGamma \sim \varGamma^M) \to (A^M : \mathsf{Ty}^M\,\varGamma^M) \to \mathsf{Set}$$

Note that the relation on contexts must be defined mutually with the relation on types (see for example the case of context extension), but here, the relation on contexts appears as the type of an argument of the relation on types. We want to avoid using such recursive-recursive definitions as they are not allowed by the elimination principles of indexed inductive types, so we instead just remove the hypothesis $\varGamma \sim \varGamma^M$ from the list of arguments. We proceed similarly for terms and substitutions. Actually, this removal is not without harm. For example, consider relating the empty substitution $\varGamma \vdash \epsilon^{\mathsf{P}} \Rightarrow \cdot^{\mathsf{P}}$ to a semantic substitution $\sigma^M : \mathsf{Sub}^M\,\varGamma^M\,\varDelta^M$. We would like to assert that σ^M equals the empty semantic substitution ϵ^M, but this is not possible because typechecking requires that $\varDelta^M$ is the empty semantic context. This is precisely what was ensured by the hypothesis $\cdot^{\mathsf{I}} \sim \varDelta^M$ we removed. Our way out here is to state that σ^M is related to the empty substitution if the target semantic context $\varDelta^M$ is empty, and, acknowledging this equality, if σ^M is the empty substitution.

Let us mention another possible solution for avoiding recursion-recursion: defining $A \sim A^M$ so that it is equivalent to $(e : [\![\varGamma]\!] = \varGamma^M) \times ([\![A]\!] =_{e\#} A^M)$. In comparison, our approach yields a more concise definition of the relation. For example, in the case of the universe, this would lead to the definition $\mathsf{U}^{\mathsf{w}}\,\varGamma^{\mathsf{w}} \sim A^M := (\varGamma^{\mathsf{w}} \sim \varGamma^M) \times (A^M = \mathsf{U}^M)$, instead of our definition $\mathsf{U}^{\mathsf{w}}\,\varGamma^{\mathsf{w}} \sim A^M := (A^M = \mathsf{U}^M)$.

▶ **Definition 45** (Relation $- \sim -$). *We define the relation by recursion on the typing judgments. In the following, we abbreviate $A^{\mathsf{w}} \sim_{\Gamma^M} A^M$ by $A^{\mathsf{w}} \sim A^M$ when Γ^M can be inferred, and similarly for terms and substitutions.*

(1) Substitution calculus

$$
\begin{aligned}
&- \sim - &&: \Gamma \vdash \to \mathsf{Con}^M \to \mathsf{Set} \\
&- \sim_{\Gamma^M} - &&: \Gamma \vdash A \to \mathsf{Ty}^M\,\Gamma^M \to \mathsf{Set} \\
&- \sim_{\Gamma^M \vdash A^M} - &&: \Gamma \vdash t \in A \to \mathsf{Tm}^M\,\Gamma^M\,A^M \to \mathsf{Set} \\
&- \sim_{\Gamma^M \vdash A^M} - &&: \Gamma \vdash x \in_{\mathbb{N}} A \to \mathsf{Tm}^M\,\Gamma^M\,A^M \to \mathsf{Set} \\
&- \sim_{\Gamma^M \Rightarrow \Delta^M} - &&: \Gamma \vdash \sigma \Rightarrow \Delta \to \mathsf{Sub}^M\,\Gamma^M\,\Delta^M \to \mathsf{Set}
\end{aligned}
$$

$$
\begin{aligned}
&\cdot^{\mathsf{w}} \sim \Gamma^M &&:= \Gamma^M = \cdot^M \\
&\epsilon^{\mathsf{w}} \sim_{\Gamma^M \Rightarrow E^M} \delta^M &&:= (e_E : E^M = \cdot^M) \times (\delta^M =_{e_E\#} \epsilon^M) \\
&(\Gamma^{\mathsf{w}} \rhd^{\mathsf{w}} A^{\mathsf{w}}) \sim \Delta^M &&:= \sum_{\Gamma^M} (\Gamma^{\mathsf{w}} \sim \Gamma^M) \times \sum_{A^M} (A^{\mathsf{w}} \sim A^M) \times \\
& &&\quad (\Delta^M = \Gamma^M \rhd^M A^M) \\
&(,^{\mathsf{w}} \Delta^{\mathsf{w}} \sigma^{\mathsf{w}} A^{\mathsf{w}} t^{\mathsf{w}}) \sim_{\Gamma^M \Rightarrow E^M} \delta^M &&:= \sum_{\Delta^M} (\Delta^{\mathsf{w}} \sim \Delta^M) \times \sum_{\sigma^M} (\sigma^{\mathsf{w}} \sim \sigma^M) \times \\
& &&\quad \sum_{A^M} (A^{\mathsf{w}} \sim A^M) \times \sum_{t^M} (t^{\mathsf{w}} \sim t^M) \times \\
& &&\quad (e_E : E^M = \Delta^M \rhd^M A^M) \times \\
& &&\quad (\delta =_{e_E\#} \sigma^M ,^M t^M) \\
&\mathsf{var}^{\mathsf{w}}\, x^{\mathsf{w}} \sim t^M &&:= x^{\mathsf{w}} \sim t^M \\
&\mathsf{0}^{\mathsf{w}} \Gamma^{\mathsf{w}} A^{\mathsf{w}} \sim_{\Delta^M \vdash B^M} t^M &&:= \sum_{\Gamma^M} (\Gamma^{\mathsf{w}} \sim \Gamma^M) \times \sum_{A^M} (A^{\mathsf{w}} \sim A^M) \times \\
& &&\quad (e_\Delta : \Delta^M = \Gamma^M \rhd^M A^M) \times \\
& &&\quad (e_B : B^M =_{e_\Delta\#} \mathsf{wk}^M\, A^M) \times (t^M =_{e_\Delta, e_B\#} \mathsf{vz}^M) \\
&\mathsf{S}^{\mathsf{w}} \Gamma^{\mathsf{w}} A^{\mathsf{w}} n^{\mathsf{w}} B^{\mathsf{w}} \sim_{\Delta^M \vdash C^M} t^M &&:= \sum_{\Gamma^M} (\Gamma^{\mathsf{w}} \sim \Gamma^M) \times \sum_{A^M} (A^{\mathsf{w}} \sim A^M) \times \\
& &&\quad \sum_{B^M} (B^{\mathsf{w}} \sim B^M) \times \sum_{n^M} (n^{\mathsf{w}} \sim n^M) \times \\
& &&\quad (e_\Delta : \Delta^M = \Gamma^M \rhd^M B^M) \times \\
& &&\quad (e_C : C^M =_{e_\Delta\#} \mathsf{wk}^M\, A^M) \times \\
& &&\quad (t^M =_{e_\Delta, e_C\#} \mathsf{vs}^M\, n^M)
\end{aligned}
$$

(2) Universe

$$
\begin{aligned}
&\mathsf{U}^{\mathsf{w}}\, \Gamma^{\mathsf{w}} A^{\mathsf{w}} \sim A^M &&:= A^M = \mathsf{U}^M \\
&\mathsf{El}^{\mathsf{w}}\, \Gamma^{\mathsf{w}} a^{\mathsf{w}} \sim A^M &&:= \sum_{a^M} (a^{\mathsf{w}} \sim a^M) \times (A^M = \mathsf{El}^M\, a^M)
\end{aligned}
$$

(3) Inductive parameters

$$
\begin{aligned}
&\Pi^{\mathsf{w}}\, \Gamma^{\mathsf{w}} a^{\mathsf{w}} B^{\mathsf{w}} \sim C^M &&:= \sum_{a^M} (a^{\mathsf{w}} \sim a^M) \times \sum_{B^M} (B^{\mathsf{w}} \sim B^M) \\
& &&\quad \times (C^M = \Pi^M\, a^M\, B^M)
\end{aligned}
$$

$$\begin{aligned}
\mathsf{app}^{\mathsf{w}}\,\Gamma^{\mathsf{w}} a^{\mathsf{w}} B^{\mathsf{w}} t^{\mathsf{w}} u^{\mathsf{w}} \sim_{\Gamma^M \vdash C^M} x^M :=\ & \sum_{a^M} (a^{\mathsf{w}} \sim a^M) \times \sum_{B^M} (B^{\mathsf{w}} \sim B^M) \times \\
& \sum_{t^M} (t^{\mathsf{w}} \sim t^M) \times \sum_{u^M} (u^{\mathsf{w}} \sim u^M) \times \\
& (e_C : C^M = B^M [0 := u^M]^M) \times \\
& (x^M =_{e_C \#} t^M @^M u^M)
\end{aligned}$$

(4) Metatheoretic parameters

$$\begin{aligned}
\hat{\Pi}^{\mathsf{w}} T\, A\, \Gamma^{\mathsf{w}} A^{\mathsf{w}} \sim B^M\ :=\ & \sum_{A^M} ((t : T) \to A^{\mathsf{w}} \sim A^M\, t) \times (B^M = \hat{\Pi}^M\, T\, A^M) \\
\mathsf{a\hat{p}p}^{\mathsf{w}} T\, A\, \Gamma^{\mathsf{w}} A^{\mathsf{w}} t^{\mathsf{w}} u \sim_{\Gamma^M \vdash B^M} x^M :=\ & \sum_{A^M} ((t : T) \to A^{\mathsf{w}} \sim A^M\, t) \times \sum_{t^M} (t^{\mathsf{w}} \sim t^M) \times \\
& (e_B : B^M = \hat{\Pi}^M\, T\, A^M) \times (x^M =_{e_B \#} t^M\, \hat{@}^M\, u)
\end{aligned}$$

4.2.2 Right Uniqueness

Next, we prove that this relation is right unique. Then, we show that the relation is stable under weakening and substitution. The last step consists of showing left-totality, i.e. giving a related semantic counterpart to any well-typed context, type or term. Everything is proved by induction on the typing judgments.

▶ **Lemma 46** (Right uniqueness). *The relation is right unique in the following sense:*

$$\begin{aligned}
&\Sigma{\sim}^{\mathsf{p}} : (\Gamma^{\mathsf{w}} : \Gamma \vdash) && \to \mathsf{is\text{-}prop}\,(\sum_{\Gamma^M} \Gamma^{\mathsf{w}} \sim \Gamma^M) \\
&\Sigma{\sim}^{\mathsf{p}} : (A^{\mathsf{w}} : \Gamma \vdash A) && \to \mathsf{is\text{-}prop}\,(\sum_{A^M} A^{\mathsf{w}} \sim A^M) \\
&\Sigma{\sim}^{\mathsf{p}} : (t^{\mathsf{w}} : \Gamma \vdash t \in A) && \to \mathsf{is\text{-}prop}\,(\sum_{t^M} t^{\mathsf{w}} \sim t^M) \\
&\Sigma{\sim}^{\mathsf{p}} : (x^{\mathsf{w}} : \Gamma \vdash x \in_{\mathbb{N}} A) && \to \mathsf{is\text{-}prop}\,(\sum_{x^M} x^{\mathsf{w}} \sim x^M) \\
&\Sigma{\sim}^{\mathsf{p}} : (\sigma^{\mathsf{w}} : \Gamma \vdash \sigma \Rightarrow \Delta) && \to \mathsf{is\text{-}prop}\,(\sum_{\sigma^M} \sigma^{\mathsf{w}} \sim \sigma^M)
\end{aligned}$$

▶ Remark 47. We mentioned that in order to avoid a recursive-recursive definition, we removed some hypotheses in the list of arguments of the relation. Such hypotheses are sometimes missed, for example in the case of the empty substitution or in the case of variables, requiring us to state additional equalities. Because of this, we need UIP to show that $\sum_{\Gamma^M} \Gamma \sim \Gamma^M$ and $\sum_{A^M} A \sim A^M$ are propositions. One may think that the use of UIP could be avoided by using the alternative verbose definition that we suggested before, expecting that $\sum_{\Gamma^M} \sum_{A^M} A \sim A^M$, rather than $\sum_{A^M} A \sim A^M$, is a proposition. However, this is not obvious. For example, we were not able to define $\mathsf{El}^{\mathsf{w}}\, \Gamma^{\mathsf{w}}\, a^{\mathsf{w}} \sim A^M$ in this fashion. In related work, Hugunin investigated constructing IITs without UIP [19] in cubical type theory, and demonstrated that well-formedness predicates used in syntactic algebras can subtly break in that setting. Also, while Hugunin does not use UIP, he only shows the simple version version of dependent elimination for the constructed IITs. Hence, the question whether IITs are reducible to inductive types in a UIP-free setting remains open.

4.2.3 Stability under Weakening and Substitution

Stability of the relation under weakening must be proved before stability under substitution. Indeed, in the proof of stability under substitution, the Π case requires to show that $\Pi\,(A[\sigma])\,(B[\mathsf{keep}\,\sigma])$ is related to $\Pi^M\,(A^M[\sigma]^M)\,(B^M[\mathsf{keep}^M\,\sigma]^M)$. We would like to apply the induction hypothesis, so we need to show that $\mathsf{keep}\,\sigma = \mathsf{wk}_0\,\sigma\ ,^{\mathsf{p}}\ \mathsf{var}^{\mathsf{p}}\,0$ is related to $\mathsf{keep}^M\,\sigma^M$, knowing that σ is related to σ^M. As $\mathsf{keep}\,\sigma = \mathsf{wk}_0\,\sigma\ ,^{\mathsf{p}}\ \mathsf{var}^{\mathsf{p}}\,0$, we are left with showing that $\mathsf{wk}_0\,\sigma = \sigma \circ \mathsf{wk}$ (where $\mathsf{wk} = \mathsf{wk}_0\,\mathsf{id}$) relates to its semantic counterpart.

To achieve that, we show that wk_0 preserves the relation, for types and terms. This requires to generalise a bit and show that wk_n preserves the relation, as $\mathsf{wk}_0\,(\Pi\,A\,B) = \Pi\,(\mathsf{wk}_0\,A)\,(\mathsf{wk}_1\,B)$. But remember that wk_n performs a weakening in the middle of a context, so we first define the semantic counterpart of this:

$$\begin{aligned}\Sigma\mathsf{wk}_0{\Rightarrow}^M : (\Gamma^{\mathsf{w}} : \Gamma \vdash) \to (\Gamma^{\mathsf{w}} \sim \Gamma^M) \to\ & \\ (\Delta^{\mathsf{w}} : \Gamma ; \Delta \vdash) \to (\Delta^{\mathsf{w}} \sim \Delta^M) \to\ & \\ (A^M : \mathsf{Ty}^M \Gamma^M) \to (\Delta'^M : \mathsf{Con}^M) \times (\mathsf{Sub}^M\,\Delta'^M\,\Delta^M)&\end{aligned}$$

Here, Δ'^M should be thought of as the context Δ^M where the weakening has happened in the middle of the context, by inserting the type A^M after the prefix Γ^M. Indeed, we expect that Γ^M is a prefix of Δ^M, as Γ^M relates to Γ and Δ^M to $\Gamma;\Delta$. The substitution from the weakened context to the original one must be computed at the same time otherwise the induction hypothesis is not strong enough. Then, we separate the two components under the same (implicit) hypotheses:

$$\begin{aligned}&\mathsf{wk_0}^M\,A^M\,\Delta^M &&: \mathsf{Con}^M\\ &\mathsf{wk}{\Rightarrow}^M\,A^M\,\Delta^M &&: \mathsf{Sub}^M(\mathsf{wk_0}^M A^M\,\Delta^M)\Delta^M\end{aligned}$$

Note that if recursion-recursion is available in the metatheory, $\mathsf{wk_0}^M$ and $\mathsf{wk}{\Rightarrow}^M$ can be defined directly without introducing this intermediate $\Sigma\mathsf{wk}_0 \Rightarrow^M$.

▶ **Lemma 48** (Weakening preserves typing)**.** *The following statements are all under the hypotheses* $(\Gamma^{\mathsf{w}} : \Gamma \vdash)$, $(\Gamma^{\mathsf{w}} \sim \Gamma^M)$, $(\Delta^{\mathsf{w}} : \Gamma;\Delta \vdash)$, $(\Delta^{\mathsf{w}} \sim \Delta^M)$, $(A^{\mathsf{w}} : \Gamma \vdash A)$, *and* $(A^{\mathsf{w}} \sim A^M)$*.*

$$\begin{aligned}&\mathsf{wk_0}{\sim} : \mathsf{wk_0}^{\mathsf{w}}\,A^{\mathsf{w}}\,\Delta^{\mathsf{w}} \sim \mathsf{wk_0}^M A^M \Delta^M\\ &\mathsf{wk}{\sim}\ \ : (T^{\mathsf{w}} : \Gamma;\Delta \vdash T) \to (T^{\mathsf{w}} \sim T^M) \to \mathsf{wk}^{\mathsf{w}}\,A^{\mathsf{w}}\,T^{\mathsf{w}} \sim T^M[\mathsf{wk_0}{\Rightarrow}^M A^M \Delta^M]^M\\ &\mathsf{wk}{\sim}\ \ : (t^{\mathsf{w}} : \Gamma;\Delta \vdash t \in T) \to (t^{\mathsf{w}} \sim t^M) \to \mathsf{wk}^{\mathsf{w}}\,A^{\mathsf{w}}\,t^{\mathsf{w}} \sim t^M[\mathsf{wk_0}{\Rightarrow}^M A^M \Delta^M]^M\\ &\mathsf{wk}{\sim}\ \ : (x^{\mathsf{w}} : \Gamma;\Delta \vdash t \in_{\mathbb{N}} T) \to (x^{\mathsf{w}} \sim x^M) \to \mathsf{wk}^{\mathsf{w}}\,A^{\mathsf{w}}\,x^{\mathsf{w}} \sim x^M[\mathsf{wk_0}{\Rightarrow}^M A^M \Delta^M]^M\end{aligned}$$

Proof. By mutual induction on the typing judgments. ◀

▶ **Lemma 49** (Weakening of substitution preserves $- \sim -$)**.**

$$\begin{aligned}\mathsf{wk_0}{\sim} : (\Gamma^{\mathsf{w}} : \Gamma \vdash) \to (\Gamma^{\mathsf{w}} \sim \Gamma^M) \to (A^{\mathsf{w}} : \Gamma \vdash A) \to (A^{\mathsf{w}} \sim A^M) \to&\\ (\sigma^{\mathsf{w}} : \Gamma \vdash \sigma \Rightarrow \Delta) \to (\sigma^{\mathsf{w}} \sim \sigma^M) \to (\mathsf{wk_0}^{\mathsf{w}} A^{\mathsf{w}} \sigma^{\mathsf{w}} \sim \sigma^M \circ^M \mathsf{wk}^M)&\end{aligned}$$

Proof. By induction on the typing judgments. ◀

Next, we want to prove that given any well-typed substitution $\sigma : \mathsf{Sub}\,\Gamma\,\Delta$ and semantic contexts Γ^M and Δ^M, related to Γ and Δ, respectively, there is a semantic substitution related to σ. In the extension case $\Gamma \vdash \sigma \,,^{\mathsf{p}}\, t \Rightarrow \Delta \rhd^{\mathsf{p}} A$, the induction hypothesis provides σ^M, Δ^M, A^M related to their syntactic counterpart. However, the premises of the induction hypothesis for getting a relevant t^M require showing that the type $A^M[\sigma^M]^M$ is related to the syntactic type $A[\sigma]$.

▶ **Lemma 50** (Preservation of the relation by substitution for variables)**.**

$$[]\!\sim : (\sigma^{\mathsf{w}} : \Gamma \vdash \sigma \Rightarrow \Delta) \to (\sigma^{\mathsf{w}} \sim \sigma^M) \to (x^{\mathsf{w}} : \Delta \vdash x \in_{\mathbb{N}} A) \to (x^{\mathsf{w}} \sim x^M) \to \\ []^{\mathsf{w}} x^{\mathsf{w}} \sigma^{\mathsf{w}} \sim x^M[\sigma^M]^M$$

Proof. Induction on typing. ◀

▶ **Lemma 51** (Preservation of the relation by substitution for types and terms)**.** *We assume* $(\sigma^{\mathsf{w}} : \Gamma \vdash \sigma \Rightarrow \Delta)$, $(\sigma^{\mathsf{w}} \sim \sigma^M)$, $(\Gamma^{\mathsf{w}} : \Gamma \vdash)$, $(\Gamma^{\mathsf{w}} \sim \Gamma^M)$, $(\Delta^{\mathsf{w}} : \Delta \vdash)$, *and* $(\Delta^{\mathsf{w}} \sim \Delta^M)$*:*

$$[]\!\sim : (A^{\mathsf{w}} : \Delta \vdash A) \to (A^{\mathsf{w}} \sim A^M) \to []^{\mathsf{w}} \Gamma^{\mathsf{w}} A^{\mathsf{w}} \sigma^{\mathsf{w}} \sim A^M[\sigma^M]^M$$
$$[]\!\sim : (t^{\mathsf{w}} : \Delta \vdash t \in A) \to (t^{\mathsf{w}} \sim t^M) \to []^{\mathsf{w}} \Gamma^{\mathsf{w}} t^{\mathsf{w}} \sigma^{\mathsf{w}} \sim t^M[\sigma^M]^M$$

Proof. Mutual induction on typing. ◀

▶ **Lemma 52** (The relation is preserved by composition and identity)**.** *We have the same hypotheses as in the previous lemma.*

$$\circ\!\sim\ : (E^{\mathsf{w}} : E \vdash) \to (E^{\mathsf{w}} \sim E^M) \to (\delta^{\mathsf{w}} : \Delta \vdash \delta \Rightarrow E) \to (\delta^{\mathsf{w}} \sim \delta^M) \to \\ \circ^{\mathsf{w}}\, \Gamma^{\mathsf{w}}\, \delta^{\mathsf{w}}\, \sigma^{\mathsf{w}} \sim \delta^M \circ^M \sigma^M$$
$$\mathsf{id}\!\sim : (\Gamma^{\mathsf{w}} : \Gamma \vdash) \to (\Gamma^{\mathsf{w}} \sim \Gamma^M) \to \mathsf{id}^{\mathsf{w}}\, \Gamma^{\mathsf{w}} \sim \mathsf{id}_{\Gamma^M}$$

4.2.4 Left-Totality and the Recursor

Before defining the recursor $[\![-]\!]$, we show left totality of the relation: that is, the image of a syntactic context is a unique semantic context which is related to it, and similarly for types and terms.

▶ **Lemma 53** (Left totality of $- \sim -$)**.**

$$\Sigma\mathsf{Con}\!\sim : (\Gamma^{\mathsf{w}} : \Gamma \vdash) \to \sum_{\Gamma^M} \Gamma^{\mathsf{w}} \sim \Gamma^M$$
$$\Sigma\mathsf{Ty}\!\sim\ : (\Gamma^{\mathsf{w}} : \Gamma \vdash) \to (\Gamma^{\mathsf{w}} \sim \Gamma^M) \to (A^{\mathsf{w}} : \Gamma \vdash A) \to (A^M : \mathsf{Ty}^M \Gamma^M) \times (A^{\mathsf{w}} \sim A^M)$$
$$\Sigma\mathsf{Tm}\!\sim\ : (\Gamma^{\mathsf{w}} : \Gamma \vdash) \to (\Gamma^{\mathsf{w}} \sim \Gamma^M) \to (A^{\mathsf{w}} : \Gamma \vdash A) \to (A^{\mathsf{w}} \sim A^M) \to \\ (t^{\mathsf{w}} : \Gamma \vdash t \in A) \to (t^M : \mathsf{Tm}^M \Gamma^M A^M) \times (t^{\mathsf{w}} \sim t^M)$$
$$\Sigma\mathsf{Var}\!\sim\ : (\Gamma^{\mathsf{w}} : \Gamma \vdash) \to (\Gamma^{\mathsf{w}} \sim \Gamma^M) \to (A^{\mathsf{w}} : \Gamma \vdash A) \to (A^{\mathsf{w}} \sim A^M) \to \\ (x^{\mathsf{w}} : \Gamma \vdash x \in_{\mathbb{N}} A) \to (x^M : \mathsf{Tm}^M \Gamma^M A^M) \times (x^{\mathsf{w}} \sim x^M)$$
$$\Sigma\mathsf{Sub}\!\sim : (\Gamma^{\mathsf{w}} : \Gamma \vdash) \to (\Gamma^{\mathsf{w}} \sim \Gamma^M) \to (\Delta^{\mathsf{w}} : \Delta \vdash) \to (\Delta^{\mathsf{w}} \sim \Delta^M) \to \\ (\sigma^{\mathsf{w}} : \Gamma \vdash \sigma \Rightarrow \Delta) \to (\sigma^M : \mathsf{Sub}^M \Gamma^M \Delta^M) \times (\sigma^{\mathsf{w}} \sim \sigma^M)$$

Proof. By induction on well-formedness judgments. The right uniqueness of the relation is used in this induction. ◀

▶ **Lemma 54** (Existence of the recursor). *For any M : SignAlg there is a $[\![-]\!]$: SignMor I M where* I *is given in Definition 43.*

Proof. Using the first projections in the construction of the left-totality construction and right uniqueness. ◀

4.3 Uniqueness

It remains to show that the morphism constructed in Lemma 54 is unique. We exploit right uniqueness of the relation: it is enough to show that any such morphism maps a syntactic context to a related semantic context, and similarly for types and terms.

▶ **Lemma 55.** *We assume an arbitrary signature morphism f from* I *to M. This induces the following maps:*

$$\begin{aligned}
&\mathsf{Con}^f : (\Gamma \vdash) \to \mathsf{Con}^M\\
&\mathsf{Ty}^f \quad : (\Gamma^{\mathsf{w}} : \Gamma \vdash) \to (\Gamma \vdash A) \to \mathsf{Ty}^M\,(\mathsf{Con}^f\,\Gamma^{\mathsf{w}})\\
&\mathsf{Tm}^f \;: (\Gamma^{\mathsf{w}} : \Gamma \vdash) \to (A^{\mathsf{w}} : \Gamma \vdash A) \to (\Gamma \vdash t \in A) \to \mathsf{Tm}^M\,(\mathsf{Con}^f\,\Gamma^{\mathsf{w}})\,(\mathsf{Ty}^f\,\Gamma^{\mathsf{w}}\,A^{\mathsf{w}})\\
&\mathsf{Var}^f \;: (\Gamma^{\mathsf{w}} : \Gamma \vdash) \to (A^{\mathsf{w}} : \Gamma \vdash A) \to (\Gamma \vdash x \in_{\mathbb{N}} A) \to \mathsf{Tm}^M\,(\mathsf{Con}^f\,\Gamma^{\mathsf{w}})\,(\mathsf{Ty}^f\,\Gamma^{\mathsf{w}}\,A^{\mathsf{w}})\\
&\mathsf{Sub}^f \;: (\Gamma^{\mathsf{w}} : \Gamma \vdash) \to (\Delta^{\mathsf{w}} : \Delta \vdash) \to (\Gamma \vdash \sigma \Rightarrow \Delta) \to \mathsf{Sub}^M\,(\mathsf{Con}^f\,\Gamma^{\mathsf{w}})\,(\mathsf{Con}^f\,\Delta^{\mathsf{w}})
\end{aligned}$$

The images of the above maps are related by $- \sim -$:

$$\begin{aligned}
&{\sim}\mathsf{Con}^f : (\Gamma^{\mathsf{w}} : \Gamma \vdash) \to \Gamma^{\mathsf{w}} \sim \mathsf{Con}^f\,\Gamma^{\mathsf{w}}\\
&{\sim}\mathsf{Ty}^f \quad : (\Gamma^{\mathsf{w}} : \Gamma \vdash) \to (A^{\mathsf{w}} : \Gamma \vdash A) \to \Gamma^{\mathsf{w}} \sim \mathsf{Ty}^f\,\Gamma^{\mathsf{w}}\,A^{\mathsf{w}}\\
&{\sim}\mathsf{Tm}^f \;: (\Gamma^{\mathsf{w}} : \Gamma \vdash) \to (A^{\mathsf{w}} : \Gamma \vdash A) \to (t^{\mathsf{w}} : \Gamma \vdash t \in A) \to \Gamma^{\mathsf{w}} \sim \mathsf{Tm}^f\,\Gamma^{\mathsf{w}}\,A^{\mathsf{w}}\,t^{\mathsf{w}}\\
&{\sim}\mathsf{Var}^f \;: (\Gamma^{\mathsf{w}} : \Gamma \vdash) \to (A^{\mathsf{w}} : \Gamma \vdash A) \to (x^{\mathsf{w}} : \Gamma \vdash x \in_{\mathbb{N}} A) \to \Gamma^{\mathsf{w}} \sim \mathsf{Var}^f\,\Gamma^{\mathsf{w}}\,A^{\mathsf{w}}\,x^{\mathsf{w}}\\
&{\sim}\mathsf{Sub}^f \;: (\Gamma^{\mathsf{w}} : \Gamma \vdash) \to (\Delta^{\mathsf{w}} : \Delta \vdash) \to (\sigma^{\mathsf{w}} : \Gamma \vdash \sigma \Rightarrow \Delta) \to \Gamma^{\mathsf{w}} \sim \mathsf{Sub}^f\,\Gamma^{\mathsf{w}}\,\Delta^{\mathsf{w}}\,\sigma^{\mathsf{w}}
\end{aligned}$$

Proof. By induction on typing relations. ◀

▶ **Corollary 56** (Uniqueness of the recursor). *By right uniqueness of $- \sim -$, there is only one morphism* SignMor I M *for any M.*

▶ **Theorem 57.** *If a model of ETT supports indexed W-types, it supports the theory of IIT signatures.*

Proof. We define the syntax I by Definition 43 which only used indexed W-types, the recursor by Lemma 54 and we prove its uniqueness property by Corollary 56. ◀

▶ **Corollary 58.** *If a model of ETT supports indexed W-types, it supports all IITs.*

Proof. Combining Theorem 57 and Theorem 23. ◀

5 Further Work

The current work only concerns finitary IITs. An extension would be to also allow infinitely branching inductive types such as W-types. This would first require giving semantics for infinitary IITs and adapting the term model construction. These would be straightforward following [24]. However, it seems to be more difficult to construct the syntax of infinitary

IIT signatures without using quotients. The reason is that such syntax would not be strictly restricted to neutral terms: the term model construction for infinitary IITs requires λ-abstraction and $\beta\eta$-rules for infinitary Π types. A definition of normal preterms and typing judgments on them may still be possible, but it appears to be much more complicated than before (the current authors have attempted this without conclusive success).

As mentioned in Section 4.2.2, it also remains an open problem whether IITs are reducible to inductive types in a UIP-free setting. To show this, we would need to construct the syntax of signatures without UIP, and also reproduce the semantics and term model construction for IITs without UIP.

References

1 Benedikt Ahrens, Ralph Matthes, and Anders Mörtberg. From signatures to monads in unimath. *Journal of Automated Reasoning*, 63(2):285–318, August 2019. `doi:10.1007/s10817-018-9474-4`.

2 Thorsten Altenkirch, Paolo Capriotti, Gabe Dijkstra, Nicolai Kraus, and Fredrik Nordvall Forsberg. Quotient inductive-inductive types. In Christel Baier and Ugo Dal Lago, editors, *Foundations of Software Science and Computation Structures*, pages 293–310, Cham, 2018. Springer International Publishing.

3 Thorsten Altenkirch, Neil Ghani, Peter Hancock, Conor McBride, and Peter Morris. Indexed containers. *J. Funct. Program.*, 25, 2015. `doi:10.1017/S095679681500009X`.

4 Thorsten Altenkirch and Ambrus Kaposi. Type theory in type theory using quotient inductive types. In Rastislav Bodik and Rupak Majumdar, editors, *Proceedings of the 43rd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2016, St. Petersburg, FL, USA, January 20 - 22, 2016*, pages 18–29. ACM, 2016. `doi:10.1145/2837614.2837638`.

5 Thorsten Altenkirch, Ambrus Kaposi, András Kovács, and Jakob von Raumer. Constructing inductive-inductive types via type erasure. In Marc Bezem, editor, *25th International Conference on Types for Proofs and Programs, TYPES 2019*. Centre for Advanced Study at the Norwegian Academy of Science and Letters, 2019.

6 Thorsten Altenkirch, Nuo Li, and Ondřej Rypáček. Some constructions on Ω-groupoids. In *Proceedings of the 2014 International Workshop on Logical Frameworks and Meta-languages: Theory and Practice*, LFMTP '14, pages 4:1–4:8, New York, NY, USA, 2014. ACM. `doi:10.1145/2631172.2631176`.

7 Robert Atkey, Neil Ghani, and Patricia Johann. A relationally parametric model of dependent type theory. In *Proceedings of the 41st ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, POPL '14, page 503–515, New York, NY, USA, 2014. Association for Computing Machinery. `doi:10.1145/2535838.2535852`.

8 Steve Awodey, Jonas Frey, and Sam Speight. Impredicative encodings of (higher) inductive types. In Anuj Dawar and Erich Grädel, editors, *Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2018, Oxford, UK, July 09-12, 2018*, pages 76–85. ACM, 2018. `doi:10.1145/3209108.3209130`.

9 Andrej Bauer, Philipp G. Haselwarter, and Théo Winterhalter. A modular formalization of type theory in Coq. In Ambrus Kaposi, editor, *23rd International Conference on Types for Proofs and Programs, TYPES 2017*. Eötvös Loránd University, 2017.

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

11 Paolo Capriotti. Notions of type formers. In Ambrus Kaposi, editor, *23rd International Conference on Types for Proofs and Programs, TYPES 2017*. Eötvös Loránd University, 2017.

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

13 James Chapman. Type theory should eat itself. *Electronic Notes in Theoretical Computer Science*, 228:21–36, January 2009. doi:10.1016/j.entcs.2008.12.114.

14 Jesper Cockx and Andreas Abel. Sprinkles of extensionality for your vanilla type theory. In Silvia Ghilezan and Ivetić Jelena, editors, *22nd International Conference on Types for Proofs and Programs, TYPES 2016*. University of Novi Sad, 2016.

15 Nils Anders Danielsson. A formalisation of a dependently typed language as an inductive-recursive family. In Thorsten Altenkirch and Conor McBride, editors, *TYPES*, volume 4502 of *Lecture Notes in Computer Science*, pages 93–109. Springer, 2006. doi:10.1007/978-3-540-74464-1_7.

16 Peter Dybjer. Internal type theory. In *Lecture Notes in Computer Science*, pages 120–134. Springer, 1996.

17 Peter Dybjer and Anton Setzer. A finite axiomatization of inductive-recursive definitions. In *Typed Lambda Calculi and Applications, volume 1581 of Lecture Notes in Computer Science*, pages 129–146. Springer, 1999.

18 Martin Hofmann. Syntax and semantics of dependent types. In *Semantics and Logics of Computation*, pages 79–130. Cambridge University Press, 1997.

19 Jasper Hugunin. Constructing inductive-inductive types in cubical type theory. In Mikołaj Bojańczyk and Alex Simpson, editors, *Foundations of Software Science and Computation Structures*, pages 295–312, Cham, 2019. Springer International Publishing.

20 Ambrus Kaposi. Re: separate definition of constructors? Email to the Agda mailing list, May 2019. URL: https://lists.chalmers.se/pipermail/agda/2019/011176.html.

21 Ambrus Kaposi, András Kovács, and Thorsten Altenkirch. Constructing quotient inductive-inductive types. *Proc. ACM Program. Lang.*, 3(POPL):2:1–2:24, January 2019. doi:10.1145/3290315.

22 Ambrus Kaposi, András Kovács, and Ambroise Lafont. Closed inductive-inductive types are reducible to indexed inductive types. In Marc Bezem, editor, *25th International Conference on Types for Proofs and Programs, TYPES 2019*. Centre for Advanced Study at the Norwegian Academy of Science and Letters, 2019.

23 Ambrus Kaposi and Jakob von Raumer. A syntax for mutual inductive families. In *5th International Conference on Formal Structures for Computation and Deduction (FSCD 2020)*, 2020. To appear.

24 András Kovács and Ambrus Kaposi. Large and infinitary quotient inductive-inductive types. In *35th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2020, Saarbrücken, Germany, July 8-11, 2020*, 2020. To appear.

25 Lorenzo Malatesta, Thorsten Altenkirch, Neil Ghani, Peter Hancock, and Conor McBride. Small induction recursion, indexed containers and dependent polynomials are equivalent, 2013. TLCA 2013.

26 Fredrik Nordvall Forsberg. *Inductive-inductive definitions*. PhD thesis, Swansea University, 2013.

27 Fredrik Nordvall Forsberg and Anton Setzer. Inductive-inductive definitions. In Anuj Dawar and Helmut Veith, editors, *CSL 2010*, volume 6247 of *Lecture Notes in Computer Science*, pages 454–468. Springer, Heidelberg, 2010.

28 Ulf Norell. *Towards a practical programming language based on dependent type theory*. PhD thesis, Chalmers University of Technology, 2007.

29 The Univalent Foundations Program. Homotopy type theory: Univalent foundations of mathematics. Technical report, Institute for Advanced Study, 2013.

30 Thomas Streicher. *Semantics of Type Theory: Correctness, Completeness, and Independence Results*. Birkhauser Boston Inc., Cambridge, MA, USA, 1991.