For 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@inria.fr

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 James Chapman-style syntax of type theory with conversion relations for each sort where e.g. the sort of types is indexed by contexts. It follows from previous work that all finitary IITs can be constructed from a quotient inductive-inductive type (QIIT), namely the theory of IIT signatures. This is a small domain-specific type theory where a context is a signature for an IIT. In this paper we construct the theory of IIT signatures using only inductive types, thereby showing a reduction of all finitary IITs to inductive types. We rely on function extensionality and uniqueness of identity proofs. We formalised the construction of the theory of IIT signatures in Agda.

2012 ACM Subject Classification Theory of computation → Logic

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

Digital Object Identifier 10.4230/LIPIcs.CVIT.2016.23

Funding *Ambrus Kaposi*: this author was supported by Thematic Excellence Programme, Industry and Digitization Subprogramme (NRDI Office, 2019) and by the European Union, co-financed by the European Social Fund (EFOP-3.6.2-16-2017-00013, Thematic Fundamental Research Collaborations Grounding Innovation in Informatics and Infocommunication).
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).

Acknowledgements The authors would like to thank Thorsten Altenkirch, Simon Boulier, Fredrik Nordvall-Forsberg and Jakob von Raumer for discussions on the topics of this paper.

1 Introduction

Many mutual inductive types can be reduced to indexed inductive types, where the index disambiguates different sorts. An example is the predicates isEven and isOdd defined mutually by the following constructors.

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

These can be reduced to the following single inductive family.

$$\mathsf{isEven?} \quad : \mathsf{Bool} \to \mathbb{N} \to \mathsf{Set}$$

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

Inductive-inductive types (IITs [19]) 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 the main example is still the Chapman-style [8] syntax for a type theory. A minimal use case is the IIT of contexts and types given by the following constructors.

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

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 at the same time. 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 shows up in higher inductive types [21]) is that later constructors can refer to previous constructors: in our case, El mentions ext.

The elimination principle for the above IIT has two motives and one method for each constructor. It is an example of a *recursive-recursive* function (following the nomenclature of [19]) which means that there are two mutually defined functions, where the type of the second function depends on the first function. The proof assistant Agda [20] allows the definition of such functions (even from non-IITs) and it is also currently the only proof assistant supporting IITs[1]. The elimination principle for Con–Ty says that given the following components,

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

we obtain two functions

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

with the following computation rules.

$$
\mathsf{elimCon}\,\mathsf{empty} \quad = \mathsf{empty}^D
$$

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

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

Reducing IITs to inductive types is an open problem. Forsberg [19] showed a reduction which only supports a simpler, non-recursive-recursive elimination principle. Hugunin [13] reduced several IITs to inductive types, working inside a cubical type theory, but he also only constructed a weaker eliminator.

The theory of signatures in [15] is a small type theory in which every context encodes a valid quotient inductive-inductive type (QIIT) signature. For example, the signature for natural numbers is simply the context $(Nat : \mathsf{U}, zero : Nat, suc : Nat \to Nat)$ of length three (Nat, $zero$ and suc are variable names). [15] showed that if this theory of signatures exists, then any finitary QIIT can be derived from it via a term model construction. If we omit the type former for equality constructors from the theory of signatures, we get a theory of finitary IIT signatures, and the construction of [15] still works: constructing an arbitrary IIT from the theory of IIT signatures does not need the Id type former to be present.

However, the QIIT term model construction assumes that we already have the syntax of QIIT signatures, and we can do induction over it. Hence, it only shows that all finitary QIITs are derivable from a particular fairly complicated QIIT. Likewise, restriced to finitary IITs, we only know that they are constructible from a particular QIIT.

In this paper, we construct the theory of IIT signatures, which is a large infinitary QIIT, using only indexed inductive types, uniqueness of identity types (UIP) and function extensionality. Thus, we show that all finitary IITs are reducible to the same feature set.

The contents of this paper were presented at the TYPES 2019 conference in Oslo [16].

1.1 Structure of the Paper

We first define a theory of IIT signatures in which every context is a signature (Section 2). We present this as an algebraic [3] theory, specifically as a quotient inductive-inductive type (QIIT). In Section 3 we show that this QIIT can be constructed using inductive types, UIP and function extensionality. In Section 4 we describe the construction of IITs from the syntax of IIT signatures.

1.2 Related Work

The current work builds heavily on the work of Kaposi et al. on quotient inductive-inductive types [15]; we reuse both QIIT syntax and semantics by restricting to IITs, and we reuse the term model construction of QIITs as well.

Forsberg [19] specifies inductive-inductive types and develops a semantics based on dialgebras. He also provides a reduction of IITs to indexed inductive types, but only constructs a weak version of the eliminator.

Hugunin [13] 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 does not construct full recursive-recursive eliminators.

Streicher [22] 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 3.2 for a comparison).

Voevodsky was interested in constructing initial models of type theories from presyntaxes. Inspired by this, Brunerie et al. [6] formalised Streicher's proof in Agda for a type theory with Π, Σ, $\mathbb{N}$, identity types and an infinite hierarchy of universes. He used UIP, function extensionality and quotient types in his formalisation. In this paper we construct a type theory without computation rules, hence we avoid using quotients.

Intrinsic (well-typed) syntax for type theories were constructed using IITs [8], inductive-recursive types [10, 5] and QIITs [3]. 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 [2] (using the essentially Streicher's idea of preterms and a typing predicate), small inductive-recursive types to indexed W-types [18], mutual inductive types to indexed W-types [17], W-types to natural numbers and quotients [1]. Quotient inductive-inductive types (and inductive-inductive types) can be reduced to quotient inductive types using the reduction of generalised algebraic theories to essentially algebraic theories [7].

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 [4]: here the initial algebra would be constructed directly for any IIT signature without going through an intermediate step.

1.3 Metatheory and Notation

We work in intensional Martin-Löf type theory, equipped with the following features:

- Predicative universes Set_i. We usually omit level indices in this paper.
- Dependent functions, 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. We use $A \times B$ for non-dependent pairs.
- Intensional propositional equality as $t = u$, with $\mathsf{refl} : t = t$. In this paper, we informally use extensional notation, and do not write explicit transports over equalities. However, we occasionally indicate by $e_1, \ldots, e_n \# t$ that t is well-typed thanks to the equalities $e_1, \ldots, e_n$.
- Uniqueness of identity proofs (UIP), expressing $(e : t = t) \to e = \mathsf{refl}$.
- Function extensionality, expressing $((x : A) \to f\,x = g\,x) \to f = g$.
- Natural numbers $\mathbb{N}$ with constructors $\mathsf{0} : \mathbb{N}$ and $\mathsf{S} : \mathbb{N} \to \mathbb{N}$.

1.4 Formalisation

Agda formalisation of Sections 2 and 3 can be found at `https://github.com/amblafont/UniversalII`. In addition to the features in the previous section, this formalisation also uses equality reflection for some equalities, through rewriting pragmas [9]. This is purely for convenience, and in principle we could eliminate the instances of equality reflection via UIP and function extensionality.

2 The Theory of IIT Signatures

The theory of IIT signatures is given as a category with families (CwF) [11] with certain additional type formers. The CwF part constitutes the substitution calculus, while the type formers allow declaring new sorts and operators in a signature. Below, we list all components for the theory of signatures; this can be viewed as a notion of model for the theory, which is an iterated Σ type. In this paper, we construct the *initial* such model, i.e. a model equipped with a unique recursion principle.

Note that initiality refers to a notion of homomorphism between models. As of now, there is no general metatheory for (large, infinitary) QIITs from which this notion would fall out. Hence, we give an ad-hoc definition for homomorphisms, which can be summarized as a functor between underlying categories which strictly preserves all additional family structure and type formers.

(1) Substitution calculus

$$\begin{array}{ll}
\mathsf{Con} & : \mathsf{Set} \\
\mathsf{Ty} & : \mathsf{Con} \to \mathsf{Set} \\
\mathsf{Sub} & : \mathsf{Con} \to \mathsf{Con} \to \mathsf{Set} \\
\mathsf{Tm} & : (\Gamma : \mathsf{Con}) \to \mathsf{Ty}\,\Gamma \to \mathsf{Set} \\
\mathsf{id} & : \mathsf{Sub}\,\Gamma\,\Gamma \\
- \circ - & : \mathsf{Sub}\,\Theta\,\Delta \to \mathsf{Sub}\,\Gamma\,\Theta \to \mathsf{Sub}\,\Gamma\,\Delta \\
\mathsf{ass} & : (\sigma \circ \delta) \circ \nu = \sigma \circ (\delta \circ \nu) \\
\mathsf{idl} & : \mathsf{id} \circ \sigma = \sigma \\
\mathsf{idr} & : \sigma \circ \mathsf{id} = \sigma \\
-[-] & : \mathsf{Ty}\,\Delta \to \mathsf{Sub}\,\Gamma\,\Delta \to \mathsf{Ty}\,\Gamma \\
-[-] & : \mathsf{Tm}\,\Delta\,A \to (\sigma : \mathsf{Sub}\,\Gamma\,\Delta) \to \mathsf{Tm}\,\Gamma\,(A[\sigma]) \\
[\mathsf{id}] & : A[\mathsf{id}] = A \\
[\circ] & : A[\sigma \circ \delta] = A[\sigma][\delta] \\
[\mathsf{id}] & : t[\mathsf{id}] = t \\
[\circ] & : t[\sigma \circ \delta] = t[\sigma][\delta] \\
\bullet & : \mathsf{Con} \\
\epsilon & : \mathsf{Sub}\,\Gamma\,\bullet \\
\bullet\eta & : \{\sigma : \mathsf{Sub}\,\Gamma\,\bullet\} \to \sigma = \epsilon \\
- \rhd - & : (\Gamma : \mathsf{Con}) \to \mathsf{Ty}\,\Gamma \to \mathsf{Con} \\
-,- & : (\sigma : \mathsf{Sub}\,\Gamma\,\Delta) \to \mathsf{Tm}\,\Gamma\,(A[\sigma]) \to \mathsf{Sub}\,\Gamma\,(\Delta \rhd A) \\
\pi_1 & : \mathsf{Sub}\,\Gamma\,(\Delta \rhd A) \to \mathsf{Sub}\,\Gamma\,\Delta \\
\pi_2 & : (\sigma : \mathsf{Sub}\,\Gamma\,(\Delta \rhd A)) \to \mathsf{Tm}\,\Gamma\,(A[\pi_1\sigma]) \\
\pi_1\beta & : \pi_1(\sigma, t) = \sigma \\
\pi_2\beta & : \pi_2(\sigma, t) = t \\
\pi\eta & : (\pi_1\,\sigma, \pi_2\,\sigma) = \sigma \\
,\circ & : (\sigma, t) \circ \delta = (\sigma \circ \delta, t[\delta])
\end{array}$$

(2) Universe

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

(3) Inductive parameters

$$
\begin{aligned}
&\Pi &&: (a : \mathsf{Tm}\,\Gamma\,\mathsf{U}) \to \mathsf{Ty}\,(\Gamma \triangleright \mathsf{El}\,a) \to \mathsf{Ty}\,\Gamma\\
&- @ - &&: \mathsf{Tm}\,\Gamma\,(\Pi\,a\,B) \to (u : \mathsf{Tm}\,\Gamma\,(\mathsf{El}\,a)) \to \mathsf{Tm}\,\Gamma\,(\mathsf{El}\,(B[\mathsf{id},\,u]))\\
&\Pi[] &&: (\Pi\,a\,B)[\sigma] = \Pi\,(a[\sigma])\,(B[\sigma^{\uparrow}])\\
&@[] &&: (t \mathbin{@} \alpha)[\sigma] = (t[\sigma]) \mathbin{@} (\alpha[\sigma])
\end{aligned}
$$

(4) Metatheoretic parameters

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

2.1 Notation and abbreviations

We define $\mathsf{wk} : \mathsf{Sub}\,(\Gamma \triangleright A) \to \Gamma$ as $\pi_1\,\mathsf{id}$. We can recover de Bruijn indices by having $0 = \pi_2\,\mathsf{id}$ and $n + 1 = n[\mathsf{wk}]$. However, for informal examples we instead use a nameful notation, where we introduce names in context extension and in the Π function domain, for example as $\cdot \triangleright A : \mathsf{U} \triangleright a : \mathsf{El}\,A$ in a signature for pointed sets.

We define $\sigma^{\uparrow}$ as $(\sigma \circ \mathsf{wk}, 0)$. We also abbreviate non-dependent Π functions by writing $\Pi\,a\,(B[\mathsf{wk}])$ as $a \Rightarrow B$.

2.2 Example signatures

Natural numbers:

$$\cdot \triangleright N : \mathsf{U} \triangleright z : \mathsf{El}\,N \triangleright s : N \Rightarrow \mathsf{El}\,N$$

Lists with elements of a given $A : \mathsf{Set}$ type. Here we use the $\hat{\Pi}$ function to include an external type into the signature.

$$\cdot \triangleright L : \mathsf{U} \triangleright \mathit{nil} : \mathsf{El}\,L \triangleright \mathit{cons} : \hat{\Pi}\,A\,(\lambda\_.\,L \Rightarrow L)$$

The inductive-inductive example from Section 1:

$$
\begin{aligned}
\cdot &\triangleright \mathit{Con} : \mathsf{U} \triangleright \mathit{Ty} : \mathit{Con} \Rightarrow \mathsf{U} \triangleright \mathit{empty} : \mathsf{El}\,\mathit{Con} \triangleright \mathit{ext} : \Pi\,(\Gamma : \mathit{Con})\,(\mathit{Ty} \mathbin{@} \Gamma \Rightarrow \mathit{Con})\\
&\triangleright U : \Pi\,(\Gamma : \mathit{Con})\,(\mathsf{El}\,(\mathit{Ty} \mathbin{@} \Gamma)) \triangleright \mathit{El} : \Pi\,(\Gamma : \mathit{Con})(\mathsf{El}\,(\mathit{Ty} \mathbin{@} (\mathit{ext} \mathbin{@} \Gamma \mathbin{@} (U \mathbin{@} \Gamma))))
\end{aligned}
$$

3 Constructing the Theory of IIT Signatures

The construction consists of the following steps:

1. Definition of untyped syntax (as a family of inductive datatypes) together with typing judgments (as inductive relations on the untyped syntax), and construction of a model of the theory of IIT signatures from well-formed terms, denoted S.
2. Construction of a morphism $\mathsf{rec} : \mathsf{S} \to M$ for arbitrary M, by:
   a. defining a relation $- \sim -$ between the well-typed syntax and a given model. The idea is that given a syntactic context Γ and a semantic context Γ^M of the model M, we have $\Gamma \sim \Gamma^M$ if and only if $\mathsf{rec}\ \Gamma = \Gamma^M$, and similarly for types, terms, and substitutions;
   b. showing that this relation is functional.
3. Proving the uniqueness of this morphism by showing that any morphism $f : \mathsf{S} \to M$ satisfies the relation. For example, for any syntactic context Γ we have $\Gamma \sim f\,\Gamma$.

The next sections detail each of these steps.

3.1 Syntactic Model

The goal is to define the syntactic model 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 giving the syntactic model. 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 IITs to a minimal feature set, and we get a stronger result if we do not use quotients.

The next sections present the definition of the untyped syntax and the associated typing judgments.

3.1.1 Untyped syntax

The untyped syntax is defined as the following inductive datatype. 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 $\mathsf{err}^{\mathsf{p}}$.

(1) Substitution calculus

$$
\begin{aligned}
&\mathsf{Con}^{\mathsf{p}} &&: \mathsf{Set}\\
&\mathsf{Ty}^{\mathsf{p}} &&: \mathsf{Set}\\
&\mathsf{Sub}^{\mathsf{p}} &&: \mathsf{Set}\\
&\mathsf{Tm}^{\mathsf{p}} &&: \mathsf{Set}\\
&\cdot^{\mathsf{p}} &&: \mathsf{Con}^{\mathsf{p}}\\
&\epsilon^{\mathsf{p}} &&: \mathsf{Sub}^{\mathsf{p}}\\
&- \rhd^{\mathsf{p}} - &&: \mathsf{Con}^{\mathsf{p}} \to \mathsf{Ty}^{\mathsf{p}} \to \mathsf{Con}^{\mathsf{p}}\\
&- ,^{\mathsf{p}} - &&: \mathsf{Sub}^{\mathsf{p}} \to \mathsf{Tm}^{\mathsf{p}} \to \mathsf{Sub}^{\mathsf{p}}\\
&\mathsf{var}^{\mathsf{p}} &&: \mathbb{N} \to \mathsf{Tm}^{\mathsf{p}}
\end{aligned}
$$

(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

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

(4-5) Metatheoretic parameters

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

(6) Default value

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

3.1.2 Untyped weakening

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.

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

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 Section 3.1.6, we show that weakening preserves typing. Stating a typing rule for this operation requires weakening at the middle of a context. We consider pairs of untyped contexts, which should be thought of as a splitting of a context at some position. The full context is recovered by merging the two components:

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

We think of the second context as a telescope over the first one. We also define weakening for telescopes, which will be used to give typing rules for telescopes in Section 3.1.5:

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

$\|\Gamma\|$ denotes the length of a the context Γ.

3.1.3 Untyped substitution

We define single substitution by recursion on presyntax:

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

$$-[- := -] : \mathsf{Tm}^{\mathsf{P}} \to \mathbb{N} \to \mathsf{Tm}^{\mathsf{P}} \to \mathsf{Tm}^{\mathsf{P}}$$

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

However, to construct the initial model of the QIIT, we need to define the full 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 it is defined by recursion. We also make use of the following definition:

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

The idea is that if σ is a substitution between contexts Γ and Δ, then $\mathsf{keep}\,\sigma$ is a substitution between contexts $\Gamma \rhd A[\sigma]$ and $\Delta \rhd A$ for any type A. This occurs when defining $(\Pi^{\mathsf{P}}\,A\,B)[\sigma]$ as $\Pi^{\mathsf{P}}\,(A[\sigma])\,(B[\mathsf{keep}\,\sigma])$.

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

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

3.1.4 Exchange laws: weakening and substitution

Many lemmas for types and terms are shown by induction on the untyped syntax. Below, Z denotes either a term or a type.

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

Below, keep^n denotes keep iterated n times.

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

As particular cases for $n = 0$, we get

$$\begin{aligned}
\circ\mathsf{wk}_0 &: \sigma \circ (\mathsf{wk}_0\tau) = \mathsf{wk}_0(\sigma \circ \tau) \\
\mathsf{wk}_0\circ, &: \mathsf{wk}_0\,\sigma \circ (t\ ,^{\mathsf{P}}\ \tau) = \sigma \circ \tau \\
[\mathsf{wk}_0] &: t[\mathsf{wk}_0\,\sigma] = \mathsf{wk}_0(t[\sigma])
\end{aligned}$$

$$\begin{aligned}
&\mathsf{wk}_0[,] \;: (\mathsf{wk}_0 Z)[u \,,^{\mathsf{p}}\, \sigma] = Z[\sigma] \\
&[0:=][] : Z[0 := u][\sigma] = Z[\mathsf{keep}\,\sigma][0 := u[\sigma]]
\end{aligned}$$

Finally, we show the following:

$$\begin{aligned}
&[][] \;: Z[\sigma][\tau] = Z[\sigma \circ \tau] \\
&\mathsf{ass} : (\sigma \circ \delta) \circ \tau = \sigma \circ (\delta \circ \tau)
\end{aligned}$$

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

3.1.5 Typing judgments

The typing judgments are defined as the following inductive datatype indexed over the untyped syntax:

(1) Substitution calculus

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

(2) Universe

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

(3) Inductive parameters

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

(4) Inductive parameters

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

There is possibility of redundancy in the arguments of the constructors. Here, we are "paranoid", so that we get more inductive hypotheses when performing recursion.

3.1.6 Weakening preserves typing

We prove by mutual induction that typing judgments are stable under weakening, for contexts, types, terms, and substitutions.

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

3.1.7 Substitution preserves typing

We show that judgments are stable under substitution.

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

3.1.8 Laws for identity substitutions

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

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

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

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

3.1.9 Proof irrelevance and unicity of typing

A type is a proposition, or proof-irrelevant, if it has at most one inhabitant.

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

We show that each of the typing judgments is unique in the following sense:

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

We also show unicity of typing:

$$\mathsf{Tm}^{\mathsf{w}}{=}\mathsf{Ty} : (\Gamma \vdash t \in A) \to (\Gamma \vdash t \in B) \to A = B$$

$$\mathsf{Var^w{=}Ty} : (\Gamma \vdash x \in_{\mathbb{N}} A) \to (\Gamma \vdash x \in_{\mathbb{N}} B) \to A = B$$

Let us consider for instance the application constructor $\mathsf{app^w}$: for a codomain type B it yields an overall type $C = B[\mathsf{0} := u]$ for an application. Even if C is known a priori, there may be another B for which $B[\mathsf{0} := u] = C$, 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.

3.1.10 Syntactic model

The syntactic model is obtained by packing the untyped syntax with the typing judgments:

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

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 Section 3.1.9, the proofs of typing judgments are automatically equal.

Up until the construction of the syntactic model, 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 the type preservation proof for identity substitutions (this is an equation satisfied by a model of the QIIT), 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 $(t : T) \to (A\, t)[\mathsf{id}] = A\, t$.

3.2 Relating the Syntax to a Model

It remains to show that the constructed syntactic model is initial. To achieve this, we first define a relation between the syntactic model and an arbitrary model, then show that the relation is functional, which lets us extract a homomorphism from the relation.

This approach is an alternative version of Streicher's method for interpreting preterms in an arbitrary model [22]. 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. Too 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.

3.2.1 The functional relation

Given a model M of the QIIT, we define the functional relation satisfied by the initial morphism $\mathsf{rec} : \mathsf{S} \to M$ by recursion on the typing judgments. If Γ is a context in S and Γ^M is a semantic context (i.e. a context of the model M), we want to define a type $\Gamma \sim \Gamma^M$ equivalent to $\mathsf{rec}\,\Gamma = \Gamma^M$. Of course, at this stage, rec is not available yet since the point of defining this relation is to construct rec in the end.

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

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

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 $\Gamma \sim \Gamma^M$ 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 $\Gamma \vdash \epsilon^{\mathsf{P}} \Rightarrow \cdot^{\mathsf{P}}$ to a semantic substitution $\sigma^M : \mathsf{Sub}^M\,\Gamma^M\,\Delta^M$. We would like to assert that σ^M equals the empty semantic substitution ϵ^M, but this is not possible because typechecking requires that Δ^M is the empty semantic context. This is precisely what was ensured by the hypothesis $\cdot^{\mathsf{S}} \sim \Delta^M$ we removed. Our way out here is to state that σ^M is related to the empty substitution if the target semantic context Δ^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 $(e : \mathsf{rec}\,\Gamma = \Gamma^M) \times (\mathsf{rec}\,A = e\#A^M)$. 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 $\mathsf{U}^{\mathsf{w}}\,\Gamma^{\mathsf{w}} \sim A^M := (\Gamma^{\mathsf{w}} \sim \Gamma^M) \times (A^M = \mathsf{U}^M)$, instead of our definition $\mathsf{U}^{\mathsf{w}}\,\Gamma^{\mathsf{w}} \sim A^M := (A^M = \mathsf{U}^M)$.

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

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

$$\begin{array}{ll}
\cdot^{\mathsf{w}} \sim \Gamma^M & := \Gamma^M = \cdot^M \\
\epsilon^{\mathsf{w}} \sim_{\Gamma^M \Rightarrow E^M} \delta^M & := (e_E : E^M = \cdot^M) \times (\delta^M = e_E\#\epsilon^M) \\
(\Gamma^{\mathsf{w}} \rhd^{\mathsf{w}} A^{\mathsf{w}}) \sim \Delta^M & := \sum_{\Gamma^M}(\Gamma^{\mathsf{w}} \sim \Gamma^M) \times \sum_{A^M}(A^{\mathsf{w}} \sim A^M) \times
\end{array}$$

$$(\Delta^M = \Gamma^M \triangleright^M A^M)$$

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

(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

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

(4) Metatheoretic parameters

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

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.

3.2.2 Right uniqueness

We show by induction that the relation is right unique in the following sense:

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

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 [14] in cubical type theory, and demonstrated that well-formedness predicates used in syntactic models can subtly break in that setting. Also, while Hugunin does not use UIP, he only shows a weak 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.

3.2.3 Stability under weakening

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 $\mathsf{keep}\,\sigma = \mathsf{var}^{\mathsf{p}}\,0\,,^{\mathsf{p}}\,\mathsf{wk}_0\,\sigma$ is related to $\mathsf{keep}^M\,\sigma^M$, knowing that σ is related to σ^M. As $\mathsf{keep}\,\sigma = \mathsf{var}^{\mathsf{p}}\,0\,,^{\mathsf{p}}\,\mathsf{wk}_0\,\sigma$, 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, ${\Delta'}^M$ 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 recursion hypothesis is not strong enough. Then, we seperate the two components under the same (implicit) hypotheses:

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

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 $\Sigma\mathsf{wk_0} \Rightarrow^M$.

Now, we are ready to prove by mutual recursion on well-typed judgments that 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 $(A^{\mathsf{w}} \sim A^M)$.

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

Then we deduce, still by induction, that weakening of substitution preserves the relation:

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

3.2.4 Stability under substitution

We want to prove that given any well-typed substitution $\sigma : \mathsf{Sub}\,\Gamma\,\Delta$ and semantic contexts Γ^M and Δ^M, respectively related to Γ and Δ, there exists a semantic substitution which is 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 $A[\sigma]$. We first establish preservation of the relation by substitution for variables:

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

Then we show it for terms and types by mutual induction under the common hypotheses $(\sigma^{\mathsf{w}} : \Gamma \vdash \sigma \Rightarrow \Delta)$, $(\sigma^{\mathsf{w}} \sim \sigma^M)$, $(\Gamma^{\mathsf{w}} : \Gamma \vdash)$, $(\Gamma^{\mathsf{w}} \sim \Gamma^M)$, $(\Delta^{\mathsf{w}} : \Delta \vdash)$, and $(\Delta^{\mathsf{w}} \sim \Delta^M)$:

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

Eventually, we show under the same hypotheses the following statement

$$\circ{\sim} : (E^{\mathsf{w}} : E \vdash) \to (E^{\mathsf{w}} \sim E^M) \to (\delta^{\mathsf{w}} : \Delta \vdash \delta \Rightarrow E) \to (\delta^{\mathsf{w}} \sim \delta^M) \to$$

$$\circ^{\mathsf{w}}\,\Gamma^{\mathsf{w}}\,\delta^{\mathsf{w}}\,\sigma^{\mathsf{w}} \sim \delta^M \circ^M \sigma^M$$

and the fact that identity preserves the relation:

$$\mathsf{id}{\sim} : (\Gamma^{\mathsf{w}} : \Gamma \vdash) \to (\Gamma^{\mathsf{w}} \sim \Gamma^M) \to \mathsf{id}^{\mathsf{w}}\,\Gamma^{\mathsf{w}} \sim \mathsf{id}_{\Gamma^M}$$

3.3 Left-Totality and the Recursor

For the recursor, we build a morphism from the syntactic model to the semantic one. The image of a syntactic context is a unique semantic context which is related to it, and similarly for types and terms. Thus, as a first step, we use induction on well-formedness judgments to construct semantic counterparts:

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

The right uniqueness of the relation is used in this induction. It is then straightforward to show (without induction, but using right uniqueness) that the first projection of these constructions yields a model morphism from the syntax to the model.

3.4 Uniqueness

Our goal is to show that the syntactic model is initial. Thus, it remains to show that the morphism constructed in the previous section 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.

More formally, we assume an arbitrary morphism f from the syntax to the model, inducing the following maps:

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

Then, we show by induction on typing judgments that the image is related:

$$\begin{aligned}
&{\sim}\mathsf{Con}^f : (\Gamma^{\mathsf{w}} : \Gamma \vdash) \to \Gamma^{\mathsf{w}} \sim \mathsf{Con}^f\,\Gamma^{\mathsf{w}} \\
&{\sim}\mathsf{Ty}^f \;\;: (\Gamma^{\mathsf{w}} : \Gamma \vdash) \to (A^{\mathsf{w}} : \Gamma \vdash A) \to \Gamma^{\mathsf{w}} \sim \mathsf{Ty}^f\,\Gamma^{\mathsf{w}}\,A^{\mathsf{w}} \\
&{\sim}\mathsf{Tm}^f \;: (\Gamma^{\mathsf{w}} : \Gamma \vdash) \to (A^{\mathsf{w}} : \Gamma \vdash A) \to (t^{\mathsf{w}} : \Gamma \vdash t \in A) \to \Gamma^{\mathsf{w}} \sim \mathsf{Tm}^f\,\Gamma^{\mathsf{w}}\,A^{\mathsf{w}}\,t^{\mathsf{w}}
\end{aligned}$$

$$\sim\mathsf{Var}^f \;: (\Gamma^{\mathsf{w}} : \Gamma \vdash) \to (A^{\mathsf{w}} : \Gamma \vdash A) \to (x^{\mathsf{w}} : \Gamma \vdash x \in_{\mathbb{N}} A) \to \Gamma^{\mathsf{w}} \sim \mathsf{Var}^f\, \Gamma^{\mathsf{w}}\, A^{\mathsf{w}}\, x^{\mathsf{w}}$$
$$\sim\mathsf{Sub}^f \;: (\Gamma^{\mathsf{w}} : \Gamma \vdash) \to (\Delta^{\mathsf{w}} : \Delta \vdash) \to (\sigma^{\mathsf{w}} : \Gamma \vdash \sigma \Rightarrow \Delta) \to \Gamma^{\mathsf{w}} \sim \mathsf{Sub}^f\, \Gamma^{\mathsf{w}}\, \Delta^{\mathsf{w}}\, \sigma^{\mathsf{w}}$$

This justifies the uniqueness of the morphism, by right uniqueness of $- \sim -$.

4 Constructing IITs from the Theory of IIT Signatures

We construct all IITs describable by the theory of IIT signatures through the term model construction of [15]. There, it is shown that from the syntax of the theory of signatures for finitary QIITs, one can construct any particular finitary QIIT. The idea is that for a signature Γ, the initial algebra can be built from sets of terms of the form $\mathsf{Tm}\,\Gamma\,(\mathsf{El}\,a)$. For example, considering the signature for natural numbers

$$\Gamma := \big(\cdot \rhd N : \mathsf{U} \rhd z : \mathsf{El}\,N \rhd s : N \Rightarrow \mathsf{El}\,N\big),$$

we have that $\mathsf{Tm}\,\Gamma\,(\mathsf{El}\,N)$ is the set of natural numbers up to isomorphism, since the only way to construct these terms is by using s and z from Γ.

In this section, we only need to check that the signatures and term model construction in [15] restrict to IITs, i.e. that from a syntax for the theory of IIT signatures all finitary IITs are constructible.

1. Finitary IIT signatures are obtained exactly by dropping equality constructors from finitary QIIT signatures. It follows that any model for the theory of QIIT signatures restricts to IITs, hence we inherit the categorical semantics of QIITs.
2. The term model construction also restricts in a straightforward way. For QIITs, term algebras are constructed by induction on the syntax of signatures, and then another induction constructs the eliminator (i.e. dependent induction principle). In both cases, simply dropping equality constructors from the induction yields restriction to IITs.

However, we shall mention that the Agda formalisation for the current paper cannot be directly plugged into the Agda code for [15] (which includes most of the term model construction). One reason is that the QIIT formalisation uses strict computation rules (given by Agda rewrite rules) for induction over signatures, while here we show propositional computation. This mismatch can be in principle remedied by noting that both formalisations use UIP and function extensionality, hence we can switch between strict and propositional equalities, via the known translations between extensional and intensional type theories [12, 23].

Also, the current paper proves initiality, i.e. unique recursion for signatures, while [15] uses dependent elimination. We expect that the two notions are equivalent. An extension of [15] to large infinitary QIITs would entail this equivalence, since that would cover the (large, infinitary) theory of IIT signatures.

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 (to our knowledge there is no previously published semantics that we can borrow), and also giving a term model construction analogously to finitary QIITs. These steps seem feasible. 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: we would need λ-abstraction and $\beta\eta$-rules for infinitary Π types, in order to allow a term model construction for infinitary IITs. 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 3.2.2, 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, Aug 2019. URL: `https://doi.org/10.1007/s10817-018-9474-4`, `doi:10.1007/s10817-018-9474-4`.

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

3 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. URL: `http://doi.acm.org/10.1145/2837614.2837638`, `doi:10.1145/2837614.2837638`.

4 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.

5 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. URL: `http://doi.acm.org/10.1145/2631172.2631176`, `doi:10.1145/2631172.2631176`.

6 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`.

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

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

9 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.

10 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.

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

12 Martin Hofmann. *Extensional concepts in intensional type theory*. Thesis. University of Edinburgh, Department of Computer Science, 1995.

13 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.

14 Jasper Hugunin. Constructing inductive-inductive types in cubical type theory. In *International Conference on Foundations of Software Science and Computation Structures*, pages 295–312. Springer, 2019.

15 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`.

16 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.

17 Ambrus Kaposi and Jakob von Raumer. A syntax for mutual inductive families. Unpublished draft, October 2019. URL: `https://bitbucket.org/javra/inductive-families/raw/master/main.pdf`.

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

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

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

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

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

23 Théo Winterhalter, Matthieu Sozeau, and Nicolas Tabareau. Eliminating reflection from type theory. In *Proceedings of the 8th ACM SIGPLAN International Conference on Certified Programs and Proofs*, pages 91–103. ACM, 2019.