Relative induction principles for second-order generalized algebraic theories

RAFAËL BOCQUET

SUBMITTED IN FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

EÖTVÖS LORÁND UNIVERSITY
DOCTORAL SCHOOL OF INFORMATICS

Contents

Chapter 1

Introduction

This thesis is a contribution towards the goal of having general mathematical tools for the metatheory of dependent type theories. Type theorists study many different type theories, including Martin-Löf type theory (MLTT), homotopy type theory (HoTT), observational type theory, cubical type theory, simplicial type theory, modal and multimodal type theories, etc. Each type theory listed above is not quite a single type theory, but rather a family of type theories which differ in more minor ways: Do we have an η-rule for Π-types or for Σ-types? Do we have W-types or more general inductive types? Do we have universes with first-class universe levels? Do we have some form of equality reflection?

We want to be able to reuse the same mathematical framework when studying different type theories, so as to be able to focus on the specificities of the type theory under consideration, and modularly extend results to other type theories. More importantly, we do not want to study type theories in isolation, but also understand how they can be compared. At the very least, we want to assemble them into a category of type theories.

Dependent type theories are used for two main purposes, which are related but distinct:

- They are used as general formal frameworks that can express most mathematics, and as the core languages of proof assistants such as Agda (Agda Developers 2025), Lean (Moura and Ullrich 2021) or Rocq (The Rocq Development Team 2025).
- Some type theories are the internal languages of some mathematical structures, more commonly classes of categories or ∞-categories. For example, extensional type theory can be interpreted in any topos, and homotopy type theory can be interpreted into ∞-toposes.

  Most of the time, when using internal languages, we use the internal language of a topos or ∞-topos (this can require embedding a category into a larger category that is a topos, e.g. using the Yoneda lemma). The strength of internal languages is that constructive mathematics can be interpreted in the internal language of a topos; we can work in the internal language with the mathematical intuitions acquired from ordinary mathematics.

These two purposes approximately correspond to the use of type theory to express *analytic* and *synthetic* mathematics. When doing synthetic mathematics by working in an internal language, one is often interested in the interpretation of the internal development in a model that has been defined analytically. [1]

[1] A current shortcoming of proof assistants is that while they are often used for either general math-

There are at least two reasons that motivate the study of various type theories, rather than a single type theory. One is that the use of type theories as internal languages often require specific type theories, especially if one wants the type theory to have good computational behavior. For example HoTT as defined in the HoTT book (Univalent Foundations Program 2013) is essentially an extension of MLTT by the univalence axiom, and shares many of the properties of MLTT. However, if one wants a type theory with (strict) canonicity, we have to move to cubical type theory. The other reason is that there is a tension between using type theory as a language for mathematics, and doing the semantics of type theory. When studying a type theory, it helps to have a type theory that is as simple as possible; when using a type theory, it helps to have a type theory that is as expressive as possible, which usually means a more complex type theory.

1.1 Conservativity conjectures

The starting point for most of my research has been a conjecture proposed at TYPES 2017 by Altenkirch, Capriotti, et al. (2017). Although very little of the content of this document is directly related to that conjecture, most of it has been motivated by it, and it seems appropriate to start this introduction by a short discussion of the conjecture.

One of the main motivation for cubical type theory is to provide a *computational interpretation* of univalence. Homotopy type theory does not satisfy strict canonicity, as the univalence axiom blocks computation. In cubical type theory, univalence is not an axiom, but a theorem derived from primitives that admit computational content. It has been proven that cubical type theory satisfies strict canonicity and normalization, and proof assistants for cubical type theory have been implemented. Thus, it should be possible to take a term written in HoTT, interpret it in cubical type theory (or copy it in a cubical proof assistant), and compute with the resulting term in cubical type theory.

For early variants of cubical type theory, this was however not possible, due to a mismatch between the identity types of cubical type theory (which are defined as path types, whose elements are maps from the cubical interval to some other type), and the identity type of HoTT (defined as an indexed inductive family, or at least with the corresponding induction principle, called the J-rule).

While path types satisfy the elimination principle of the identity types, they do not satisfy its computation rule (the Jβ-rule). More precisely, they only satisfy a weak version of the computation rule, which holds up to an identification instead of holding judgmentally. This is related to the problem of *regularity* in semantic models of cubical type theory. As a result, there is no easy way to interpret HoTT into cubical type theory. It is possible to circumvent this problem using ideas of Swan (2018), who showed how to have both path types and identity types in semantic models of cubical type theory.

In practice however, it seems possible to do everything without the judgmental Jβ-rule. This lead Altenkirch, Capriotti, et al. (2017) to pose the following conjecture: is type theory with the judgmental Jβ-rule conservative over type theory with the weak Jβ-rule. More generally, we can consider arbitrary computation rules, and try to compare the judgmental variant with the weak variant. For example, one can consider variants of HoTT or MLTT with weak β- and η- rules for Π- and Σ- types, with universes weakly à

ematical proofs, or for proofs in an axiomatized internal language, we lack the means to interpret the internal constructions in external models. One could imagine that this could be achieved by having the proof assistant for the internal language produce a certificate that is then imported in the proof assistant for the external constructions. Alternatively, one could have a proof assistant that understands the interface between internal and external constructions.

la Tarski, etc. Going in the other direction, one can consider variants of HoTT or MLTT with additional computation rules and definitional equalities, for example by adding a universe of strict propositions (as done by Gilbert, Cockx, Sozeau, and Tabareau (2019)), natural numbers with a strict semi-ring structure, identity types with a strict 1-groupoid structure, etc.

In order to study this family of conservativity conjectures in full generality, it becomes essential to have a formal definition of type theory. The conservativity conjectures can then be stated as properties of morphisms $\mathcal{T}_w \to \mathcal{T}_s$ between a weak type theory $\mathcal{T}_w$ and a strong type theory $\mathcal{T}_s$. Slightly more precisely, assuming that the type theories have homotopical content in the sense of Kapulkin and Lumsdaine (2016), we want $\mathcal{T}_w \to \mathcal{T}_s$ to be a Morita equivalence in the sense of Isaev (2018).

In this thesis, I won't go into the homotopy theory of type theories (except for en passant definitions of some (cofibrations, trivial fibrations) weak factorization systems).

1.2 Type theory as algebraic theories

Traditionally, the syntax of type theory is described by collection of inference rules (including typing rules, conversion and rewriting relations, etc.). This is sometimes called *presyntax*, leaving the denomination of syntax to the quotient of well-typed terms by conversion relations. The semantics are studied using various approaches, including through the definition of categorical models of type theory. Properties of the syntax are typically proven by induction on typing derivations.

A more modern approach is to define the syntax through the semantics; the syntax is defined as the initial model, which is known to exist thanks to general initiality theorems from category theory, and which is unique (up to isomorphism). Equivalently, the syntax can be defined as a QIIT (quotient inductive-inductive type), which are closely related to the general initiality theorems. This syntax can be called *algebraic syntax*, since it is defined as the initial algebra of a (generalized) algebraic theory.

There has been a bit of debate, among type theory researchers, whether defining the syntax through initiality is a satisfactory approach. There is quite a gap between that syntax and the internal syntax used in the implementations of proof assistants. It can be proven that one can recover the initial model by quotienting the extrinsically-typed presyntax by its conversion relation. Streicher (1991) proved this result for a variant of the calculus of constructions. De Boer and Brunerie proved (De Boer 2020; Brunerie 2020), and formalized in Agda, another version of this result for a different type theory. That second proof was a consequence of the Initiality Project (Shulman 2018), an initiative to clarify matters after concerns raised by Voevodsky.

My stance is that the extrinsically-typed syntax is not necessarily closer to what happens in proof assistants, and it is expected that the structures that are most efficient for algorithms and implementations differ from the structures that are most efficient for proofs. As algebraic syntax is well-suited for proofs and presents other advantages, we shall work only with algebraic syntax. Note however that it is possible to reduce some finitary QIITs to quotients of IITs, a result which is similar to initiality theorems; the IITs that are generated when applying these constructions to our preferred notion of model (categories with families) correspond to the syntax with explicit substitutions and explicit coercions of Chapman (2008).

We will consider a class of type theories that can be described by *second-order generalized algebraic theories*, or *SOGATs*. They were introduced by Uemura (2019) and Uemura

(2021). We now review algebraic theories, their generalizations and semantics, building up to SOGATs.

1.2.1 Algebraic theories and generalizations

In universal algebra, algebraic theories are descriptions of mathematical structures consisting of sets equipped with some first-order operations and subject to some equations. Typical examples include monoids, groups, rings, etc. A non-example is the theory of fields: in a field, the inverse operation is partial, being only defined for non-zero elements. This partial operation cannot be part of an algebraic theory, especially because being non-zero is a negative condition.

Algebraic theories are often presented by a syntactic *signature*: a listing of its defining operations and equations. For example, the theory of monoids has the following signature:

$$
\begin{aligned}
&X : \mathsf{Set},\\
&e : X,\\
&m : X \to X \to X,\\
&m(e, x) = x,\\
&m(x, e) = x,\\
&m(x, m(y, z)) = m(m(x, y), z).
\end{aligned}
$$

The above can be read as the usual definition of monoid, but should instead be understood syntactically. The theory of monoids has a sort X, an operation symbol e of arity 0, an operation symbol m of arity 2, and three equations (given by pairs of their left- and right-hand sides, which are first-order terms built from the operations symbols and variables).

In his doctoral thesis, Lawvere (1963) has developed the categorical study of (single-sorted) algebraic theories and functorial semantics. Lawvere's notion of algebraic theory (now called Lawvere theory) is a certain $\mathcal{T}$ with finite limits, an algebra is a functor $\mathcal{T} \to$ **Set** that preserves finite limits, and a morphism of algebras is a natural transformation between such functors. This is a presentation-independent notion of theory: the same Lawvere theory may have multiple presentations, and the Lawvere theory can be reconstructed from its category of algebras.

There is a hierarchy of classes of algebraic theories and generalization thereof, with functorial semantics in various categories.

- Both single-sorted and multisorted algebraic theories have functorial semantics in the 2-category of categories with finite products.

  The finite products are used to construct objects corresponding to arities of operations; a constant $e : X$ becomes a morphism $e : 1 \to X$, a binary operation $m : X \to X \to X$ becomes a morphism $m : X \times X \to X$.

- Essentially algebraic theories have functorial semantics in finitely complete categories.

  Essentially algebraically allow for operations with equational constraints, such as composition in a presentation of categories with a non-dependent sort of mor-

phisms and source and target maps:

$$\begin{aligned}&\mathsf{Ob}, \mathsf{Hom} : \mathsf{Sort},\\ &s, t : \mathsf{Hom} \to \mathsf{Ob},\\ &\mathsf{comp} : (f, g : \mathsf{Hom}) \times (t(f) = s(g))\\ &\quad \to \{h : \mathsf{Hom} \mid s(h) = s(f) \times t(h) = t(g)\}.\end{aligned}$$

Semantically, equalizers are used to interpret the equational constraints, e.g. $(f, g : \mathsf{Hom}) \times (t(f) = s(g))$ is the equalizer of morphisms $s, t : \mathsf{Hom} \to \mathsf{Ob}$.

- Generalized algebraic theories have functorial semantics in clans, which are categories equipped with a class of morphisms called fibrations or display maps, required to be stable under pullbacks.

  Generalized algebraic theories allow for dependent sorts. An example is a definition of categories in which morphisms depend on their source and target:

  $$\begin{aligned}&\mathsf{Ob}, \mathsf{Hom} : \mathsf{Set},\\ &\mathsf{Hom} : \mathsf{Ob} \to \mathsf{Ob} \to \mathsf{Set},\\ &\mathsf{comp} : (x, y, z : \mathsf{Ob}) \times \mathsf{Hom}(x, y) \times \mathsf{Hom}(y, z) \to \mathsf{Hom}(x, z).\end{aligned}$$

  Semantically, the dependent sorts are interpreted by fibrations into their base, e.g. $p_{\mathsf{Hom}} : \mathsf{Hom} \twoheadrightarrow \mathsf{Ob} \times \mathsf{Ob}$. The operations are then interpreted as sections of such fibrations, e.g. comp is interpreted as a section of $\mathsf{Hom} \times_{\mathsf{Ob}} \mathsf{Hom} \twoheadrightarrow \mathsf{Hom}$.

It is sometimes said that essentially algebraic theories and generalized algebraic theories are equivalent because have the same expressive power. What is meant is that a category is the category of algebras of an EAT if and only if it is the category of algebras of a GAT. Any GAT can be translated to an EAT by replacing every dependent sort $B : A \to \mathsf{Set}$ by a non-dependent sort $\Sigma B : \mathsf{Set}$ together with a projection map $p_B : \Sigma B \to A$. An operation

$$f : (a : A) \to B(a) \to C(a)$$

then corresponds to a partial operation

$$f : (a : A)(b : \Sigma B) \to (p_B(b) = a) \to \{c : C \mid p_B(c) = a\}.$$

Conversely, any EAT can be translated to a GAT by introducing an equality sort $\mathsf{Eq}_A : A \to A \to \mathsf{Set}$ for every sort A, together with rules inducing a logical equivalence $\mathsf{Eq}_A(x, y) \cong (x = y)$.

However, multiple non-equivalent GATs can correspond to the same EAT; even though the Set-valued algebras are the same, algebras valued in categories with a class of fibrations (say simplicial sets) can be different, because the dependent sorts of a GAT need to be sent to fibrations. Thus the distinction between EATs and GATs becomes important when considering homotopical structures.

We can also understand this from the point of view of the "language of the theory". The idea of the "language of category theory" is that since mentioning equalities between objects breaks invariance under isomorphisms between objects and under equivalences of categories, it should not be possible to even talk about equalities between objects. This corresponds to the fact that the GAT of categories does not include a sort of equalities

between objects. Because we want equalities between morphisms to be part of the language of category theory, we include it in the presentation of the GAT of categories.

The informal notion of "language of a theory" can be made formal in various ways. One such formalization is the first-order logic with dependent sorts (FOLDS) of Makkai (1995). Henry (2020) explained how GATs, FOLDS and homotopy theory are related.

1.2.2 Generalized algebraic presentations of type theories

Type theories can be presented by generalized algebraic theories. More precisely, there are notions of models of type theory that are the algebras of some GAT, and the syntax of type theory is the initial algebra of that GAT.

Many definitions of categorical models of type theory exist in the literature. They can be classified along two dimensions:

1. Whether types are additional structure (there is a set of types over any contexts) or specified by a class of maps (which called display maps; types over a context are then identified with display maps into that context). In the former case, substitution is strictly functorial, whereas in the latter case substitution is usually only pseudo-functorial.

2. Whether the objects of the underlying category are generated by the empty context and extensions, in which case we say that the model is contextual (Cartmell 1986) or democratic (Clairambault and Dybjer 2014).

This gives a partition of the notions of models into four classes.

- In Categories with Families (Dybjer 1995) and natural models (Awodey 2018), we have sets of types, and we have arbitrary contexts.

- In display map categories (Taylor 1999), types are induced by a class of maps and contexts can be arbitrary.

- In contextual categories (Cartmell 1986), also called C-systems (Voevodsky 2016), in B-systems (Voevodsky 2014) and in democratic CwFs (Clairambault and Dybjer 2014), we have sets of types, and contexts are (either strictly equal to or isomorphic to) iterated extensions of the empty context.

- In clans (Joyal 2017), types induced by a class of maps and contexts coincide with types over the empty context.

There are meaningful differences between the different classes; they are useful for different applications. However different notions within the same classes are equivalent, and the choice between the different notions is mostly a matter of preference.

In this thesis, we will mainly use *categories with families*, abbreviated as *CwF*, which were introduced by Dybjer (1995). They can be presented by a generalized algebraic theory with four sorts (any many operations and equations):

$$
\begin{aligned}
&\mathsf{Ob} : \mathsf{Set}, && \text{(Objects or contexts)}\\
&\mathsf{Hom} : \mathsf{Ob} \to \mathsf{Ob} \to \mathsf{Set}, && \text{(Morphisms or substitutions)}\\
&\mathsf{Ty} : \mathsf{Ob} \to \mathsf{Set}, && \text{(Types)}\\
&\mathsf{Tm} : (\Gamma : \mathsf{Ob}) \to \mathsf{Ty}(\Gamma) \to \mathsf{Set}. && \text{(Terms)}
\end{aligned}
$$

In fact, categories with families do not quite respect the previous discussion about languages of GATs, in the sense that there is a mismatch between the language of the GAT of CwFs and the "language of type theory". Indeed, contexts and substitutions are not really part of the language of type theory, but are included because they are necessary to describe dependent types.

Because of this, we sometimes also have to consider a second notion of categorical models: contextual CwFs (introduced as contextual categories by Cartmell (1986), also called C-systems by Voevodsky (2016)). Contextual CwFs can also be presented as a generalized algebraic theory, which has the disadvantage of requiring an infinite amount of sorts and operations. However the language of the GAT of contextual CwFs is a more accurate description of the "language of type theory". From some point of view, contextual CwFs are morally the correct notion of model, but CwFs are more convenient.

At the homotopical level, this phenomenon can be seen in the fact that Kapulkin and Lumsdaine (2016) construct a (left semi)-model structure over contextual CwFs, rather than over CwFs.

Another possible approach could be to define CwFs as a theory in which the sorts Ty and Tm are fibrant, but the sorts Ob and Hom are not. This would require an alternative notion of theory, that lies somewhere between EATs and GATs. We won't explore this other notion of theory.

1.2.3 Second-order algebraic theories and higher-order abstract syntax

Second-order algebraic theories, introduced by Fiore and Mahmoud (2010) (after earlier work by Fiore and Hur (2010)), are a generalization of algebraic theories that is meant to describe languages with bindings. This is achieved by allowing second-order operations, whose domain is a first-order arity.

For example, untyped lambda calculus can be presented by the following signature:

$$\begin{aligned}
&\mathsf{tm} : \mathsf{Sort},\\
&\mathsf{app} : \mathsf{tm} \to \mathsf{tm} \to \mathsf{tm},\\
&\mathsf{lam} : (\mathsf{tm} \to \mathsf{tm}) \to \mathsf{tm},\\
&\beta : \mathsf{app}(\mathsf{lam}(b), a) = b(a),\\
&\eta : \mathsf{lam}(\lambda x \mapsto \mathsf{app}(f, x)) = f.
\end{aligned}$$

Here lam is a second-order operation, whose domain $(\mathsf{tm} \to \mathsf{tm})$ is first-order, indicating that lam binds a variable.

Second-order operations are closely related to the ideas of logical frameworks (Harper, Honsell, and Plotkin 1993), which are also frameworks used for the definition of syntax with bindings, in which the binders are represented by functions in the logical framework. The ideas behind logical frameworks is also referred to as higher-order abstract syntax (HOAS).

The sort tm in the above signature should not be seen as a set of terms, and $(\mathsf{tm} \to \mathsf{tm})$ should not be seen as a function between sets. Instead, it should only be possible to construct elements of $(\mathsf{tm} \to \mathsf{tm})$ that correspond to terms over an additional bound variable. A semantic explanation of this phenomenon was given by Hofmann (1999): tm is interpreted by a representable presheaf over a category of contexts. Hofmann observes that the exponential presheaf $(\mathsf{tm} \to \mathsf{tm})$ then admits a simple description: $(\mathsf{tm} \to \mathsf{tm})(\Gamma) = \mathsf{tm}(\Gamma \times T)$, where T is the object representing tm. In other words, elements of $(\mathsf{tm} \to \mathsf{tm})$ at a context Γ really correspond to terms in an extended context $\Gamma \times T$. This generalizes to dependently-typed settings.

The restriction from higher-order to second-order is due to the fact that we have to specify ahead of time which sorts can be used to bind variables; just because we can bind variables of sort tm does not mean that we can bind variables of sort $(\mathsf{tm} \to \mathsf{tm})$. What is gained from this restriction is the ability to define not only syntax, but a whole category of algebras.

The second-order generalized algebraic theories of Uemura (2019) generalize second-order algebraic theories by allowing dependent sorts. In SOGATs, we also have a distinction between general sorts and representable sorts, only the representable sorts can be used for variables in binders.

As a SOGAT, the signature for a dependent type theory with Π-types is as follows:

$$
\begin{aligned}
&\mathsf{ty} : \mathsf{Sort},\\
&\mathsf{tm} : \mathsf{ty} \to \mathsf{Sort}_{\mathsf{rep}},\\
&\Pi : (A : \mathsf{ty})(B : \mathsf{tm}(A) \to \mathsf{ty}) \to \mathsf{ty},\\
&\mathsf{app} : (A : \mathsf{ty})(B : \mathsf{tm}(A) \to \mathsf{ty}) \to \mathsf{tm}(\Pi(A, B)) \to (a : \mathsf{tm}(A)) \to \mathsf{tm}(B(a)),\\
&\mathsf{lam} : (A : \mathsf{ty})(B : \mathsf{tm}(A) \to \mathsf{ty}) \to ((a : \mathsf{tm}(A)) \to \mathsf{tm}(B(a))) \to \mathsf{tm}(\Pi(A, B)),\\
&\beta : \mathsf{app}(A, B, \mathsf{lam}(A, B, b), a) = b(a),\\
&\eta : \mathsf{lam}(A, B, \lambda x \mapsto \mathsf{app}(A, B, f, x)) = f.
\end{aligned}
$$

As only tm is a representable sort, this type theory supports extensions of contexts by term variables $((\Gamma, a : A))$, but not by type variables $((\Gamma, A\ \mathsf{type}))$.

Uemura defined the semantics of SOGATs as functorial semantics in the 2-category of *representable map categories*, which are finitely complete categories equipped with an additional class of representable maps, corresponding to the representable sorts.

Any SOGAT can also be seen as a higher-order theory, in which there are no restriction on the order of operations. These have functorial semantics in LCCCs. Gratzer and Sterling (2020) have proposed to use LCCCs as a simpler framework for the general semantics of type theory. While functorial semantics in LCCCs is indeed simpler than functorial semantics in representable map categories, and sufficient for many applications, higher-order theories do not really have categories of algebras; there are many aspects of the semantics of SOGATs that cannot be studied at the level of higher-order theories.

1.2.4 Other general definitions of dependent type theory

In addition to SOGATs, other general definitions of dependent type theory have been proposed and studied.

Isaev (2016) gives a general definition of type theory as a class of essentially algebraic theories extending the essentially algebraic theory of contextual categories, with the constraint that all added symbols interact with substitution.

Haselwarter and Bauer (2023) give a general definition of type theory (based on earlier work by Bauer, Haselwarter, and Lumsdaine (2020)) based on the presentation of type theories with inference rules. The goal is to have a general definition that is closely related to the traditional definitions of type theories using typing relations, yet can be used to prove general metatheorems (uniqueness of typing, etc.) that are often tedious to prove for concrete type theories.

Previous approaches, notably logical frameworks, could be used to specify the syntax of dependent type theories, but not the semantics. Moreover, they were not seen as a way to define the syntax, but rather as a way to represent it; adequacy theorems would

be proven, providing a (bijective) correspondence between the components of the LF and conventionally defined syntax.

1.3 The metatheory of type theories, algebraically

Once a precise general definition of type theory is chosen, one wants to use it to prove metatheoretic results about the syntax and semantics of type theory. Metatheory generally refers to any theorem proven externally about the type theory (as opposed to theorems proven internally to the type theory). More specifically, it often refers to the properties of the syntax that are needed to justify that a type theory is well-behaved.

- *Consistency*: there is no closed term of type $\bot$.
- *Canonicity*: every closed term of the boolean type or natural number type computes to an actual boolean or an actual natural number.
- *Normalization*: every term (in any context) admits a unique normal form (for a suitable notion of normal form).

  This has to be distinguished from weak normalization and strong normalization, which only make sense for type theories equipped with a reduction relation on (pre)terms.
- Some properties are often obtained as a consequence of normalization:
  - Decidability of equality.
  - Decidability of type checking.
  - Injectivity of type formers.

We want to establish these properties algebraically, meaning that we can only rely on the initiality of the syntax. In order to prove anything about the syntax, one has to construct another model, interpret the syntax into that model using initiality, and derive the desired property from the interpretation.

The main method that has emerged for constructing the relevant models involves logical relations and categorical gluing.

- *Logical relations* involve an interpretation of a type A as a family $\mathsf{Tm}(-, A) \to \mathsf{Set}$ indexed by terms of type A over some context. The family can be a family of sets as above, a family of propositions $\mathsf{Tm}(-, A) \to \mathsf{Prop}$, or a family valued in another semantic domain.
- *Categorical gluing* involves the construction of a model whose underlying category is a comma category associated to a functor $F : \mathcal{C} \to \mathcal{D}$. Typically $\mathcal{C}$ is a syntactic category and $\mathcal{D}$ is a more semantic category. Then the objects of the glued category are triples (x, y, α), where $x \in \mathcal{C}$, $y \in \mathcal{D}$ and $\alpha : y \to F(x)$. This is a way to glue together syntax and semantics.
- *Sconing* is a special instance of categorical gluing over the global sections functor

  $$\Gamma \mapsto \mathcal{C}(1, \Gamma).$$

  This means that objects are interpreted by triples (Γ, Y, p) where Y is a set and

  $$p : Y \to \mathcal{C}(1, \Gamma).$$

Equivalently, objects can be seen as pairs (Γ, Γ') where

$$\Gamma' : \mathcal{C}(1, \Gamma) \to \mathsf{Set}$$

is a family of sets. Elements $\gamma : \mathcal{C}(1, \Gamma)$ are "closing substitution", and elements $\gamma' : \Gamma'(\gamma)$ can often be seen as "evaluation environments"; assigning a semantic value to every variable in Γ.

Categorical gluing and logical relations are closely related, and arguably the same construction if types are seen as objects in a slice category. They will however be considered separately in this thesis; logical relations involve types while gluing involves contexts.

For non-dependent type systems, the logical relations are usually defined by induction on the structure of types prior to the definition of the rest of the model. In dependent type theory, the interpretation of types has to be given mutually with the rest of the model.

Algebraic or reduction-free proofs of normalization have been given for simply typed lambda calculus by Altenkirch, Hofmann, and Streicher (1995) and for system F by the same authors (1997). An algebraic proof of normalization for a dependent type theory without universes was given by Altenkirch and Kaposi (2017), and also formalized in Agda.

Coquand (2019) gave an algebraic and reduction-free proof of normalization for a dependent type theory with universes. The normalization proof involves the construction of a gluing model.

The construction of these gluing models requires in principle to check many naturality conditions: all components of the model need to respect substitutions, and moreover many of the subconstructions need to be stable under renamings. While checking these conditions can be seen as routine verifications, they are not easy to check fully explicitly. It also seemed that Coquand's model could be factored as the composition of multiple constructions. This has led type theorists to seek alternative ways to present the model, that would isolate the core of the normalization proof from the rest of the construction; with the hope of obtain tools that could be reused for other applications.

Naturality and functoriality conditions can often be checked automatically by using the internal language of presheaf toposes. In this case, we want to use both presheaves over the syntactic category, and presheaves over its category of renamings.

Sterling (2022) has developed *synthetic Tait computability* (STC). This involves the internal language of a glued topos, which contains two base toposes as open/closed subtoposes. In the internal language of the glued topos, one has access to a pair of open/closed modalities, which can be used to access the internal languages of the two subtoposes and express constructions that involve interactions between the two subtoposes. This has notably been employed to prove normalization for cubical type theory (Sterling and Angiuli 2021).

1.3.1 Other approaches

Other approaches to the metatheory of type theory exist, that rely on other, usually non-algebraic, presentations of the syntax and semantics of type theory.

It should be noted that (strong) normalization and reduction-free normalization are not the same result, and normalization results for different presentations of the syntax are not the same result; different normalization theorems are not trivially interderivable.

A proof of normalization for dependent type theory with a predicative hierarchy of universes and dependent function types with both β- and η- rules was given in the Habilitation thesis of Abel (2013). The proof involves normalization by evaluation into an inductively defined semantic domain, and partial equivalence relations on that semantic domain.

Abel, Öhman, and Vezzosi (2017) give another proof of normalization and of the decidability of equality. Their proof is also fully formalized in Agda.

Adjedj et al. (2024) give a proof similar to the one of Abel, Öhman, and Vezzosi (2017), and formalize it in Coq. In particular, they present a way to remove induction-recursion from the proof, thus proving the result in a weaker metatheory.

While proofs of strong normalization and proofs of reduction-free normalization both use logical relations, the logical relations do not quite serve the same purpose. In presence of a deterministic reduction relation, the computational behavior is already fully specified by the reduction, and the goal is to prove that reduction terminates and that equality of normal forms coincides with equality of terms. The logical relations are used to prove these properties. In the reduction-free setting, the logical relations are used to construct a function that assigns a normal form to every term; thus it plays the role of both the reduction relation and the logical relation from the reduction-full setting.

1.4 Overview of the thesis

I now present the main contributions of the thesis.

1.4.1 A 1-categorical presentation of functorial semantics

I will present the semantics of GATs as functorial semantics in the 1-category of Σ-CwFs (which are the models of a dependent type theory with Σ- and $\mathbf{1}$- types). A more usual presentation of functorial semantics would take place in a 2-category of categories with additional structure, such as the 2-category of clans.

The main difference lies in the choice between preserving structure, such as finite limits, weakly or strictly. In the 1-category of Σ-CwFs, the Σ-types are preserved strictly, while the corresponding finite limits is preserved weakly by morphisms of clans.

Ultimately, the two possible presentations are related by the fact that Σ-CwFs ought to be a 1-categorical presentation of the 2-category of clans, in the sense that there should be a model structure on $\mathbf{CwF}_\Sigma$ that presents the $(2,1)$-category of clans, and $\mathbf{CwF}_\Sigma$ has arrow-objects that classify the additional (non-invertible) 1-cells in the 2-category of clans.

The choice of working with 1-categorical notions was originally motivated by the fact that $\mathbf{CwF}_\Sigma$ is itself the category of algebras of a GAT (and of a SOGAT). This means that we can reuse the semantics of general GATs to study $\mathbf{CwF}_\Sigma$.

In addition to the original motivation, this proved to have other advantages:

- In practice, GATs are not defined by explicitly constructing Σ-CwFs, but by writing a signature S that presents a Σ-CwF $\mathbf{0}_{\mathbf{CwF}_\Sigma}[S]$. In particular, we will see how QIIT-signatures from Kaposi, Kovács, and Altenkirch (2019) present GATs. Because signatures are usually syntactic and quite strict, having a stricter notion of theory makes their interpretation easier.

  Moreover, $\mathbf{0}[S]$ then satisfies a 1-categorical property that determines $\mathbf{0}[S] \to \mathcal{E}$ up to isomorphism (of sets), rather than up to equivalence of category. Thanks

to this, the category of algebras of the theory is determined up to isomorphism, rather than up to equivalence, and its components can be computed exactly from the structure of the signature S. This can be relevant when doing computations for a concrete signature S.

- One is often interested in functors involving $\mathbf{CwF}_\Sigma$, and in their left or right adjoints when they exist. In particular, translations between theories (e.g. from SOGATs to GATs, or from a GAT $\mathcal{T}$ to the GAT of morphisms between $\mathcal{T}$-algebras, etc.) can be described by such adjunctions. Working 1-categorically means that it is easier to obtain these adjunctions, which are often themselves induced by morphisms between GATs.

- One may be interested in generalizing GATs to ∞-EATs or ∞-GATs. These should have higher-categorical functorial semantics in finitely complete ∞-categories, or variants thereof. Finitely complete ∞-categories can be described 1-categorically as CwFs with Σ- and Id- types. By considering the 1-category $\mathbf{CwF}_\Sigma$, it makes it easier to compare GATs and ∞-GATs, by comparing $\mathbf{CwF}_\Sigma$ and $\mathbf{CwF}_{\Sigma,\mathsf{Id}}$.

The price to pay is that not every $\mathbf{CwF}_\Sigma$ can be seen as a GAT. Morphisms in $\mathbf{CwF}_\Sigma$ are only well-behaved when they have a cofibrant source. Because the functorial semantics are defined through morphisms out of the theory, we need to restrict the theories to objects that are cofibrant. Moreover, some useful functors between $\mathbf{CwF}_\Sigma$ are only pseudo-morphisms, preserving the structure up to isomorphism. In such cases, we will have to replace them by strict morphisms (which is possible when the source is cofibrant).

1.4.2 Reduction from SOGATs to GATs

Uemura presents the functorial semantics of SOGATs in the 2-category of representable map categories. A SOGAT is a (presentation of a) representable map category $\mathcal{T}$. A model of $\mathcal{T}$ is a category together with a representable map functor $\mathcal{T} \to \mathbf{DF}_\mathcal{C}$, where $\mathbf{DF}_\mathcal{C}$ is the representable map category of discrete fibrations over $\mathcal{C}$. Two equivalent definitions of morphism of models are given by Uemura (2019):

- The first definition explicitly quantifies over all objects and morphisms in $\mathcal{T}$, with an additional condition for representable maps.

- The other definition makes use of a generalization of $\mathbf{DF}_\mathcal{C}$: for any functor $F : \mathcal{C} \to \mathcal{D}$, there is a representable map category $\mathbf{DF}_F^{\to}$ that lies over $\mathbf{DF}_\mathcal{C}$ and $\mathbf{DF}_\mathcal{D}$. A morphism is then a representable map functor $\mathcal{T} \to \mathbf{DF}_F^{\to}$ that lies over models $\mathcal{T} \to \mathbf{DF}_\mathcal{C}$ and $\mathcal{T} \to \mathbf{DF}_\mathcal{D}$.

  More generally it is possible to construct $\mathbf{DF}^I$ for any indexing category I; representable map functors $\mathcal{T} \to \mathbf{DF}^I$ correspond to I-indexed diagrams of models, lying over an I-indexed diagram of categories.

We will present the functorial semantics of SOGATs by reduction to the functorial semantics of GATs. For any SOGAT $\mathcal{T}$, we construct a GAT $\mathcal{T}^{\mathsf{fo}}$ that is an extension of the GAT of categories with a terminal object. This has a universal property: algebras of $\mathcal{T}^{\mathsf{fo}}$ into any Σ-CwF $\mathcal{E}$ with an internal category $\mathbb{C}$ are in bijective correspondence with morphisms from $\mathcal{T}$ to $\mathsf{psh}_\mathcal{E}(\mathbb{C})$, which is the $(\Sigma, \Pi_{\mathsf{rep}})$-CwF of internal presheaves in $\mathcal{E}$ over $\mathbb{C}$. In other words, $\mathcal{T} \mapsto \mathcal{T}^{\mathsf{fo}}$ is a left adjoint of $\mathsf{psh}_\_(-)$ While $\mathcal{T}$ classifies its second-order models, $\mathcal{T}^{\mathsf{fo}}$ is a first-order GAT classifying first-order algebras of $\mathcal{T}$.

The $(\Sigma, \Pi_{\mathsf{rep}})$-CwF $\mathsf{psh}_{\mathcal{E}}(\mathbb{C})$ plays the same role as $\mathbf{DF}^I$ in the functorial semantics. However, having the left adjoint additionally ensures that all of the semantic notions developed for GATs can be lifted to SOGATs. When developing the semantics of SOGATs, we can then focus on aspects that are specific to the second-order case.

An alternative reduction from SOGATs to GATs at the level of signatures (QIIT-signatures for GATs and their generalization to SOGATs) was given by Kaposi and Xie (2024).

1.4.3 Internal algebras of SOGATs and relative induction principles

Because any presheaf category is a model of extensional type theory, it is possible to replace constructions performed "externally" by constructions in the "internal" language of a presheaf category. For the purpose of doing the metatheory of type theory algebraically, the internal language of presheaf categories proves to be an essential tool, for multiple reasons. One reason is that syntactic constructions are typically stable under substitutions or under some other class of morphisms; by working in an internal language, these stability conditions can be ignored. Another reason is that complex external constructions can correspond to constructions performed in the internal language that are simpler, or instances of better known constructions.

Thanks to the reduction from SOGATs to GATs, one can consider algebras of a SOGAT that are internal to an arbitrary Σ-CwF $\mathcal{E}$, not only **Set**. In particular, one can consider algebras internal to presheaf categories $\mathbf{Psh}(\mathcal{C})$.

A general result says that algebras internal to a presheaf categories are in bijective correspondence with presheaves of algebras, i.e. functors

$$\mathcal{C}^{\mathsf{op}} \to \mathbf{Alg}_{\mathcal{T}}.$$

This is used as an interface between internal and external constructions, as internal algebras make sense in the internal language of presheaf categories, while presheaves of algebras can be easier to construct and compose externally.

The idea of relative induction principles is to prove universal properties for certain internal algebras, either among internal algebras or in functor categories $\mathcal{C}^{\mathsf{op}} \to \mathbf{Alg}_{\mathcal{T}}$.

The internal algebras $\mathcal{S}_F$ that are involved are determined by a functor $F : \mathcal{C} \to \mathcal{S}$, where $\mathcal{S}$ is the initial model of the type theory under consideration (an arbitrary SOGAT).

The internal algebra $\mathcal{S}_F$ has the property that its closed components (closed types and terms) correspond to open components of $\mathcal{S}$ over contexts of the form $F(\Gamma)$. More specifically, seeing $\mathcal{S}_F : \mathcal{C}^{\mathsf{op}} \to \mathbf{Alg}_{\mathcal{T}}$ as a presheaf of algebras, there is a natural isomorphism

$$\mathcal{S}_F(\Gamma).\mathsf{Ty}(1) \cong \mathcal{S}.\mathsf{Ty}(F(\Gamma))$$

between closed types and open types, and similarly for other components.

Once we have a universal property for $\mathcal{S}_F$, one can prove properties of $\mathcal{S}_F$ using methods that are normally used to prove properties of closed components, in the internal language of $\mathbf{Psh}(\mathcal{C})$. In particular, categorical sconing, which is typically used to prove properties of closed components such as canonicity, can be used to prove properties of $\mathcal{S}_F$. Thanks to the correspondence between closed components of $\mathcal{S}_F$ and open components of $\mathcal{S}$, we can obtain properties of open components of $\mathcal{S}$, such as normalization.

I discovered while writing this introduction that the core idea of replacing gluing by internal sconing had appeared in a note by Altenkirch, Hofmann, and Streicher 1997.

1.4.4 Applications of relative induction principles

Normalization proofs

As an application of the relative induction principles over renamings, we will give multiple proofs of normalization for dependent type theory, and of its main corollary, decidability of equality.

- We prove normalization for Martin-Löf Type Theory, in which we include a hierarchy of universes, closed under Π-types, Σ-types, Id-types, W-types, a type of booleans and an empty type.
- We then investigate normalization proofs for extensions of MLTT with definitional algebraic structures. More specifically, we consider MLTT with list types with a definitional monoid structure, meaning that the monoid laws

$$\begin{aligned}&\mathsf{nil} +\!\!+ x = x,\\ &x +\!\!+ \mathsf{nil} = x,\\ &(x +\!\!+ y) +\!\!+ z = x +\!\!+ (y +\!\!+ z)\end{aligned}$$

  hold up to definitional equality. We also include two eliminators, which compute respectively on left-nested lists and right-nested lists.

  The main idea of the normalization proof is that normal forms of these lists types should be semantic lists of neutral lists. The same idea was also proposed by Corbyn et al. (2022).

  Lists with a definitional monoid structure were studied by Allais, McBride, and Boutillier (2013) (in the simply typed setting, but their methods would extend to the dependently typed setting). Our presentation is however more general: most of the construction can be generalized to other algebraic theories: one just replaces the semantic lists by semantic free algebras of the corresponding algebraic structure. In particular, the presented method can deal with algebraic structures with commutative operations (e.g. definitional commutative monoids), and with some additional ad-hoc eliminators, such as a reversal operation on lists, with the expected rules holding definitionally.

  There are two places where the choice of algebraic theory intervenes: normalization for eliminators and the decidability of equality for normal forms. Deciding equalities between normal forms requires some properties of the algebraic theory, notably that the algebraic theory itself is decidable (for any set with decidable equality, the free algebra over that set has decidable equality).

  At the moment, it is not clear whether the proposed normalization strategy can be used for algebraic theories with non-linear equations (such as groups, which include the non-linear equation $(x - x = 0)$). In presence of both non-linear equations and eliminators, normalization needs to be interleaved with equality checking, which cannot be handled by the normalization proof (although it seems unproblematic when implementing a type-checking algorithm).

- We also show how to combine the definitional algebraic structures with definitional functoriality, which was also covered by the work of Allais, McBride, and Boutillier (2013) and further studied by Laurent, Lennon-Bertrand, and Maillard (2024), whose work we build on.

- Previous algebraic proofs of normalization that made use of internal languages would prove normalization in the internal language, and then deduce the external decidability of equality. This requires a tedious comparison between internal and external definitions of normal forms. We simplify this step by proving the decidability of equality internally, relying on an axiomatization of the levelwise decidable propositions.

Conservativity of two-level type theory

As another application of relative induction principles, we prove the conservativity of two-level type theory over its inner theory.

The conservativity of two-level type theory over its inner theory was conjectured by Annenkov, Capriotti, Kraus, and Sattler (2023) and Capriotti (2017), but only a weaker version was proven. It was proven by Kovács (2022). We prove a slightly more general version: instead of looking at morphism $\mathbf{0}_{\mathsf{inner}} \to \mathbf{0}_{\mathsf{2ltt}}$ between initial models, we look at the morphism $\mathcal{T}_{\mathsf{inner}} \to \mathcal{T}_{\mathsf{2ltt}}$ between SOGATs. As a consequence, we obtain properties of morphisms $\mathcal{C}_{\mathsf{inner}} \to \mathsf{Free}_{\mathsf{2ltt}}(\mathcal{C}_{\mathsf{inner}})$ for arbitrary models of the inner theory. We recover the original conservativity when $\mathcal{C}_{\mathsf{inner}}$ is the initial model.

The result we prove is similar to the adequacy of locally cartesian closed categories over representable map categories, proven by Gratzer and Sterling (2020).

In general, two-level type theory can be defined with any SOGAT as the inner theory and almost any SOGAT as the outer theory. We will consider two-level type theory with MLTT as the outer theory, and an arbitrary SOGAT as the inner theory.

As an application of relative induction principles, the proof is quite interesting, because we need to perform induction over both the inner theory and the two-level type theory, with interactions between the two applications of the induction principles.

Chapter 2

Categorical preliminaries

2.1 Notations and required background

This thesis assumes that the reader is comfortable with basic concepts in category theory: categories, functors, limits and colimits, universal properties, the Yoneda lemma, etc.

We will attempt to give the categorical definitions in such a way that functoriality and naturality conditions are obtained for free, notably by making use of classifying objects and generic operations.

For example, the arrow category $\mathbf{Arr}(\mathcal{D}) = [\{\underline{x} \to \underline{y}\}, \mathcal{D}]$ classifies natural transformations, i.e. a natural transformation $\alpha : F \Rightarrow G$ is equivalently a functor

$$\alpha : \mathcal{C} \to \mathbf{Arr}(\mathcal{D})$$

that restricts to the functors F and G after composition with $\pi_1, \pi_2 : \mathbf{Arr}(\mathcal{D}) \to \mathcal{D}$. The advantage of such a definition is that basic operations on natural transformations can be obtained by composing functors, e.g. pre-composition with $\mathcal{C}' \to \mathcal{C}$, post-composition $\mathbf{Arr}(\mathcal{D}) \to \mathbf{Arr}(\mathcal{D}')$, or with functors $[I, \mathcal{D}] \to [J, \mathcal{D}]$ induced by functors $J \to I$. For example, the vertical composition of natural transformations is determined by the functor

$$\{\underline{x} \to \underline{z}\} \to \{\underline{x} \to \underline{y} \to \underline{z}\},$$

which is the "generic" morphism composition.

We sometimes write $(f : x \to y) \in \mathcal{C}$ instead of $f : \mathcal{C}(x, y)$.

Objects that are free algebras or free extensions of algebras will be written using a generalization of the notation $R[\underline{X}]$ for polynomial rings. For example, given a category $\mathcal{C}$ and objects $x, y \in \mathcal{C}$, we write $\mathcal{C}[\underline{f} : x \to y]$ for the free extension $\mathcal{C}$ by a new morphism $\underline{f} : x \to y$. Its universal property says that a functor $\mathcal{C}[\underline{f} : x \to y] \to \mathcal{D}$ is uniquely determined by a functor $F : \mathcal{C} \to \mathcal{D}$ along with a morphism $(f : F(x) \to F(y)) \in \mathcal{D}$. Similarly, $\mathcal{C}[\Gamma \vdash \underline{A}\ \mathsf{type}]$ would be the free extension of a model $\mathcal{C}$ of type theory by a new type $\underline{A}$ in context Γ. We use underlines to distinguish the newly adjoined elements from the pre-existing elements.

We use $\langle - \rangle$ for morphisms induced by universal properties on the right (usually products or limits), and $[-]$ for morphisms induced by universal properties on the left (usually coproducts or colimits). For example, a morphism $[f; g] : (A + B) \to X$ is induced by morphisms $f : A \to X$ and $g : B \to X$. A morphism $\langle f, g \rangle : A \to (X \times Y)$ is induced by morphisms $f : A \to X$ and $g : B \to Y$.

2.2 Displayed categories

The notion of displayed category over a base category $\mathcal{C}$, due to Ahrens and Lumsdaine (2019), is equivalent to the data of a category $\mathcal{D}$ together with a functor $p : \mathcal{D} \to \mathcal{C}$. The main original motivation for displayed categories is that they allow for definitions of categorical notions, notably notions of categorical fibrations, without referring to equalities between objects.

Another motivation for the use of displayed categories (and other displayed structures) is that they can lead to more compositional and modular definitions. Indeed, displayed categories can be substituted over a functor between base categories, and this substitution is strictly functorial. This substitution corresponds to pullbacks of a functor $p : \mathcal{D} \to \mathcal{C}$ along a base functor $\mathcal{C}' \to \mathcal{C}$, but in general these pullbacks are only pseudo-functorial.

Notions of displayed algebras also exist for theories other than the theory of categories. The notion of displayed algebra corresponds exactly to the premises (motives and methods) of an induction principle. In categories of algebras of algebraic theories, the universal property of the initial object is equivalent to the fact that any displayed algebra over it admits a section.

Definition 2.2.1. A **displayed category** $\mathcal{D}$ over a base category $\mathcal{C}$ consists of the following data:

- For every object $x \in \mathcal{C}$, a set $\mathcal{D}[x]$ of displayed objects over x.
- For every morphism $f : \mathcal{C}(x, y)$, a family $\mathcal{D}[f] : \mathcal{D}[x] \to \mathcal{D}[y] \to \mathsf{Set}$ of displayed morphisms over f. We may write $(f' : x' \to y') \in \mathcal{D}[f]$ when $f' : \mathcal{D}[f](x', y')$.
- We have displayed identities and a displayed composition operation: given $x' \in \mathcal{D}[x]$, we have $(\mathsf{id} : x' \to x') \in \mathcal{D}[\mathsf{id}_x]$ and given $(f' : y' \to z') \in \mathcal{D}[f]$ and $(g' : x' \to y') \in \mathcal{D}[g]$, we have $(f' \circ g' : x' \to z') \in \mathcal{D}[f \circ g]$.
- Such that displayed identity and associativity laws hold: $\mathsf{id} \circ f' = f' = f' \circ \mathsf{id}$ and $(f' \circ g') \circ h' = f' \circ (g' \circ h')$. ⌟

Example 2.2.2. The displayed category of elements **El** is displayed over the category **Set**.

- A displayed object over a set X is an element $x : X$.
- A displayed morphism from x to y over $f : X \to Y$ is a witness of the equality $f(x) = y$. ⌟

The terminology "category of elements" for **El** clashes with the standard notion of category of elements of a functor $F : \mathcal{C} \to \mathbf{Set}$. This should not cause too much confusion, as the two notions are closely related. We will use **El** to define the category of elements $\int \mathbf{El}[F]$ in example 2.2.9.

Example 2.2.3. The category of families **Fam** is displayed over the category **Set**.

- A displayed object over a set X is a family $X' : X \to \mathsf{Set}$.
- A displayed morphism over $f : X \to Y$ is a dependent function

$$f' : (x : X) \to X'(x) \to Y'(f(x)).$$ ⌟

Example 2.2.4. There is a category **Func** of functions, displayed over **Set** $\times$ **Set**.

- A displayed object of **Func** over (X_1, X_2) is a function $\alpha_X : X_1 \to X_2$.
- A displayed morphism over $(f_1 : X_1 \to Y_1, f_2 : X_2 \to Y_2)$ is a witness of the equality $\alpha_Y \circ f_1 = f_2 \circ \alpha_X$.

More generally, there is an arrow category $\mathbf{Arr}(\mathcal{E})$ displayed over $\mathcal{E} \times \mathcal{E}$ for any base category $\mathcal{E}$. ⌟

Example 2.2.5. There is a category **Sect** of sections, displayed over **Fam**.

- A displayed object of **Sect** over (X, X') is a dependent function

$$\alpha_X : (x : X) \to X'(x).$$

- A displayed morphism over $(f : X \to Y, f' : (x : X) \to X' \to Y'(f(x)))$ is a witness of the equalities

$$(x : X) \to f'(\alpha_X(x)) = \alpha_Y(f(x)).$$ ⌟

Construction 2.2.6. Any displayed category $\mathcal{D}$ over a base $\mathcal{C}$ has a **total category** $\int \mathcal{D}$ and a **projection functor** $\pi_{\mathcal{D}} : (\int \mathcal{D}) \to \mathcal{C}$.

- An object of $\int \mathcal{D}$ is a pair (x_c, x_d) where $x_c \in \mathcal{C}$ and $x_d \in \mathcal{D}[x_c]$.
- A morphism of $\int \mathcal{D}$ from (x_c, x_d) to (y_c, y_d) is a pair (f_c, f_d) where $(f_c : x_c \to y_c) \in \mathcal{C}$ and $(f_d : x_d \to y_d) \in \mathcal{D}[f_c]$.

The projection functor $\pi_{\mathcal{D}} : (\int \mathcal{D}) \to \mathcal{C}$ projects the first components of these pairs. ⌟

In practice we will often identify a displayed category with its total category. When a functor should be seen as the projection functor out of the total category of a displayed category, we sometimes use the distinctive arrow $\pi : \mathcal{D} \twoheadrightarrow \mathcal{C}$. In diagrams, these arrows are almost always written vertically: the total category $\mathcal{D}$ should be seen as lying over the base category $\mathcal{C}$.

Definition 2.2.7. A **section** of a displayed category $\mathcal{D}$ over a base category $\mathcal{C}$ consists of a function sending any object $x \in \mathcal{C}$ to an object $S(x) : \mathcal{D}[x]$, of a function sending any morphism $(f : x \to y) \in \mathcal{D}$ to a morphism $(S(f) : S(x) \to S(y)) \in \mathcal{D}[f]$, subject to the functoriality laws $S(\mathsf{id}) = \mathsf{id}$ and $S(f \circ g) = S(f) \circ S(g)$.

Sections of $\mathcal{D}$ are in bijective correspondence with sections of the projection functor $\pi_{\mathcal{D}} : (\int \mathcal{D}) \twoheadrightarrow \mathcal{C}$:

$$\begin{array}{c} \int \mathcal{D} \\ S \uparrow \; \downarrow \pi_{\mathcal{D}} \\ \mathcal{C} \end{array}$$

⌟

Definition 2.2.8. Given a functor $F : \mathcal{A} \to \mathcal{C}$ and a displayed category $\mathcal{D}$ over $\mathcal{C}$, there is a **restricted displayed category** $\mathcal{D}[F]$ over $\mathcal{A}$.

- A displayed object of $\mathcal{D}[F]$ over $x \in \mathcal{A}$ is a displayed object of $\mathcal{D}$ over $F(x)$.
- A displayed morphism of $\mathcal{D}[F]$ over $(f : x \to y) \in \mathcal{A}$ is a displayed object of $\mathcal{D}$ over $F(a)$.

This construction is strictly functorial in F.

The total category of $\mathcal{D}[F]$ is a pullback of $\int \mathcal{D}$ over F:

$$\begin{array}{ccc} \int \mathcal{D}[F] & \longrightarrow & \int \mathcal{D} \\ \downarrow & \lrcorner & \downarrow{\scriptstyle \pi_{\mathcal{D}}} \\ \mathcal{A} & \xrightarrow{F} & \mathcal{C} \end{array}$$

⌟

Example 2.2.9. For any functor $F : \mathcal{C} \to \mathbf{Set}$, the restriction $\mathbf{El}[F]$ of the displayed category $\mathbf{El}$ over F is a displayed category over $\mathcal{C}$ called the **category of elements** of F.

An object of $\mathbf{El}[F]$ displayed over $c \in \mathcal{C}$ is an element $x \in F(c)$. A unique morphism of $\mathbf{El}[F]$ from $x \in F(c)$ to $y \in F(d)$ over $(f : c \to d) \in \mathcal{C}$ exists if and only if $F(f, x) = y$. ⌟

Definition 2.2.10. Given a functor $F : \mathcal{A} \to \mathcal{C}$ and a displayed category $\mathcal{D}$ over $\mathcal{C}$, a **dependent functor** $F' : \mathcal{A} \to \mathcal{D}[F]$ (over F) is a section of the restriction $\mathcal{D}[F]$ (which is displayed over $\mathcal{A}$).

This corresponds to a functor $F' : \mathcal{A} \to \int \mathcal{D}$ such that $\pi_{\mathcal{D}} \circ F' = F$.

$$\begin{array}{ccc} & {\scriptstyle F'} \nearrow & \int \mathcal{D} \\ & & \downarrow{\scriptstyle \pi_{\mathcal{D}}} \\ \mathcal{A} & \xrightarrow{F} & \mathcal{C} \end{array}$$

⌟

Remark 2.2.11. We now use the notation $\mathcal{D}[-]$ both for the sets of displayed components of $\mathcal{D}$, and for the restriction of $\mathcal{D}$ over a functor. The former can be seen as a special case of the latter, when considering functors and dependent functors out of the walking object and walking arrow categories. Indeed, one can identify an object $x \in \mathcal{C}$ with a functor $\langle x \rangle : 1_{\mathbf{Cat}} \to \mathcal{C}$, and a displayed object in $\mathcal{D}[x]$ with a dependent functor $1_{\mathbf{Cat}} \to \mathcal{D}[\langle x \rangle]$. Similarly, one can identify an arrow $(f : x \to y) \in \mathcal{C}$ with a functor $\langle f \rangle : \{\underline{x} \to \underline{y}\} \to \mathcal{C}$ that sends $\underline{x}$ and $\underline{y}$ to x and y. A displayed arrow $(f' : x' \to y') \in \mathcal{D}[f]$ can then be identified with a dependent functor $\{\underline{x} \to \underline{y}\} \to \mathcal{D}[\langle f \rangle]$ that sends $\underline{x}$ and $\underline{y}$ to x' and y'. ⌟

Remark 2.2.12. We sometimes use the notation $F' : (a : \mathcal{A}) \to \mathcal{D}[F(a)]$ instead of $F' : \mathcal{A} \to \mathcal{D}[F]$ for dependent functors, mainly when $F(a)$ is an expression written without explicitly naming the functor F.

An example is the dependent functor

$$\mathsf{Id} : (X : \mathbf{Set}) \to \mathbf{Func}[X, X]$$

whose action on objects sends a set to its identity function.

In that notation, a cannot be seen as just an object of $\mathcal{A}$. There are two ways to interpret that notation.

- The element a is seen as a generalized element of $\mathcal{A}$ (a functor $\mathcal{X} \to \mathcal{A}$ for an arbitrary category $\mathcal{X}$). Then the actions of F' on objects and morphisms can be recovered for $\mathcal{X} = 1_{\mathbf{Cat}}$ and $\mathcal{X} = \{\underline{x} \to \underline{y}\}$.
- Categories, functors, displayed categories and sections form a *category with families*, a model of dependent type theory. Then $F' : (a : \mathcal{A}) \to \mathcal{D}[F(a)]$ can be seen as a notation for $(a : \mathcal{A} \vdash F' : \mathcal{D}[F(a)]) \in \mathbf{Cat}$, asserting that F' is a term in that category with families. ⌟

2.3 Categories of presheaves and their internal language

Fix a base category $\mathcal{C}$.

Definition 2.3.1. A **presheaf** is a functor $\mathcal{C}^{\mathsf{op}} \to \mathbf{Set}$. A **morphism of presheaves** between $X, Y : \mathcal{C}^{\mathsf{op}} \to \mathbf{Set}$ is a dependent functor $\mathcal{C}^{\mathsf{op}} \to \mathbf{Func}[X, Y]$. The **category of presheaves** is written $\mathbf{Psh}(\mathcal{C})$. ⌟

Definition 2.3.2. A **presheaf family** (or **dependent presheaf**) over $X : \mathcal{C}^{\mathsf{op}} \to \mathbf{Set}$ is a dependent functor $\mathcal{C}^{\mathsf{op}} \to \mathbf{Fam}[X]$. A **section** of a presheaf family $Y : \mathcal{C}^{\mathsf{op}} \to \mathbf{Fam}[X]$ is a dependent functor $\mathcal{C}^{\mathsf{op}} \to \mathbf{Sect}[Y]$. ⌟

Presheaves are used everywhere in the semantics of type theory, starting with the fact that our preferred notion of model of type theory (categories with families) consists of a category together with presheaves of types and terms over a category of contexts and substitutions. The actions on morphisms of these presheaves are the substitution operations.

When working with presheaves, the internal language of the presheaf topos is a very convenient tool. Indeed, any presheaf category can be extended to a model of extensional type theory, and most (constructive) mathematical arguments can be interpreted in this model. Constructions performed in the internal language correspond externally to the constructions of presheaves and natural transformations between them, but all functoriality and naturality conditions don't need to be proven explicitly. More importantly, complex external constructions may correspond to well-known and simple internal constructions.

This means however that we need some way to understand the correspondence between internal constructions and external objects. Using the internal language usually consists of multiple steps. First we axiomatize the data of interest in the internal language. This means that we construct some closed type $(\vdash I\ \mathsf{type})$ in the internal language, and provide a global element of the corresponding presheaf $[\![I]\!]$. Then we perform some construction in the internal language, relying on our axiomatization. We obtain an element $(i : I \vdash o(i) : O(i))$ of some dependent type $(i : I \vdash O(i)\ \mathsf{type})$. Finally, we obtain an external global element of the dependent presheaf $[\![O]\!]$ corresponding to the dependent type O, lying over the provided global element of the presheaf corresponding to I. For this to be possible, we need a way to characterize the presheaves $[\![I]\!]$ and $[\![O]\!]$ corresponding to the types I and O, or at least their sets of global elements. We could also try to keep track of the section $[\![o]\!]$ corresponding to the term o, but it is often more practical to specify what is needed in the type O, e.g. we could replace $O(i)$ by the singleton type $(o' : O(i)) \times (o' = o(i))$, but $(o' = o(i))$ can often be replaced by a simpler relation.

The presheaf topos $\mathbf{Psh}(\mathcal{C})$ is a model of extensional type theory, and we have an explicit description of this model. Whenever we want to understand internal constructions, we could "just" unfold the external definitions of the components this model. This should almost always be avoided, as it is not a robust way to use the internal language. For example, two types that are trivially isomorphic in the presheaf model may correspond externally to presheaves that are isomorphic, but not trivially so (consider $X \times Y \to Z$ and $X \to (Y \to Z)$). Depending on whether one chooses to unfold the definitions of the first or second presheaf, one may obtain very different external unfoldings.

Instead, we distinguish the presheaf topos $\mathbf{Psh}(\mathcal{C})$ from its internal language $\mathcal{L}_{\mathcal{C}}$. While the presheaf topos is a semantic model of extensional type theory, its internal

language is seen as a syntactic model, with an interpretation $[\![-]\!] : \mathcal{L}_{\mathfrak{c}} \to \mathbf{Psh}(\mathcal{C})$ from syntax to semantics. The interpretation should be an equivalence between models of extensional type theory, for some suitable notion of equivalence. We don't make the definition of $\mathcal{L}_{\mathfrak{c}}$ fully precise; it could in fact be identical to $\mathbf{Psh}(\mathcal{C})$, but left abstract. It is better perhaps to see it as a cofibrant replacement of $\mathbf{Psh}(\mathcal{C})$: a syntactic model of extensional type theory obtained by adding all of the structure that exists in $\mathbf{Psh}(\mathcal{C})$ as new axioms. Any term constructed in the internal language only uses finitely many of these axioms.

Note that instead of extensional type theory, we could use intensional type theory with the uniqueness of identity proofs axiom as our internal language. In this situation, understanding the difference between $\mathbf{Psh}(\mathcal{C})$ and $\mathcal{L}_{\mathfrak{c}}$ becomes simpler: $\mathbf{Psh}(\mathcal{C})$ remains the same semantic model of extensional type theory (which happens to also be a model of intensional type theory), while $\mathcal{L}_{\mathfrak{c}}$ ought to be a syntactic model of intensional type theory, enjoying normalization and decidability of equality.

Instead of using just extensional type theory as the internal language of $\mathbf{Psh}(\mathcal{C})$, we will use a multimodal type theory with two modes $\mathfrak{c}$ and $\mathfrak{1}$, corresponding to $\mathbf{Psh}(\mathcal{C})$ and **Set**, and a single modality $\Box$ corresponding to the global elements functor $\mathbf{Psh}(\mathcal{C}) \to \mathbf{Set}$ (which is right adjoint to the discrete presheaf functor $\mathbf{Set} \to \mathbf{Psh}(\mathcal{C})$.) This internal language consists of two models $\mathcal{L}_{\mathfrak{1}}$ and $\mathcal{L}_{\mathfrak{c}}$ of extensional type theory, with interpretations $[\![-]\!] : \mathcal{L}_{\mathfrak{1}} \to \mathbf{Set}$ and $[\![-]\!] : \mathcal{L}_{\mathfrak{c}} \to \mathbf{Psh}(\mathcal{C})$, and additional structure corresponding to the modality. This serves two purposes:

- As mentioned before, we don't want to compute explicitly the interpretation $[\![A]\!]$ of a type $(\vdash A\ \mathsf{type}) \in \mathcal{L}_{\mathfrak{c}}$ in the presheaf model $\mathbf{Psh}(\mathcal{C})$, because the construction of the presheaf model is rather complicated.

  Computing the interpretation $[\![A]\!]$ in **Set** of a type $(\vdash A\ \mathsf{type}) \in \mathcal{L}_{\mathfrak{1}}$ is however fine, at least when the type A does not involve the modality $\Box$, because of how simple the description of **Set** as a model of extensional type theory is.

  Including the second mode $\mathfrak{1}$ provides a better interface between the internal language and external constructions.

- While staying in mode $\mathfrak{c}$ is sufficient for most applications of the internal language, verifying that particular axioms are satisfied often rely on properties that cannot be expressed without modalities, such as the fact that colimits are computed objectwise in presheaf categories, or the fact that some representable presheaf is a tiny object.

  Having access to the modalities thus makes the axiomatization easier.

Kripke-Joyal forcing (Awodey, Gambino, and Hazratpour 2024) is another method that could be used to mediate between the internal language and external constructions. Yet another alternative to our approach is the framework recently proposed by Kovács and Sattler (2025), which should provide a way to avoid substructural modalities.

2.3.1 Multimodal type theory

We present the multimodal type theory to be used as the internal language as a dual-context modal type theory. We use notations close to the crisp type theory of Shulman (2017). We could alternatively work with the instance of MTT (Gratzer, Kavvos, Nuyts, and Birkedal 2021) whose 2-category of modes has a single non-identity arrow $\{\mathfrak{1} \xrightarrow{\Box} \mathfrak{c}\}$.

Using a dual-context theory instead has the advantage of being more self-contained for the purposes of this thesis.

We have two notions of contexts: single contexts Γ, and dual contexts $\Gamma \mid \Delta$, where Δ can depend on Γ. There are two type judgements, over the two kinds of contexts.

$$\frac{}{\Gamma \vdash A\ \mathsf{type}_{\mathfrak{1}}} \qquad\qquad \frac{}{\Gamma \mid \Theta \vdash A\ \mathsf{type}_{\mathfrak{c}}}$$

We can extend a single context by a $\mathfrak{1}$-type: $(\gamma :: \Gamma).(a :: A(\gamma))$, or a dual context by a $\mathfrak{c}$-type: $(\gamma :: \Gamma) \mid (\delta : \Delta).(a : A(\gamma, \delta))$. Following the notations of crisp type theory and spatial type theory, we use $(a :: A)$ for variables in the first half of the dual context, and $(a : A)$ for variables in the second half of the dual context. Writing $(a :: A)$ presupposes that A is a $\mathfrak{1}$-type, and $(a : A)$ presupposes that A is a $\mathfrak{c}$-type.

Both the $\mathfrak{1}$- and the $\mathfrak{c}$- types are closed under all of the operations of extensional type theory: universes, dependent products, dependent sums, extensional equality types, quotients, etc. We write $\mathsf{Set}^{\mathfrak{1}}_n$ for the universes at mode $\mathfrak{1}$, and $\mathsf{Set}^{\mathfrak{c}}_n$ for the universes at mode $\mathfrak{c}$. When working in the internal language, elements of $\mathsf{Set}^{\mathfrak{1}}$ are called $\mathfrak{1}$-sets, and elements of $\mathsf{Set}^{\mathfrak{c}}$ are called $\mathfrak{c}$-sets. Any general construction in extensional type theory can be interpreted either at mode $\mathfrak{1}$ or at mode $\mathfrak{c}$.

The modality $\Box$ sends a $\mathfrak{c}$-type to a $\mathfrak{1}$-type. The $\mathfrak{c}$-type has to lie over a dual context with an empty second half.

$$\frac{\Gamma \mid \cdot \vdash A\ \mathsf{type}_{\mathfrak{c}}}{\Gamma \vdash \Box A\ \mathsf{type}_{\mathfrak{1}}} \qquad \frac{\Gamma \mid \cdot \vdash a : A}{\Gamma \vdash \mathsf{mod}_{\Box}(a) :: \Box A} \qquad \frac{\Gamma \vdash a :: \Box A}{\Gamma \mid \cdot \vdash \mathsf{unmod}_{\Box}(a) : A}$$

$$\frac{\Gamma \vdash a :: \Box A}{\Gamma \vdash \mathsf{mod}_{\Box}(\mathsf{unmod}_{\Box}(a)) = a} \qquad \frac{\Gamma \mid \cdot \vdash a : A}{\mathsf{unmod}_{\Box}(\mathsf{mod}_{\Box}(a)) = a}$$

Note that the rule for $\mathsf{unmod}_{\Box}$ can be generalized thanks to weakening:

$$\frac{\Gamma \vdash a :: \Box A}{\Gamma \mid \Delta \vdash \mathsf{unmod}_{\Box}(a) : A}$$

However, writing $\mathsf{mod}_{\Box}(a)$ given $\Gamma \mid \Delta \vdash a : A$ is only valid when a does not actually depend on Δ.

We can check that there is, at mode $\mathfrak{1}$, a term

$$\begin{aligned} &\mathsf{app}_{\Box} :: \Box(A \to B) \to (\Box A \to \Box B), \\ &\mathsf{app}_{\Box} \triangleq \lambda f\ a \mapsto \mathsf{mod}_{\Box}(\mathsf{unmod}_{\Box}(f)(\mathsf{unmod}_{\Box}(a))), \end{aligned}$$

witnessing the fact that $\Box$ is an applicative functor (McBride and Paterson 2008). There is however no term of type

$$(\Box A \to \Box B) \to \Box(A \to B).$$

Indeed, to construct a function $A \to B$, we need to bind a variable $(a : A)$ in the second half of the dual context, and the term $(\mathsf{mod}_{\Box}(a) :: \Box A)$ is not valid, because the variable a is not in the first half of the dual context. Semantically, $(a : A)$ is a general element of a presheaf A, and cannot be seen as a global element.

From now on, we will omit $\mathsf{unmod}_{\Box}(-)$ and $\mathsf{mod}_{\Box}(-)$, they should be inserted exactly when moving between modes $\mathfrak{1}$ and $\mathfrak{c}$.

Semantically, this language is interpreted as follows:

- A single context Γ is interpreted as a set $[\![\Gamma]\!]$.
- A 1-type $(\Gamma \vdash A\ \mathsf{type}_1)$ is interpreted as a family of sets $[\![A]\!] : [\![\Gamma]\!] \to \mathsf{Set}$.
- The operations of extensional type theory at mode 1 are interpreted using the structure of **Set** as a model of extensional type theory.
- For a dual context $\Gamma \mid \Delta$, the component Δ is interpreted as a family of presheaves $[\![\Delta]\!] : [\![\Gamma]\!] \to \mathsf{Psh}(\mathcal{C})$.
- A $\mathfrak{c}$-type $(\Gamma \mid \Delta \vdash A\ \mathsf{type}_\mathfrak{c})$ is interpreted as a family of dependent presheaves $[\![A]\!] : (\gamma : [\![\Gamma]\!]) \to \mathsf{PshFam}([\![\Delta]\!](\gamma))$.
- The operations of extensional type theory at mode $\mathfrak{c}$ are interpreted using the structure of $\mathbf{Psh}(\mathcal{C})$ as a model of extensional type theory, pointwise over $[\![\Gamma]\!]$.
- The modality $\Box A$ sends the family of presheaves $[\![A]\!] : [\![\Gamma]\!] \to \mathsf{Psh}(\mathcal{C})$ to the family of its global elements $[\![\Box A]\!] = \lambda\gamma \to (\mathcal{C}^{\mathsf{op}} \to \mathbf{El}[\![A]\!])$.

2.3.2 Axiomatization in multimodal type theory

We now work in the internal language. We assume the following axiom, which internalizes a large part of the intended semantics.

Axiom 2.3.3. There is a small category $\mathcal{C} :: \mathbf{Cat}$, and for any universe level n, an equivalence $[\![-]\!] :: \Box\mathbf{Set}^\mathfrak{c}_n \cong \mathbf{Psh}_n(\mathcal{C})$ of categories, compatible with the universe lifting operation for $n \leq m$. ⌟

This axiom is validated in the intended model, with the internal category $\mathcal{C}$ interpreted as the external base category $\mathcal{C}$. Indeed, in the intended model, $\Box\mathbf{Set}^\mathfrak{c}_n$ is a category whose objects are global sections of the Hofmann-Streicher universe classifying the n-small dependent presheaves. These global sections can be identified with presheaves. The morphisms of $\Box\mathbf{Set}^\mathfrak{c}_n$ are then natural transformations between the corresponding presheaves.

Note that the statement of Axiom 2.3.3 is all in the internal language, at mode 1. Here $\mathbf{Set}^\mathfrak{c}_n$ is the category of n-small sets at mode $\mathfrak{c}$: its objects are elements of the universe $\mathsf{Set}^\mathfrak{c}_n$, its morphisms are functions. The modality $\Box$ preserves finite limits; as a consequence it has an action on algebras of any essentially algebraic or generalized algebraic theory, in particular categories. Then $\Box\mathbf{Set}^\mathfrak{c}_n$ is a category at mode 1. Its objects are elements of $\Box\mathsf{Set}^\mathfrak{c}_n$, and morphisms from X to Y are elements of $\Box(X \to Y)$. The presheaf category $\mathbf{Psh}_n(\mathcal{C})$ is the category of presheaves valued in 1-sets.

The equivalence $[\![-]\!]$ then consists of:

- A correspondence $\Box\mathsf{Set}^\mathfrak{c}_n \cong \mathsf{Psh}_n(\mathcal{C})$ between elements of $\Box\mathsf{Set}^\mathfrak{c}_n$ and presheaves.
- A bijective correspondence $\Box(X \to Y) \cong ([\![X]\!] \Rightarrow [\![Y]\!])$ between elements of $\Box(X \to Y)$ and natural transformations.
- Such that identities and compositions are preserved by these correspondences.

This equivalence can be extended from categories to categories with families. Indeed, both sides of the equivalence have the structure of a locally cartesian closed category, and locally cartesian closed categories that are equivalent as categories are also equivalent as locally cartesian closed categories. Thus we also have:

- A correspondence $\Box(X \to \mathsf{Set}^{\mathfrak{c}}_n) \cong \mathsf{PshFam}_n(\llbracket X \rrbracket)$ between elements of $\Box\mathsf{Set}^{\mathfrak{c}}_n$ and presheaf families.
- A bijective correspondence $\Box((x : X) \to Y(x)) \cong \mathsf{PshSect}(\llbracket Y \rrbracket)$ between elements of $\Box((x : X) \to Y(x))$ and sections of presheaf families.

Using these correspondences, structures can be transferred between $\Box\mathbf{Set}^{\mathfrak{c}}_n$ and $\mathbf{Psh}_n(\mathcal{C})$. Any object or type characterized by a universal property is uniquely determined up to isomorphism. For example, the Σ-types, Π-types, limits and colimits in $\Box\mathbf{Set}^{\mathfrak{c}}_n$ and $\mathbf{Psh}_n(\mathcal{C})$ have to coincide up to isomorphism.

We can use this correspondence to state the Yoneda lemma in a way that involves the mode $\mathfrak{c}$.

Lemma 2.3.4. *For every $\Gamma :: \mathcal{C}$, there is a $\mathfrak{c}$-set $\text{よ}(\Gamma) :: \Box\mathsf{Set}^{\mathfrak{c}}_0$, such that for any $X :: \Box\mathsf{Set}^{\mathfrak{c}}_n$, there is an isomorphism $\Box(\text{よ}(\Gamma) \to X) \cong \llbracket X \rrbracket(\Gamma)$, naturally in X.*

Proof. This follows from the Yoneda lemma; $\text{よ}(\Gamma)$ is obtained as the element of $\Box\mathsf{Set}^{\mathfrak{c}}_0$ corresponding to a representable presheaf in $\mathbf{Psh}_0(\mathcal{C})$. □

We can now use $\Box(\text{よ}(\Gamma) \to X)$ to talk about elements of $X :: \Box\mathsf{Set}^{\mathfrak{c}}_n$, without involving $\llbracket X \rrbracket$. We can also redefine $\llbracket X \rrbracket$ in a way that makes use of the Yoneda embedding:

$$\llbracket X \rrbracket'(\Gamma) = \Box(\text{よ}(\Gamma) \to X),$$
$$\llbracket X \rrbracket'(f : \Delta \to \Gamma) \triangleq \lambda x\ \delta \mapsto x(f(\delta)).$$

Because $\llbracket - \rrbracket$ was abstract, we can replace it by $\llbracket - \rrbracket'$ without loss of generality.

We say that a $\mathfrak{c}$-set $X :: \Box\mathsf{Set}^{\mathfrak{c}}_n$ is representable if there is $\Gamma :: \mathcal{C}$ and an isomorphism $\Box(\text{よ}(\Gamma) \cong X)$. Note that this is not a local notion: there is no family $\mathsf{is\text{-}representable} : \Box(\mathsf{Set}^{\mathfrak{c}} \to \mathsf{Set}^{\mathfrak{c}})$ classifying the representable $\mathfrak{c}$-sets.

Proposition 2.3.5 (Propositions can be tested objectwise)**.** *Let $Y :: \Box((x : X) \to \mathsf{Prop}^{\mathfrak{c}})$ be a family of propositions at mode $\mathfrak{c}$. The proposition*

$$\Box((x : X) \to Y(x))$$

is true if and only if for every $\Gamma :: \mathcal{C}$ and element $x :: \Box(\text{よ}(\Gamma) \to X)$, the proposition

$$\Box((\gamma : \text{よ}(\Gamma)) \to Y(x(\gamma)))$$

is true.

Proof. The corresponding fact is true when looking at sections of propositional presheaf families. □

Corollary 2.3.6. *Let $f :: \Box((x : X) \to Y(x) \to Z(x))$ be a family of functions. If for every element $x :: \Box(\text{よ}(\Gamma) \to X)$, the induced function*

$$f_\Gamma :: \Box((\gamma : \text{よ}(\Gamma)) \to Y(x(\gamma))) \to \Box((\gamma : \text{よ}(\Gamma)) \to Z(x(\gamma)))$$

is bijective, then for every $x : X$, $f(x) : Y(x) \to Z(x)$ is bijective.

Proof. This is an instance of proposition 2.3.5 for the family

$$(x : X)(z : Z(x)) \mapsto \mathsf{is\text{-}contr}((y : Y(x)) \times f(x, y) = z).$$ □

Proposition 2.3.7 (Colimits are computed objectwise). *Let* $\mathcal{D} :: \square(\text{よ}(\Gamma) \to \mathbf{Cat})$ *be a family of categories at mode* $\mathfrak{c}$ *and* $A :: \square(\forall\gamma \to \mathbf{Cat}(\mathcal{D}(\gamma), \mathbf{Set}^{\mathfrak{c}}))$ *be a family of functors.*

Then the canonical map

$$\operatorname*{colim}_{d \in \square(\Pi_\gamma \mathcal{D}(\gamma))} (\square(\forall\gamma \to A(\gamma, d(\gamma)))) \to \square(\forall\gamma \to \operatorname*{colim}_{d \in \mathcal{D}(\gamma)} (A(\gamma, d)))$$
$$\iota_d(a) \mapsto (\lambda\gamma \mapsto \iota_{d(\gamma)}(a(\gamma)))$$

is bijective (where $\Pi_\gamma \mathcal{D}(\gamma)$ *is a product in* **Cat***, and colimit inclusion maps are written* $\iota\_$*).* □

2.3.3 Universes in presheaf categories

Hofmann-Streicher universes (Hofmann and Streicher 1997) strictly classify the (small) dependent presheaves, meaning that the n-th universe $\mathcal{U}_n$ is a representing object for the functor

$$\mathsf{PshFam}_n : \mathbf{Psh}(\mathcal{C})^{\mathsf{op}} \to \mathbf{Set}_{n+1}$$

which sends $X \in \mathbf{Psh}(\mathcal{C})$ to the set of n-small dependent presheaves over X, i.e. the dependent functors $\mathcal{C}^{\mathsf{op}} \to \mathbf{Fam}_n[X]$. The universal property of $\mathcal{U}_n$ then says that natural transformations $X \Rightarrow \mathcal{U}_n$ are in bijective correspondence with n-small dependent presheaves over X, naturally in X.

Given a functor $P : \mathbf{Psh}(\mathcal{C})^{\mathsf{op}} \to \mathbf{Set}$, we may want to know whether it is representable and how to construct a representing object $\mathcal{U}_P$. A well-known trick is that if this functor is representable, then the representing object is the composite $(P \circ \text{よ}) : \mathcal{C}^{\mathsf{op}} \to \mathbf{Set}$. Indeed, if $\mathcal{U}_P$ is a representing object, we have natural isomorphisms $P(\text{よ}(\Gamma)) \cong (\text{よ}(\Gamma) \Rightarrow \mathcal{U}_P) \cong \mathcal{U}_P(\Gamma)$. This provides a candidate representing object, and it can be shown that it is an actual representing object when P is continuous, due to the fact that any presheaf is a colimit of representable presheaves.

Checking continuity is however not straightforward, so we give weaker criterion (lemma 2.3.9) that can be checked syntactically by unfolding the definition of P. This weaker criterion is sufficient to construct the Hofmann-Streicher universes (corollary 2.3.10), and some other classifiers that are used in this thesis, such as the classifier of locally representable dependent presheaves (Construction 2.3.14), and the classifier of levelwise decidable propositions (proposition 2.3.22).

Lemma 2.3.8. *Let* $S \in \mathbf{Cat}$ *be a small category with a terminal object* 1_S *and* T *be an* n*-small family over the set* $\mathbf{Cat}(S, \mathcal{C})$ *of* S*-shaped diagrams in* $\mathcal{C}$*. Then the functor*

$$P : \mathbf{Psh}(\mathcal{C})^{\mathsf{op}} \to \mathbf{Set}_n,$$
$$P(X) \triangleq (F : \mathbf{Cat}(S, \mathcal{C}))(x : X(F(1_S))) \to T(F),$$
$$P(\alpha : X \Rightarrow Y) \triangleq (\lambda M_Y\ F\ x \mapsto M_Y(F, \alpha_{F(1_S)}(x)))$$

is representable, and its representing object is n*-small.*

Proof. Our candidate representing object is

$$\mathcal{U}_P(d) = (F : \mathbf{Cat}(S, \mathcal{C}))(f : F(1_S) \to d) \to T(F),$$
$$\mathcal{U}_P(g : d \to c) = \lambda u\ F\ f \mapsto u(F, g \circ f).$$

An element of $P(X)$ is a function

$$(F : \mathbf{Cat}(S, \mathcal{C}))(x : X(F(1_S))) \to T(F).$$

A natural transformation $X \Rightarrow \mathcal{U}_P$ is the data of functions

$$(x : X(d))(F : \mathbf{Cat}(S, \mathcal{C}))(f : F(1_S) \to d) \to T(F),$$

naturally in d.

By naturality, these functions are uniquely determined by their evaluation at $d = F(1_S)$ and $f = \mathsf{id}$. Furthermore, evaluation at $d = F(1_S)$ and $f = \mathsf{id}$ is natural in X. This provides a natural isomorphism $P(X) \cong (X \Rightarrow \mathcal{U}_P)$, witnessing the fact that $\mathcal{U}_P$ represents the functor P. □

In concrete applications, lemma 2.3.8 is often instantiated for S a linear order $[n] = \{n \to (n-1) \to \ldots \to 1 \to 0\}$ (where 0 is the terminal object). Then a S-shaped diagram is a sequence of n-composable arrows $\{c_n \xrightarrow{f_n} c_{n-1} \to \ldots \to c_1 \xrightarrow{f_1} c_0\}$.

Lemma 2.3.9. *Let $\mathcal{D}$ be a small category, $S : \mathcal{D} \to \mathbf{Cat}$ be a functor such that $S(d)$ has a terminal object $1_{S(d)}$ for any $d \in \mathcal{D}$ and $T : \mathcal{D} \to \mathbf{Fam}_n[\mathbf{Cat}(S(-), \mathcal{C})]$ be a dependent functor over $\mathbf{Cat}(S(-), \mathcal{C}) : \mathcal{D} \to \mathbf{Set}$. Then the functor*

$$P : \mathbf{Psh}(\mathcal{C})^{\mathsf{op}} \to \mathbf{Set}_n,$$
$$P(X) \triangleq \lim_{d \in \mathcal{D}} ((F : \mathbf{Cat}(S(d), \mathcal{C}))(x : X(F(1_{S(d)}))) \to T(d, F)).$$

is representable and its representing object is n-small.

Proof. This follows from lemma 2.3.8 and the fact that a limit of representable functors is representable by the limit of the representing objects. □

In applications, lemma 2.3.9 is instantiated for limits corresponding to iterated dependent sums.

Corollary 2.3.10 (Hofmann-Streicher universes)**.** *For every small category $\mathcal{C}$ and universe level n, the functor*

$$\mathsf{PshFam}_n : \mathbf{Psh}(\mathcal{C})^{\mathsf{op}} \to \mathbf{Set}_{n+1}.$$

which sends a presheaf to the set of n-small dependent presheaves over it, is representable.

Proof. Observe that we can unfold the definition of $\mathsf{PshFam}_n(X)$ as follows:

$$\begin{aligned}
&\mathsf{PshFam}_n(X) \cong \\
&\quad (Y_0 : \forall c_0\ (x : X(c_0)) \to \mathsf{Set}_n) \\
&\times (Y_1 : \forall (c_1 \xrightarrow{f_1} c_0)(x : X(c_0)) \to Y_0(c_0, x) \to Y_0(c_1, x[f_1])) \\
&\times (- : \forall c_0\ (x : X(c_0)) \to Y_1(\mathsf{id}, x) = \mathsf{id}) \\
&\times (- : \forall (c_2 \xrightarrow{f_2} c_1 \xrightarrow{f_1} c_0)(x : X(c_0)) \to Y_1(f_2, x[f_0]) \circ Y_1(f_0, x) = Y_1(f_1 \circ f_2, x))
\end{aligned}$$

The result then follows from lemma 2.3.9, for a limit that encodes the dependencies of the above definition. The objects of the (inverse) diagram $\mathcal{D} \to \mathbf{Cat}$ are the four categories $[0], [1], [0], [2]$. There are six generating arrows in $\mathcal{D}$, corresponding to the dependencies $Y_0(c_0, x)$, $Y_0(c_1, x[f_1])$, $Y_1(\mathsf{id}, x)$, $Y_1(f_2, x[f_0])$, $Y_1(f_0, x)$ and $Y_1(f_1 \circ f_2, x)$. □

2.3.4 Local representability

We discuss the notion of locally representable dependent presheaf, which we will use to encode the notion of context extension in models of type theory. A dependent presheaf Y over X is locally representable exactly when the "total" natural transformation $\pi_1 : (\Sigma_X Y) \twoheadrightarrow X$ is (algebraically) *representable*. The notion of representable natural transformation was used by Awodey (2018) to define *natural models*, a notion of model of type theory. We prefer the terminology "locally representable" because it aligns with other uses of the adverb "locally" in category theory: it involves looking at all slices $(\mathcal{C}/\Gamma)$ of a category.

Definition 2.3.11. A **local representability structure** over a dependent presheaf $Y : \mathcal{C}^{\mathsf{op}} \to \mathbf{Fam}[X]$ consists, for every element $x : X(\Gamma)$, of an object $\Gamma.Y[x] \in \mathcal{C}$, a projection morphism $\boldsymbol{p}_x : \Gamma.Y[x] \to \Gamma$ and a generic element $\boldsymbol{q}_x : Y(\Gamma.x[\boldsymbol{p}_x])$, such that for every morphism $f : \Delta \to \Gamma$ and $y : Y(\Delta, x[f])$, there is a unique morphism $\langle f, y\rangle : \Delta \to \Gamma.Y[x]$ such that $\boldsymbol{p}_x \circ \langle f, y\rangle = f$ and $\boldsymbol{q}_x[\langle f, y\rangle] = y$. ⌟

We can give a second definition of local representability structure that makes use of the internal language. That definition is still external, because of the quantifications over Y and x are external. We will get later an internal notion of local representability by defining a universe classifying the locally representable dependent presheaves.

Definition 2.3.12 (In the internal language)**.** A **local representability structure** over $Y :: \Box(X \to \mathsf{Set}^{\mathfrak{c}})$ consists, for every element $x :: \Box(ょ(\Gamma) \to X)$, of an object $\Gamma.Y[x] \in \mathbb{C}$ representing the $\mathfrak{c}$-set

$$(\gamma : ょ(\Gamma)) \times Y(x(\gamma)).$$ ⌟

Proposition 2.3.13. *Given $Y :: \Box(X \to \mathsf{Set}^{\mathfrak{c}})$, there is a bijective correspondence between local representability structures on Y in the sense of definition 2.3.12, and local representability structures on $[\![Y]\!] :: \mathsf{PshFam}([\![X]\!])$ in the sense of definition 2.3.11.*

Proof. We already have a correspondence between elements $\Box(ょ(\Gamma) \to A)$ and $[\![A]\!](\Gamma)$. Take any $a :: \Box(ょ(\Gamma) \to A)$.

We unfold definition 2.3.12 into the following components:

$$\begin{aligned}
&\Gamma.B[a] \in \mathcal{C},\\
&\boldsymbol{p} :: \Box(ょ(\Gamma.B[a]) \to ょ(\Gamma)),\\
&\boldsymbol{q} :: \Box((f : ょ(\Gamma.B[a])) \to B(a(\boldsymbol{p}(f)))),\\
&\langle -, - \rangle :: \Box((\gamma : ょ(\Gamma)) \times (b : B(a(\gamma)))\\
&\quad \to \mathsf{is\text{-}contr}((f : ょ(\Gamma.B[a])) \times (\boldsymbol{p}(f) = \gamma) \times (\boldsymbol{q}(f) = b))).
\end{aligned}$$

By the Yoneda lemma, $\boldsymbol{p}$ and $\boldsymbol{q}$ correspond to $p :: \mathcal{C}(\Gamma.B[a], \Gamma)$ and $q :: [\![B]\!](a[p])$.

Finally, since $\mathsf{is\text{-}contr}(-)$ is propositional, the last component can be decomposed objectwise. It then says that for every $\Delta \in \mathcal{C}$, $\gamma :: \Box(ょ(\Delta) \to ょ(\Gamma))$ and $b :: \Box(\forall\delta \to B(a(\gamma(\delta))))$, there is a unique element of

$$\Box(\forall\delta \to (f : ょ(\Gamma.B[a])) \times (\boldsymbol{p}(f) = \gamma) \times (\boldsymbol{q}(f) = b)).$$

Up to applications of the Yoneda lemma, this corresponds to the universal property of $\Gamma.B[a]$ from definition 2.3.11, as needed. □

Construction 2.3.14. There is a presheaf $\mathcal{U}_{\mathsf{rep}}$ that strictly classifies local representability dependent presheaves, meaning that there is a bijective correspondence between dependent presheaves $Y : \mathcal{C}^{\mathsf{op}} \to \mathbf{Fam}[X]$ equipped with a local representability structure and natural transformations from X to $\mathcal{U}_{\mathsf{rep}}$. ⌟

Proof. Write $\mathsf{PshFam}_{\mathsf{rep}}(X)$ for the set of locally representable dependent presheaves over a base presheaf X. We can unfold the definition of $\mathsf{PshFam}_{\mathsf{rep}}(X)$ as follows:

$$
\begin{aligned}
&\mathsf{PshFam}_{\mathsf{rep}}(X) \cong \\
&\quad (Y_0 : \forall c_0\ (x : X(c_0)) \to \mathsf{Set}) \\
&\times (Y_1 : \forall (c_1 \xrightarrow{f_1} c_0)(x : X(c)) \to Y_0(c_0, x) \to Y_0(c_1, x[f_1])) \\
&\times (- : \forall c_0\ (x : X(c_0)) \to Y_1(\mathsf{id}, x) = \mathsf{id}) \\
&\times (- : \forall (c_2 \xrightarrow{f_2} c_1 \xrightarrow{f_1} c_0)(x : X(c_0)) \to Y_1(f_2, x[f_0]) \circ Y_1(f_0, x) = Y_1(f_1 \circ f_2, x)) \\
&\times (-.Y[-] : \forall c_0\ (x : X(c_0)) \to \mathsf{Ob}_{\mathcal{C}}) \\
&\times (\boldsymbol{p} : \forall c_0\ (x : X(c_0)) \to \mathcal{C}(c_0.Y[x], c_0)) \\
&\times (\boldsymbol{q} : \forall (c_1 \xrightarrow{f_1} c_0)(x : X(c_0))(f_1 = \boldsymbol{p}(c_0, x)) \to Y_0(c_0.Y[x], x[f_1])), \\
&\times (- : \forall (c_1 \xrightarrow{f_1} c_0 \xleftarrow{f_2} c_2)(x : X(c_0))(f_2 = \boldsymbol{p}(c_0, x))(y : Y_0(c_1, x[f_1])) \to \\
&\qquad \mathsf{is\text{-}contr}((g : \mathcal{C}(c_1, c_0.Y[x])) \times (\boldsymbol{p}(c_0, x) \circ g = f_1) \times (Y_1(g, x, \boldsymbol{q}(f_2, x)) = y))).
\end{aligned}
$$

The existence of the universe $\mathcal{U}_{\mathsf{rep}}$ then follows from lemma 2.3.9. We have to use a weird specification for $\boldsymbol{q}$ and extended substitutions, due to the fact that lemma 2.3.9 only allows the use of $x[f]$ when f is an arrow of the diagram. □

The above constructions implies that the notion of locally representable $\mathfrak{c}$-set makes sense in the internal language of $\mathbf{Psh}(\mathcal{C})$:

Corollary 2.3.15 (In the internal language)**.** *There is a family* $\mathsf{HasLocalRep} : \mathsf{Set}^{\mathfrak{c}} \to \mathsf{Set}^{\mathfrak{c}}$ *that strictly classifies local representability structures: for every* $Y :: \square(X \to \mathsf{Set}^{\mathfrak{c}})$*, there is a bijective correspondence between local representability structures on* Y *and maps* $Y_{\mathsf{rep}} :: \square((x : X) \to \mathsf{HasLocalRep}(Y(x)))$. □

We write $\mathsf{Set}_{\mathsf{rep}}$ for the total space $\mathsf{Set}_{\mathsf{rep}} = (X : \mathsf{Set}^{\mathfrak{c}}) \times \mathsf{HasLocalRep}(X)$.

Proposition 2.3.16. *The locally representable* $\mathfrak{c}$*-sets are closed under dependent sums.*

Proof. We want to construct an element of

$$\square((X : \mathsf{Set}_{\mathsf{rep}})(Y : X \to \mathsf{Set}_{\mathsf{rep}}) \to \mathsf{HasLocalRep}((x : X) \times Y(x))).$$

Since $\mathsf{HasLocalRep}$ classifies local representability structures, it suffices to show that the family

$$(X : \mathsf{Set}_{\mathsf{rep}})(Y : X \to \mathsf{Set}_{\mathsf{rep}}) \mapsto (x : X) \times Y(x)$$

has a local representability structure.

Take an object Γ and elements $x :: \square(ʎ(\Gamma) \to \mathsf{Set}_{\mathsf{rep}})$ and $y :: \square((\gamma : ʎ(\Gamma)) \to X(\gamma) \to \mathsf{Set}_{\mathsf{rep}})$. We have to show that $(\gamma : ʎ(\Gamma)) \times (x : X(\gamma)) \times Y(\gamma, x)$ is representable.

Since $\mathsf{Set}_{\mathsf{rep}}$ classifies local representability structures, we have local representability structures over X and Y. The local representability structure over X provides an object $\Gamma.X$ along with an isomorphism

$$よ(\Gamma.X) \cong (\gamma : よ(\Gamma)) \times (x : X(\gamma)).$$

The local representability structure on Y then provides an object $\Gamma.X.Y$ and an isomorphism

$$よ(\Gamma.X.Y) \cong (\gamma : よ(\Gamma)) \times (x : X(\gamma)) \times Y(\gamma, x),$$

showing that $(\gamma : よ(\Gamma)) \times (x : X(\gamma)) \times Y(\gamma, x)$ is represented by $\Gamma.X.Y$, as needed. □

Proposition 2.3.17. *If $\mathcal{C}$ has binary products, then any representable presheaf is locally representable (as a dependent presheaf over the terminal presheaf).*

Proof. We use the internal language.

If $\mathcal{C}$ has binary products, then for every $\Gamma, \Delta \in \mathcal{C}$, we have an isomorphism

$$よ(\Gamma \times \Delta) \cong よ(\Gamma) \times よ(\Delta).$$

Using the characterization of definition 2.3.12, this means that $よ(\Delta)$ is locally representable. □

Definition 2.3.18. Let A be a set. We say that a set X is A-null if the function

$$\begin{aligned} &X \to (A \to X), \\ &x \mapsto (\lambda a \mapsto x) \end{aligned}$$

is an isomorphism.

We say that a category is A-null if its sets of objects and morphisms are A-null. ⌟

Proposition 2.3.19. *Let A be a locally representable $\mathfrak{c}$-set. For any $X :: \mathsf{Set}^1$, the discrete $\mathfrak{c}$-set ΔX is A-null.*

Proof. We have to construct an element of

$$(X : \mathsf{Set}^1) \to \Box((A : \mathcal{U}_{\mathsf{rep}}) \to \mathsf{is\text{-}iso}(\Delta X \to (A \to \Delta X))).$$

By corollary 2.3.6, it suffices to check for any object $\Gamma \in \mathcal{C}$ and $A : \Box(よ(\Gamma) \to \mathcal{U}_{\mathsf{rep}})$ that the map

$$\Box(よ(\Gamma) \to \Delta X) \to \Box((\gamma : よ(\Gamma)) \to A(\gamma) \to \Delta X)$$

which sends x to $\lambda\gamma\, a \mapsto x(\gamma)$ is bijective.

But by definition of ΔX, we have $\Box(よ(\Gamma) \to \Delta X) \cong X$.

Because $(\gamma : よ(\Gamma)) \to A(\gamma) \cong よ(\Gamma.A)$, we also have $\Box((\gamma : よ(\Gamma)) \to A(\gamma) \to \Delta X \cong X$.

Thus we have a diagram

$$\begin{array}{ccc} & \overset{\cong}{\nearrow} & \Box(よ(\Gamma) \to \Delta X) \\ & & \downarrow \\ X & \xrightarrow[\cong]{} & \Box((\gamma : よ(\Gamma)) \to A(\gamma) \to \Delta X). \end{array}$$

This diagram commutes, so the right vertical map is an isomorphism, as needed. □

Proposition 2.3.20 (Locally representable sets are tiny)**.** *Let A be a locally representable $\mathfrak{c}$-set. Then exponentiation $(A \to -)$ preserves colimits indexed by A-null diagram categories.*

Given any any family of functors $F : A \to \mathbf{Cat}(\mathcal{D}, \mathsf{Set}^{\mathfrak{c}})$, the canonical function

$$\operatorname*{colim}_{d\in\mathcal{D}}((a : A) \to F(a, d)) \to ((a : A) \to \operatorname*{colim}_{d\in\mathcal{D}}(F(a, d)))$$

is bijective.

Proof. We have to inhabit the following type in the internal language:

$$\begin{aligned}&\Box((A : \mathsf{Set}_{\mathsf{rep}})(\mathcal{D} : \mathbf{Cat})(- : \mathsf{is\text{-}null}_A(D))(F : A \to \mathbf{Cat}(\mathcal{D}, \mathsf{Set})),\\ &\quad \to \mathsf{is\text{-}iso}(\operatorname*{colim}_{d\in\mathcal{D}}((a : A) \to F(a, d)) \to ((a : A) \to \operatorname*{colim}_{d\in\mathcal{D}}(F(a, d))))).\end{aligned}$$

By corollary 2.3.6, it suffices to construct the isomorphism objectwise. Take

$$\begin{aligned}&\Gamma \in \mathcal{C},\\ &A :: \text{よ}(\Gamma) \to \mathsf{Set}_{\mathsf{rep}},\\ &\mathcal{D} :: \text{よ}(\Gamma) \to \mathbf{Cat},\\ &- :: \forall\gamma \to \mathsf{is\text{-}null}_{A(\gamma)}(\mathcal{D}(\gamma)),\\ &F :: \forall\gamma \to A \to \mathbf{Cat}(\mathcal{D}(\gamma), \mathsf{Set}).\end{aligned}$$

We need to prove that the canonical map

$$\begin{aligned}&\Box(\forall\gamma \to \operatorname*{colim}_{d\in\mathcal{D}(\gamma)}((a : A(\gamma)) \to F(\gamma, a, d)))\\ &\quad \to \Box(\forall\gamma \to (a : A(\gamma)) \to \operatorname*{colim}_{d\in\mathcal{D}(\gamma)}(F(\gamma, a, d)))\end{aligned}$$

is an isomorphism.

First note that we have an object $\Gamma.A$ and an isomorphism

$$\text{よ}(\Gamma.A) \cong (\gamma : \text{よ}(\Gamma)) \cong A(\gamma).$$

This induces an isomorphism

$$\begin{aligned}&\Box(\forall\gamma \to (a : A(\gamma)) \to \operatorname*{colim}_{d\in\mathcal{D}(\gamma)}(F(\gamma, a, d)))\\ &\quad \to \Box(\forall((\gamma, a) : \text{よ}(\Gamma.A)) \to \operatorname*{colim}_{d\in\mathcal{D}(\gamma)}(F(\gamma, a, d))).\end{aligned}$$

Now recall that colimits are computed objectwise in presheaves. This provides isomorphisms

$$\begin{aligned}&\Box(\forall(\gamma : \text{よ}(\Gamma)) \to \operatorname*{colim}_{d\in\mathcal{D}(\gamma)}((a : A(\gamma)) \to F(\gamma, a, d)))\\ &\quad \to \operatorname*{colim}_{d:\Box(\Pi_\gamma\mathcal{D}(\gamma))}(\Box(\forall\gamma \to (a : \gamma) \to F(\gamma, a, d(\gamma))))\end{aligned}$$

and

$$\begin{aligned}&\Box(\forall((\gamma, a) : \text{よ}(\Gamma.A)) \to \operatorname*{colim}_{d\in\mathcal{D}(\gamma)}(F(\gamma, a, d)))\\ &\quad \to \operatorname*{colim}_{d:\Box(\Pi_{(\gamma,a)}\mathcal{D}(\gamma))}(\Box(\forall(\gamma, a) \to F(\gamma, a, d(\gamma, a)))).\end{aligned}$$

Because $\mathcal{D}(\gamma)$ is $A(\gamma)$-null, we have $\Pi_{(\gamma,a)}\mathcal{D}(\gamma) \cong \Pi_\gamma \mathcal{D}(\gamma)$.
By composing the four isomorphisms above, we obtain an isomorphism

$$\begin{aligned}&\Box(\forall\gamma \to \operatorname*{colim}_{d\in\mathcal{D}(\gamma)}((a : A(\gamma)) \to F(\gamma, a, d)))\\ &\quad\to \Box(\forall\gamma \to (a : A(\gamma)) \to \operatorname*{colim}_{d\in\mathcal{D}(\gamma)}(F(\gamma, a, d))),\end{aligned}$$

as needed.

If we track the underlying map of this isomorphism, we see that for any $\gamma : \text{Ł}(\Gamma)$, an element $\iota_d(f)$ is in correpondance with $\lambda a \mapsto \iota_d(f(a))$, as desired. □

2.3.5 Levelwise decidable propositions

We now discuss levelwise decidable propositions in presheaf toposes. A presheaf X is a levelwise decidable proposition if for every Γ, the set $X(\Gamma)$ is an external decidable proposition.

In this thesis, the levelwise decidable propositions will be used in proofs of decidability of equality for type theory. Decidability of equality for the syntax of a type theory can be proven by first proving normalization, and then proving that normal forms have decidable equality. We will see that normal forms can concisely be defined as inductive families, in the internal language of some presheaf topos. This definition is suitable for the normalization proofs, but the decidability of equality then presents a challenge, because these normal forms actually fail to have decidable equality in the presheaf topos. However we can still prove that they have levelwise decidable equality, which is sufficient to entail the decidability of equality for the syntax. Using the classifier for levelwise decidable proposition and an axiomatization of its closure properties, we can carry the proof completely internally to the presheaf topos.

The levelwise decidable propositions are also important for other applications outside of the scope of this thesis. For example, the realignment operations of Orton and Pitts (2018), which is used to construct Glue-type in presheaf models of cubical type theory, are only constructively definable over partial isomorphisms whose partiality is controlled by a levelwise decidable proposition.

Definition 2.3.21. A dependent presheaf $Y : \mathcal{C}^{\mathsf{op}} \to \mathbf{Fam}[X]$ is a **levelwise decidable proposition** if for every Γ and $x \in X(\Gamma)$, the set $Y(\Gamma, x)$ is a decidable proposition. ⌟

Proposition 2.3.22. *There is a classifier* $\mathsf{Prop}_{\mathsf{ldec}} \hookrightarrow \mathsf{Prop}$ *of levelwise decidable propositions.*

Proof. This follows from lemma 2.3.9, observing that the set $\mathsf{LevelwiseDec}(X)$ of levelwise decidable propositions over a base presheaf X can be unfolded as follows:

$$\begin{aligned}&\mathsf{LevelwiseDec}(X) \cong\\ &\quad (Y_0 : \forall c_0\ (x : X(c_0)) \to \{\top, \bot\})\\ &\times (Y_1 : \forall (c_1 \xrightarrow{f_1} c_0)\ (x : X(c_0)) \to Y_0(c_0, x) \to Y_0(c_1, x[f_1])).\end{aligned}$$ □

As with the classifier $\mathsf{Set}_{\mathsf{rep}}$ of locally representable sets, the existence of a classifier of levelwise decidable propositions means that we can meaningfully talk about them in the internal language of $\mathbf{Psh}(\mathcal{C})$.

Definition 2.3.23 (In the internal language)**.** We say that a set $X : \mathsf{Set}^{\mathsf{c}}$ has **levelwise decidable equality** if for every $x, y : X$, $(x = y)$ is a levelwise decidable proposition. ⌟

Lemma 2.3.24. *The levelwise decidable propositions are closed under:*

- *Limits and colimits of propositions indexed by discrete* $\mathfrak{c}$*-categories. This includes conjunctions and disjunctions, as well as dependent sums.*
- *Quantification over locally representable* $\mathfrak{c}$*-sets: if* A *is locally representable and* $P : A \to \mathsf{Prop}_{\mathsf{ldec}}$*, then* $((a : A) \to P(a)) : \mathsf{Prop}_{\mathsf{ldec}}$.

Proof. The closure under limits and colimits indexed by discrete $\mathfrak{c}$-categories follows from the fact that limits and colimits are computed levelwise in presheaf toposes. Note that a colimit of propositions should be be computed as the propositional truncation of the colimit of their underlying sets.

We now check the closure under quantification over locally representable sets. We have to define a global map

$$(A : \mathsf{Set}_{\mathsf{rep}})(P : A \to \mathsf{Prop}_{\mathsf{ldec}}) \to ((a : A) \to P(a)) \in \mathsf{Prop}_{\mathsf{ldec}}.$$

It suffices to show that the global family

$$(A : \mathsf{Set}_{\mathsf{rep}})(P : A \to \mathsf{Prop}_{\mathsf{ldec}}) \mapsto ((a : A) \to P(a))$$

is levelwise decidable. Take $\Gamma \in \mathcal{C}$ and elements $A :: \text{ぇ}(\Gamma) \to \mathsf{Set}_{\mathsf{rep}}$ and $P :: \text{ぇ}(\Gamma) \to A \to \mathsf{Prop}_{\mathsf{ldec}}$. We have to prove that the set of global elements of $\forall\gamma \to (a : A(\gamma)) \to P(\gamma, a)$ is decidable. But since A is locally representable, we have $\Gamma.A \in \mathcal{C}$ and an isomorphism $\text{ぇ}(\Gamma.A) \cong (\gamma : \text{ぇ}(\Gamma)) \times A(\gamma)$. Transporting over this isomorphism, we have $P' :: \text{ぇ}(\Gamma.A) \to \mathsf{Prop}_{\mathsf{ldec}}$ and we have to prove that the set of global elements of $\forall\gamma \to P'(\gamma)$ is levelwise decidable. But this now follows from P' being levelwise decidable. □

2.4 Factorization systems

The notion of weak factorization system is central to the theory of model categories, which can be seen as presentations of ∞-categories using 1-categorical methods. In this thesis, we will use some 1-categories in situations where we morally have some $(2,1)$-categories or 2-categories. One of the main reasons for this is that objects defined by 1-categorical universal properties are easier to construct (using for example QIITs or adjoint functor theorems) and to compute with (because they can be understood syntactically).

The counterpart is that morphisms between general objects in these 1-categories are sometimes ill-behaved, and we need to consider instead morphisms with a cofibrant source (and a fibrant target, but for our applications all objects will be fibrant), where the notions of cofibrations and fibrations are given by factorization systems.

A factorization system consists of two classes of maps, called left maps and right maps. For out applications, a typical left map will be a free extension, such as the inclusion from a ring to its polynomial ring, the free extension of a category with a new object or morphism, the free extension of a model of type theory with a new type or term, etc. A right map will be a morphism with some surjective or bijective components, e.g. a fully-faithful functor (surjective on objects and bijective on morphisms), etc. In particular, general definitions of free extensions of algebras for arbitrary generalized algebraic theories will involve factorization systems.

Definition 2.4.1. Let $l : A \to B$ and $r : X \to Y$ be two morphisms in a category $\mathcal{B}$.

- A **lifting problem** of l against r consists of morphisms $f : A \to X$ and $g : B \to Y$ such that the square

$$\begin{array}{ccc} A & \xrightarrow{f} & X \\ {\scriptstyle l}\downarrow & \overset{h}{\nearrow} & \downarrow{\scriptstyle r} \\ B & \xrightarrow[g]{} & Y \end{array}$$

  commutes. A **diagonal lift** is a morphism $h : B \to X$ such that $l = f \circ h$ and $r = h \circ g$.

- A **lifting structure** consists, for every lifting problem, of a chosen diagonal lift.

- We say that the **lifting property** holds (left lifting of l against r, or right lifting of r against l), and write $l \boxslash r$, if there merely exists a lifting structure.

- We say that the **orthogonal lifting property** holds, and write $l \perp r$, if for every lifting problem, there exists a unique diagonal lift. ⌟

Definition 2.4.2. A **weak factorization system** in a category $\mathcal{B}$ consists of classes $\mathcal{L}$ of *left maps* and $\mathcal{R}$ of *right maps*, such that

$$\mathcal{R} = \mathcal{L}^{\boxslash}$$

and

$$\mathcal{L} = {}^{\boxslash}\mathcal{R},$$

and such that every morphism in $\mathcal{B}$ can be factored as a left map followed by a right map, where

$$\mathcal{L}^{\boxslash} \triangleq \{r \mid \forall l \in \mathcal{X}, l \boxslash r\},$$
$${}^{\boxslash}\mathcal{R} \triangleq \{r \mid \forall l \in \mathcal{R}, l \boxslash r\}$$ ⌟

Definition 2.4.3. An **orthogonal factorization system** in a category $\mathcal{B}$ consists of classes $\mathcal{L}$ of *left maps* and $\mathcal{R}$ of *right maps*, such that

$$\mathcal{R} = \mathcal{R}^{\perp}$$

and

$$\mathcal{L} = {}^{\perp}\mathcal{R}$$

and such that every morphism in $\mathcal{B}$ can be factored as a left map followed by a right map, where

$$\mathcal{L}^{\perp} \triangleq \{r \mid \forall l \in \mathcal{X}, l \perp r\},$$
$${}^{\perp}\mathcal{R} \triangleq \{r \mid \forall l \in \mathcal{R}, l \perp r\}$$ ⌟

Proposition 2.4.4. *For any map $l : A \to B$ and $r : X \to Y$, we have $l \perp r$ if and only if both $l \boxslash r$ and $\nabla l \boxslash r$, where ∇l is the codiagonal map*

$$\nabla l : (B +_A B) \to B.$$

Furthermore, $\nabla l \boxslash r$ if and only if $l \boxslash \Delta r$, where Δr is the diagonal map

$$\Delta r : X \to (X \times_Y X).$$

Proof. Consider a lifting problem

$$\begin{array}{ccc} (B +_A B) & \xrightarrow{\langle h_1, h_2 \rangle} & X \\ {\scriptstyle \nabla l}\downarrow & & \downarrow{\scriptstyle r} \\ B & \xrightarrow{g} & Y, \end{array}$$

where $h_1, h_2 : B \to X$ are such that $h_1 \circ l = h_2 \circ l$. Pose $f = h_1 \circ l$.

Then h_1, h_2 are solutions to the lifting problem

$$\begin{array}{ccc} A & \xrightarrow{f} & X \\ {\scriptstyle l}\downarrow & \overset{h}{\nearrow} & \downarrow{\scriptstyle r} \\ B & \xrightarrow[g]{} & Y, \end{array}$$

and any two solutions to that lifting problem fit in the above diagram.

One can check that the above lifting problem against ∇l admits a unique solution if and only if $h_1 = h_2$. The solution is then h_1.

Thus $\nabla l \boxslash r$ is equivalent to the uniqueness of solutions to lifting problems of l against r. Dually, $l \boxslash \Delta r$ is also equivalent to the uniqueness of solutions to lifting problems of l against r. The result follows from these facts. □

Proposition 2.4.5. *An orthogonal factorization system is exactly a weak factorization system such that for every left map $l : A \to B$, the codiagonal map*

$$\nabla l : (B +_A B) \to B$$

is also a left map.

Proof. We have to show a logical equivalence

$$(\mathcal{L} = {}^{\perp}\mathcal{R} \wedge \mathcal{R} = \mathcal{L}^{\perp}) \iff (\mathcal{L} = {}^{\boxslash}\mathcal{R} \wedge \mathcal{R} = \mathcal{L}^{\boxslash} \wedge \nabla\mathcal{L} \subseteq \mathcal{L}),$$

where $\nabla\mathcal{L}$ is the set of codiagonal maps of maps in $\mathcal{L}$.

Note that for any class $\mathcal{X}$ of maps, we have ${}^{\perp}\mathcal{X} \subseteq {}^{\boxslash}\mathcal{X}$ and $\mathcal{X}^{\perp} \subseteq \mathcal{X}^{\boxslash}$.

We first prove that when $\nabla\mathcal{L} \subseteq \mathcal{L}$ and $\Delta\mathcal{R} \subseteq \mathcal{R}$, we have $\mathcal{L}^{\perp} = \mathcal{L}^{\boxslash}$ and ${}^{\perp}\mathcal{R} = {}^{\boxslash}\mathcal{R}$. This follows directly from the fact, proven in proposition 2.4.4, that $\mathcal{L}^{\perp} = \mathcal{L}^{\boxslash} \cap (\nabla\mathcal{L})^{\boxslash}$ and its dual ${}^{\perp}\mathcal{R} = {}^{\boxslash}\mathcal{R} \cap {}^{\boxslash}(\Delta\mathcal{R})$.

We then prove the forward direction. Assume that $\mathcal{L} = {}^{\perp}\mathcal{R}$ and $\mathcal{R} = \mathcal{L}^{\perp}$.

By proposition 2.4.4, $\mathcal{L} \subseteq {}^{\perp}\mathcal{R}$ implies $\nabla\mathcal{L} \subseteq {}^{\boxslash}\mathcal{R}$, and moreover $\nabla\mathcal{L} \subseteq {}^{\boxslash}\mathcal{R}$ if and only if $\nabla\mathcal{L} \subseteq {}^{\perp}\mathcal{R}$. Thus $\nabla\mathcal{L} \subseteq \mathcal{L}$. A formally dual argument shows that $\Delta\mathcal{R} \subseteq \mathcal{R}$.

We then have $\mathcal{L}^{\perp} = \mathcal{L}^{\boxslash}$ and ${}^{\perp}\mathcal{R} = {}^{\boxslash}\mathcal{R}$, so $\mathcal{L} = {}^{\boxslash}\mathcal{R}$ and $\mathcal{R} = \mathcal{L}^{\boxslash}$, as needed.

For the converse implication, note that for a weak factorization system, $\nabla\mathcal{L} \subseteq \mathcal{L}$ if and only if $\Delta\mathcal{R} \subseteq \mathcal{R}$, as proven in proposition 2.4.4. A formally dual argument shows that $\Delta\mathcal{R} \subseteq \mathcal{R}$.

Thus $\mathcal{L}^{\perp} = \mathcal{L}^{\boxslash}$ and ${}^{\perp}\mathcal{R} = {}^{\boxslash}\mathcal{R}$, so $\mathcal{L} = {}^{\perp}\mathcal{R}$ and $\mathcal{R} = \mathcal{L}^{\perp}$, as needed. □

Definition 2.4.6. Let I be a (small) set of maps.

- An **I-fibration** is a map with the right lifting property against all maps in I.
- An **I-cofibration** is a map with the left lifting property against all I-fibrations.

- A **split I-fibration** is a map equipped with a right lifting structure against all maps in I.
- An **algebraic I-cofibration** is a map equipped with a left lifting structure against all split I-fibrations.
- A **basic I-cellular map** is a pushout of a map in I:

$$\begin{array}{ccc} A & \longrightarrow & X \\ \downarrow{\scriptstyle i\in I} & \ulcorner & \downarrow \\ B & \longrightarrow & (X +_A B) \end{array}$$

  We say that $(X +_A B)$ is a basic I-cellular extension of X.
- A **finite I-cellular map** is a finite composition $X_0 \to \cdots \to X_n$ of basic I-cellular maps. We say that X_n is a finite I-cellular extension of X_0. ⌟

There are multiple possible general definitions of general I-cellular maps, which are equivalent classically but differ constructively.

- We can consider sequential compositions (or larger transfinite compositions) of basic I-cellular maps. This can be problematic constructively, because this requires an ordering on the generators (which could otherwise be provided by the well-ordering principle).
- The usual definition of I-cellular map involves transfinite compositions of pushouts of coproducts of maps in I, i.e. it considers adding multiple cells with the same boundary at the same time.
- We can consider fat I-cellular maps, which are filtered colimits of finite I-cellular extension, where the maps in the filtered diagram are required to be renamings of finite I-cellular extensions: morphisms induced by maps between the generators of the extensions.

  Such fat I-cellular maps are used in the fat small object argument of Makkai, Rosický, and Vokřínek (2014), but haven't been studied in a constructive metatheory to the best of my knowledge. The axiom of choice is used in Makkai, Rosický, and Vokřínek 2014, Proposition 4.5 to show that every fat I-cellular map is also a sequential I-cellular map, but doesn't seem to be required for the other constructions presented in that paper.

Theorem 2.4.7 (Quillen's small object argument)**.** *If $\mathcal{B}$ is a locally finitely presentable category, then for any set I of maps, (I-cofibrations, I-fibrations) forms a weak factorization system, said to be cofibrantly generated by I. Moreover any map can be factored as an I-cellular map followed by an I-fibration.*

Because orthogonal factorization systems are a special case of weak factorization systems, we obtain a small object argument for orthogonal factorization systems. I have seen this result attributed to Gabriel and Ulmer (1971).

Corollary 2.4.8 (Small object argument for orthogonal factorization systems)**.** *If $\mathcal{B}$ is a locally finitely presentable category, then for any set I of maps, ($(I \cup \nabla I)$-cofibrations, $(I \cup \nabla I)$-fibrations) forms an orthogonal factorization system, said to be generated by I.*

While the small object argument itself is valid constructively, its applications can be limited by the fact that it produces non-split I-fibrations. One way to resolve this issue is to use the algebraic small object argument of Garner (2007), which produces an algebraic weak factorization system instead.

We won't encounter such issues, because only orthogonal factorization systems are used in this thesis.

I conjecture that the fat small object argument Makkai, Rosický, and Vokřínek 2014 could be also used to construct well-behaved factorizations.

Conjecture 2.4.9. If $\mathcal{B}$ is locally finitely presentable, then every map can be factored as a fat I-cellular map followed by a split I-fibration. ⌟

Chapter 3

Categorical models of type theory

We review the notion of categories with families (CwFs), due by Dybjer (1995), and some core type-theoretic structures (Σ-types and Π-types).

Categories with families are a notion of categorical model for type theory. Many closely related notions of categorical models for type theory exist: clans, categories with attributes, natural models, display map categories, contextual categories, C-systems, B-systems, etc., sometimes with multiple names for the same definition.

CwFs are quite canonical due to their presentation as models of a generalized algebraic theory. This generalized algebraic theory will be presented later. Since the definition of generalized algebraic theory will involve some CwFs, bootstrapping the definition of generalized algebraic theories will rely on the constructions of this section.

3.1 Categories with families

3.1.1 Categories with families

Definition 3.1.1. A **category with families** (CwF) is a category $\mathcal{C}$ equipped with:

- A terminal object $1_{\mathcal{C}}$.
- A presheaf $\mathsf{Ty} : \mathcal{C}^{\mathsf{op}} \to \mathbf{Set}$.
- A dependent presheaf $\mathsf{Tm} : \mathcal{C}^{\mathsf{op}} \to \mathbf{Fam}[X]$.
- A local representability structure on Tm. ⌟

Remark that the presheaf Ty, the dependent presheaf Tm and the locally representable structure on Tm correspond to $\mathsf{ty} :: \Box(\mathsf{Set}^{\mathsf{c}})$ and $\mathsf{tm} :: \Box(\mathsf{Ty} \to \mathsf{Set}_{\mathsf{rep}})$ in the modal internal language of $\mathbf{Psh}(\mathcal{C})$. Elements of $\mathcal{C}.\mathsf{Ty}(\Gamma)$ then correspond to elements of $\Box(\text{ⅅ}(\Gamma) \to \mathsf{ty})$, and elements of $\mathcal{C}.\mathsf{Tm}(\Gamma, A)$ correspond to elements of $\Box((\gamma : \text{ⅅ}(\Gamma)) \to \mathsf{tm}(A(\gamma)))$.

We may write $(\Gamma \vdash A\ \mathsf{type}) \in \mathcal{C}$ to indicate that $A : \mathcal{C}.\mathsf{Ty}(\Gamma)$, or $(\Gamma \vdash a : A) \in \mathcal{C}$ when $a : \mathcal{C}.\mathsf{Tm}(\Gamma, A)$.

Definition 3.1.2. A (strict) morphism of CwFs from $\mathcal{C}$ to $\mathcal{D}$ consists of a functor $F : \mathcal{C} \to \mathcal{D}$, together with actions on types and terms:

$$F : \mathcal{C}.\mathsf{Ty}(\Gamma) \to \mathcal{D}.\mathsf{Ty}(F(\Gamma)),$$
$$F : \mathcal{C}.\mathsf{Tm}(\Gamma, A) \to \mathcal{D}.\mathsf{Tm}(F(\Gamma), F(A)),$$

and such that the terminal object is preserved strictly $F(1_{\mathcal{C}}) = 1_{\mathcal{D}}$, and the context extensions are preserved strictly:

$$\begin{aligned}
&F(\Gamma.A) = F(\Gamma).F(A),\\
&F(\langle\gamma, a\rangle) = \langle F(\gamma), F(a)\rangle,\\
&F(\boldsymbol{p}_A) = \boldsymbol{p}_{F(A)},\\
&F(\boldsymbol{q}_A) = \boldsymbol{q}_{F(A)}.
\end{aligned}$$ ⌟

We omit the rest of the definition of the category **CwF** of CwFs. We admit without proof that this category, as well as the categories $\mathbf{CwF}_{\Sigma}$ and $\mathbf{CwF}_{\Sigma,\Pi_{\mathsf{rep}}}$ that are defined later in this chapter, are complete and cocomplete. We would normally obtain these results from the presentation of these categories as the categories of algebras of generalized algebraic theories. We cannot do this at this point since we are still bootstrapping the theory of GATs.

Note that limits in **CwF** are quite easy to define explicitly, as they can be computed sortwise. Colimits can be constructed as QIITs in a metatheory with QIITs (this seems circular because QIITs are just the initial algebras of generalized algebraic theories, but specifying specific QIITs does not require GATs; it would be possible to have a proof assistant with support for QIITs independently of any mechanization of GATs). Alternatively, **CwF** could be shown to be equivalent to the category of algebras of a finite-limits sketch; bootstrapping sketches could be simpler than bootstrapping GATs.

3.1.2 Contextuality

For some applications, we need to consider contextual CwFs, which are CwFs whose objects are really "contexts", i.e. lists of types. One of the motivations is that in the "language of type theory", we never actually mention contexts, but only types and terms. The base category of a CwF is only included because it is needed to specify the types and terms. Contextual CwFs were introduced by Cartmell (1986) under the name of contextual categories. Voevodsky (2016) studied them under the name of C-systems.

We ideally want CwFs that are indistinguishable using statements in the "language of type theory" to be indistinguishable. In particular, CwFs with isomorphic families of types and terms but potentially different base categories should be considered the same. This is captured by the following definition:

Definition 3.1.3. A CwF morphism $F : \mathcal{C} \to \mathcal{D}$ is said to be a **contextual isomorphism** if its actions on types and terms are bijective, i.e. if the maps

$$\begin{aligned}
&F : \mathcal{C}.\mathsf{Ty}(\Gamma) \to \mathcal{D}.\mathsf{Ty}(F(\Gamma)),\\
&F : \mathcal{C}.\mathsf{Tm}(\Gamma, A) \to \mathcal{D}.\mathsf{Tm}(F(\Gamma), F(A))
\end{aligned}$$

are bijective. ⌟

The contextual isomorphisms are the maps in the right class of the orthogonal factorization system generated by $\{I^{\mathsf{ty}}, I^{\mathsf{tm}}\}$, where I^{ty} and I^{tm} are the generic extensions of a CwF by a new type or term.

$$\begin{aligned}
&I^{\mathsf{ty}} : \mathsf{Free}_{\mathbf{CwF}}(\boldsymbol{\Gamma} \vdash) \to \mathsf{Free}_{\mathbf{CwF}}(\boldsymbol{\Gamma} \vdash \boldsymbol{A}\ \mathsf{type}),\\
&I^{\mathsf{tm}} : \mathsf{Free}_{\mathbf{CwF}}(\boldsymbol{\Gamma} \vdash \boldsymbol{A}\ \mathsf{type}) \to \mathsf{Free}_{\mathbf{CwF}}(\boldsymbol{\Gamma} \vdash \boldsymbol{a} : \boldsymbol{A}).
\end{aligned}$$

The maps in the left class of the factorization system are called **left contextual** maps.

Definition 3.1.4. A CwF $\mathcal{C}$ is said to be **contextual** if the map $\mathbf{0}_{\mathbf{CwF}} \to \mathcal{C}$ is left contextual. The **contextual core** of a CwF $\mathcal{C}$ is the contextual CwF obtained by the factorization of $\mathbf{0}_{\mathbf{CwF}} \to \mathcal{C}$ as a left contextual map $\mathbf{0}_{\mathbf{CwF}} \to \mathsf{cxl}(\mathcal{C})$ followed by a contextual isomorphism $\mathsf{cxl}(\mathcal{C}) \to \mathcal{C}$. ⌟

The contextual CwFs form a coreflective subcategory $\mathbf{CwF}^{\mathsf{cxl}}$ of $\mathbf{CwF}$, with $\mathsf{cxl} : \mathbf{CwF} \to \mathbf{CwF}^{\mathsf{cxl}}$ as the right adjoint.

This definition of contextuality using an orthogonal factorization system is due to Sattler (2018). It automatically provides all of the structure that we have in presence of an orthogonal factorization system. The more traditional definition of contextuality is more explicit and involves an inductive characterization of contexts, often phrased using a length function that assigns a unique length to every context.

The equivalence between the two definitions can be understood intuitively from the fact that $\mathsf{cxl}(\mathcal{C})$ is initial among CwFs with a contextual isomorphism into $\mathcal{C}$. This means that it has the same types and terms as $\mathcal{C}$, but its objects should be freely generated by the CwF operations. The morphisms are then uniquely determined. The only CwF operations creating objects are the empty context and context extensions, so the objects of $\mathsf{cxl}(\mathcal{C})$ can be written uniquely as an iterated context extension over the empty context.

Definition 3.1.5. Let $\mathcal{C}$ be a CwF. The **telescopes** of $\mathcal{C}$ are an inductive-recursive family generated by

$$
\begin{aligned}
&\mathsf{Ty}^\star : \mathcal{C}.\mathsf{Ob} \to \mathsf{Set},\\
&-.- : (\Gamma : \mathcal{C}.\mathsf{Ob}) \to \mathsf{Ty}^\star(\Gamma) \to \mathcal{C}.\mathsf{Ob},\\
&\varepsilon : \mathsf{Ty}^\star(\Gamma),\\
&- \triangleright - : (\Delta : \mathsf{Ty}^\star(\Gamma)) \to \mathsf{Ty}(\Gamma.\Delta) \to \mathsf{Ty}^\star(\Gamma),\\
&\Gamma.\varepsilon \triangleq \Gamma,\\
&\Gamma.(\Delta \triangleright A) \triangleq (\Gamma.\Delta).A.
\end{aligned}
$$

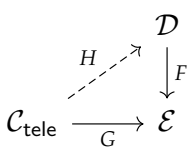
(Note that this instance of induction-recursion can be replaced by an inductive type indexed by the length, or by induction over the length of contexts). ⌟

Proposition 3.1.6. *The contextual core of a CwF $\mathcal{C}$ admits the following description:*

- *the objects of $\mathsf{cxl}(\mathcal{C})$ are closed telescopes (elements of $\mathsf{Ty}^\star(1_\mathcal{C})$).*
- *the other components are defined so that $1_\mathcal{C}.- : \mathsf{Ty}^\star(1_\mathcal{C}) \to \mathcal{C}.\mathsf{Ob}$ becomes the action on objects of a CwF morphism with bijective actions on morphisms, types and terms.*
- *the context extension is interpreted by the operation $- \triangleright -$.*

Proof. Write $\mathcal{C}_{\mathsf{tele}}$ for the CwF defined in the statement, and $1_\mathcal{C}.- : \mathcal{C}_{\mathsf{tele}} \to \mathcal{C}$ for the inclusion morphisms. By definition, the map $1_\mathcal{C}.- : \mathcal{C}_{\mathsf{tele}} \to \mathcal{C}$ is a contextual isomorphism, so it suffices to prove that $\mathcal{C}_{\mathsf{tele}}$ is contextual, i.e. that $\mathbf{0} \to \mathcal{C}_{\mathsf{tele}}$ is left contextual.

Take a lifting problem consisting of a contextual isomorphism $F : \mathcal{D} \to \mathcal{E}$ and a CwF morphism $G : \mathcal{C}_{\mathsf{tele}} \to \mathcal{E}$. We write F^{-1} for the inverses of the actions of F on types and terms.



Any factorization $H : \mathcal{C}_{\mathsf{tele}} \to \mathcal{D}$ of G through F must satisfy the equations

$$\begin{aligned}
H(\varepsilon) &= 1_{\mathcal{D}}, && \text{(action on empty contexts)}\\
H(\Delta.A) &= H(\Delta).H(A), && \text{(action on extended contexts)}\\
H(A) &= F^{-1}(G(A)), && \text{(action on types)}\\
H(a) &= F^{-1}(G(a)), && \text{(action on terms)}\\
H(\langle\rangle) &= \langle\rangle, && \text{(action on empty substitutions)}\\
H(\langle f, a\rangle) &= \langle H(f), H(a)\rangle. && \text{(action on extended substitutions)}
\end{aligned}$$

By induction over telescopes, we define a morphism H satisfiying these equations. Thus $\mathcal{C}_{\mathsf{tele}}$ is contextual, as needed.

Note that because any orthogonal factorization system is in particular a weak factorization system, we did not need to prove the uniqueness of H. □

3.1.3 Democracy

Democracy weakens contextuality by only requiring that every object is isomorphic to an iterated context extension.

Definition 3.1.7. A **democracy structure** on a CwF $\mathcal{C}$ the data of an essential section of $\mathsf{cxt}(\mathcal{C}) \to \mathcal{C}$: for every $\Gamma \in \mathcal{C}$, there is $K(\Gamma) \in \mathsf{cxt}(\mathcal{C})$ and an isomorphism $1_{\mathcal{C}}.K(\Gamma) \cong \Gamma$. ⌟

Note that since $\mathsf{cxt}(\mathcal{C}) \to \mathcal{C}$ is always fully faithful, the CwF $\mathcal{C}$ is democratic if and only if $\mathsf{cxt}(\mathcal{C}) \to \mathcal{C}$ is an equivalence of categories.

Proposition 3.1.8. *Any contextual CwF is democratic.* □

Definition 3.1.9. The structure of **constant families** on a CwF $\mathcal{C}$ consists, for every $\Gamma \in \mathcal{C}$, of a closed type $K(\Gamma)$ and an isomophism $\Gamma \cong 1.K(\Gamma)$. ⌟

Proposition 3.1.10. *If a CwF has constant families, then it is democratic. Conversely, if a CwF with Σ-types is democratic, then it has constant families.*

Proof. If a CwF has constant families, then every context is isomorphic to a telescope of length 1.

If a CwF with Σ-types is democratic, then every context is isomorphic to a telescope of some length $n \leq 0$, which we can reduce to an isomorphic closed type using iterated Σ-types (and $\mathbf{1}$ for the case $n = 0$). □

3.1.4 Trivial fibrations

We can also consider the weak factorization system generated by the same set of maps $I = \{I^{\mathsf{ty}}, I^{\mathsf{tm}}\}$ in **CwF**. The maps in the left class are called **cofibrations**. The maps in the right class are called **trivial fibrations**, they are the CwF morphisms with surjective actions on types and terms.

Definition 3.1.11. A CwF morphism $F : \mathcal{C} \to \mathcal{D}$ is said to be a **trivial fibration** if its actions on types and terms are surjective, i.e. if it satisfies the following lifting conditions:

strict type lifting For every $\Gamma \in \mathcal{D}$ and type $(F(\Gamma) \vdash A\ \mathsf{type}) \in \mathcal{D}$, there merely exists a type $(\Gamma \vdash A_0\ \mathsf{type}) \in \mathcal{C}$ such that $F(A_0) = A$.

strict term lifting For every $(\Gamma \vdash A \text{ type}) \in \mathcal{C}$ and term $(F(\Gamma) \vdash a : F(A)) \in \mathcal{D}$, there merely exists a term $(\Gamma \vdash a_0 : A) \in \mathcal{C}$ such that $F(a_0) = a$. ⌟

Logically, the strict term lifting property is a kind of conservativity property: any statement (type) of the source theory that is provable in the target theory is already provable in the source theory.

Hofmann (1995) has proven that extensional type theory (ETT) is conservative over intensional type theory (ITT) with function extensionality and uniqueness of identity proofs. Hofmann's conservativity theorem can be stated using trivial fibrations: the canonical CwF morphism $\mathbf{0}_{\mathsf{ITT}} \to \mathbf{0}_{\mathsf{ETT}}$ between the initial models of ITT and ETT is a trivial fibration.

Kapulkin and Lumsdaine (2016) have constructed a left semi-model structures on categories of CwFs with identity types, Σ-types, and optionally Π-types with function extensionality, with (cofibrations, trivial fibrations) as one of the underlying weak factorization systems.

Constructively, one has to be careful to distinguish these trivial fibrations from *algebraic trivial fibrations*, which are equipped with a choice of solution to every lifting problem.

As a consequence of the small object argument, every cofibration is a retract of an I-cellular map. An I-cellular object is a CwF that is freely generated by a collection of types and terms. The I-cellular maps are the inclusion maps for free extensions of CwFs obtained by adjoining new types and terms.

Proposition 3.1.12. *Any cofibration in* **CwF** *is also a left contextual map. In particular, any cofibrant CwF is contextual.*

Proof. This follows from the fact that any contextual isomorphism is also a trivial fibration. □

3.1.5 Renamings

We describe the CwF of renamings of any base CwF $\mathcal{C}$. A renaming is a substitution that is built by repeatedly extending the empty substitution by a variable.

There are two ways to define the category of renamings. It can be defined by inductively characterizing the substitutions that are renamings. It can also be defined with a universal property, as an initial *renaming algebra*. This second definition is a variant of a definition originally due to Sattler (2018). Both definitions are useful; the universal property will be used when proving relative induction principles in chapter 6.

Definition 3.1.13. A **renaming algebra** over a CwF $\mathcal{C}$ is a CwF $\mathcal{R}$ equipped with a CwF morphism $F : \mathcal{R} \to \mathcal{C}$ such that $\mathcal{R}.\mathsf{Ty} = F^*(\mathcal{C}.\mathsf{Ty})$ and the action of F on types is the identity. ⌟

Almost equivalently, we can ask that the action of $F : \mathcal{R} \to \mathcal{C}$ on types is bijective. However this alternative definition is not essentially algebraic, because it involves equations in $\mathcal{C}$. The above definition is essentially algebraic, because the equality $\mathcal{R}.\mathsf{Ty} = F^*(\mathcal{C}.\mathsf{Ty})$ can be substituted in the specification of the components of $\mathcal{R}$.

The category of renaming algebras over $\mathcal{C}$ is the category of algebras of an essentially algebraic theory. In particular, it has an initial object $\mathbf{Ren}(\mathcal{C})$, which we call the CwF of renamings of $\mathcal{C}$. The terms of $\mathbf{Ren}(\mathcal{C})$ are called **variables**, and we will write Var for the family of variables.

Proposition 3.1.14. *The CwF of renamings* $\mathbf{Ren}(\mathcal{C})$ *is contextual.*

Proof. The CwF morphism $\mathsf{cxl}(\mathbf{Ren}(\mathcal{C})) \to \mathbf{Ren}(\mathcal{C})$ has a bijective action on types. This equips $\mathsf{cxl}(\mathbf{Ren}(\mathcal{C}))$ with the structure of a renaming algebra over $\mathcal{C}$. By the universal property of $\mathbf{Ren}(\mathcal{C})$, there is a unique section of $\mathsf{cxl}(\mathbf{Ren}(\mathcal{C})) \to \mathbf{Ren}(\mathcal{C})$ in the category of renaming algebras, hence also a unique section in the category of CwFs. Thus $\mathbf{Ren}(\mathcal{C})$ is a retract of $\mathsf{cxl}(\mathbf{Ren}(\mathcal{C}))$. Since contextual CwFs are closed under retracts, $\mathbf{Ren}(\mathcal{C})$ is contextual. □

We can show that this definition of the CwF of renamings is isomorphic to the inductive definition of renamings.

Proposition 3.1.15. *The CwF of renamings admits the following explicit description:*

- *An object of* $\mathbf{Ren}(\mathcal{C})$ *is an object of* $\mathsf{cxl}(\mathcal{C})$*, i.e. a closed telescope.*
- *The morphisms and variables are inductive families generated by:*

$$\begin{aligned}
&\langle\rangle : (\Gamma \to \varepsilon),\\
&\langle -, - \rangle : (f : \Gamma \to \Delta) \to \mathsf{Var}(\Gamma, A[f]) \to (\Gamma \to (\Delta \triangleright A)),\\
&\boldsymbol{q} : \mathsf{Var}(\Gamma \triangleright A, A[\boldsymbol{p}_A]),\\
&-[\boldsymbol{p}] : \mathsf{Var}(\Gamma, X) \to \mathsf{Var}(\Gamma \triangleright A, X[\boldsymbol{p}_A]).
\end{aligned}$$

- *Compositions of morphisms and substitutions of variables are determined by the following equations:*

$$\begin{aligned}
&\boldsymbol{q}[\langle f, x\rangle] = x,\\
&v[\boldsymbol{p}][\langle f, x\rangle] = v[f],\\
&\langle\rangle \circ g = \langle\rangle,\\
&\langle f, x\rangle \circ g = \langle f \circ g, x[g]\rangle.
\end{aligned}$$

- *The identities and projections* $\boldsymbol{p}$ *are defined by induction on telescopes.*

Proof. This defines a renaming algebra $\mathcal{R}$. It then suffices to show that it is initial.
For any other renaming algebra $\mathcal{D}$, we define a renaming algebra morphism $\mathcal{R} \to \mathcal{D}$.

- The action on objects is defined by induction on telescopes.
- The action on morphisms and variables are defined by induction on the above inductive families. □

Proposition 3.1.16. *If* $\mathcal{C}$ *is contextual, then the action on objects of the morphism* $\mathbf{Ren}(\mathcal{C}) \to \mathcal{C}$ *is bijective.*

Proof. This follows from the characterization of the objects of $\mathbf{Ren}(\mathcal{C})$ from proposition 3.1.15. □

Proposition 3.1.17. *For any* $(\Gamma \vdash A\ \mathsf{type}) \in \mathbf{Ren}(\mathcal{C})$*, the set* $\mathsf{Var}(\Gamma, A)$ *has decidable equality.*

Proof. This follows from the characterization of variables in proposition 3.1.15. □

Lemma 3.1.18. *Let* $(\Gamma \vdash x, y : A) \in \mathbf{Ren}(\mathcal{C})$ *be two variables. Then we can construct an object* $\Gamma.(x = y)$ *representing the presheaf* $(\gamma : \mathbb{k}(\Gamma)) \times (x(\gamma) = y(\gamma))$.

Proof. By proposition 3.1.17, we can decide the equality of x and y. If $x = y$, then Γ is already a representing object.

Otherwise, x and y are different variables from the context Γ. Without loss of generality, assume that x occurs before y in Γ. We can write $\Gamma = (\gamma_1 : \Gamma_1).(x : A(\gamma_1)).(\gamma_2 : \Gamma_2(\gamma_1, x)).(y : A(\gamma_1)).\Gamma_3(\gamma_1, x, \gamma_2, y)$. Then $\Gamma.(x = y) \triangleq (\gamma_1 : \Gamma_1).(x : A(\gamma_1)).(\gamma_2 : \Gamma_2(\gamma_1, x)).\Gamma_3(\gamma_1, x, \gamma_2, x)$ represents the presheaf $(\gamma : \yoneda(\Gamma)) \times (x(\gamma) = y(\gamma))$. □

Proposition 3.1.19. *The underlying category of* $\mathbf{Ren}(\mathcal{C})$ *has all finite limits.*

Proof. We check that $\mathbf{Ren}(\mathcal{C})$ has a terminal object, binary products and equalizers.

Terminal object By its definition as a renaming algebra, $\mathbf{Ren}(\mathcal{C})$ has a terminal object.

Binary products This follows from the contextuality of $\mathbf{Ren}(\mathcal{C})$.

Equalizers The equalizer of $f, g : \Gamma \to \Delta$ can be constructed by induction on the structure of Δ, using lemma 3.1.18 in the recursive case. □

3.2 Type structures

We recall the type structures most commonly found on CwFs: Σ-types, equality types and Π-types. These will be needed to define the functorial semantics of GATs and SOGATs. We will define other type structures (universes, natural number types, W-types, etc.) through SOGATs.

Even without SOGATs, it is already convenient to define these structures using the internal language of $\mathbf{Psh}(\mathcal{C})$: this means that we define type structures for a family $(\mathsf{ty} : \mathsf{Set}^{\mathsf{c}}, \mathsf{tm} : \mathsf{ty} \to \mathsf{Set}_{\mathsf{rep}})$, and use the correspondence between $\mathbf{Psh}(\mathcal{C})$ and its internal language to obtain the corresponding external type structure.

3.2.1 Dependent sums

Definition 3.2.1. A family $(\mathsf{ty}, \mathsf{tm})$ has **dependent sums**, or **dependent binary product types**, or **Σ-types**, if it is equipped with operations

$$\begin{aligned}&\Sigma : (A : \mathsf{ty})(B : \mathsf{tm}(A) \to \mathsf{ty}) \to \mathsf{ty},\\ &\mathsf{pair} : (a : \mathsf{tm}(A)) \times \mathsf{tm}(B(a)) \cong \mathsf{tm}(\Sigma(A, B)).\end{aligned}$$

A family has a **1-type**, or a **nullary product type**, if it is equipped with operations

$$\begin{aligned}&\mathbf{1} : \mathsf{ty},\\ &\mathsf{tt} : \{\star\} \cong \mathsf{tm}(\mathbf{1}).\end{aligned}$$ ⌟

Definition 3.2.2. A CwF has **Σ-types** if for every $\Gamma \in \mathcal{C}$, we have operations

$$\begin{aligned}&\Sigma : (A : \mathsf{Ty}(\Gamma))(B : \mathsf{Ty}(\Gamma.A)) \to \mathsf{Ty}(\Gamma),\\ &\mathsf{pair} : (a : \mathsf{Tm}(\Gamma, A)) \times \mathsf{Tm}(\Gamma, B[a]) \cong \mathsf{Tm}(\Gamma, \Sigma(A, B)),\\ &\mathbf{1} : \mathsf{Ty}(\Gamma),\\ &\mathsf{tt} : \{\star\} \cong \mathsf{Tm}(\Gamma, \mathbf{1}),\end{aligned}$$

that are natural in Γ: for every $(f : \Delta \to \Gamma) \in \mathcal{C}$, we have

$$\begin{aligned}
&\Sigma(A, B)[f] = \Sigma(A[f], B[f^+]),\\
&\mathsf{pair}(a, b)[f] = \mathsf{pair}(a[f], b[f]),\\
&\mathbf{1}[f] = \mathbf{1},\\
&\mathsf{tt}[f] = \mathsf{tt}.
\end{aligned}$$ ⌟

When assuming that a CwF has Σ-types, we always implicitly assume that it is also equipped with a $\mathbf{1}$-type.

3.2.2 Equality types

Definition 3.2.3. A family $(\mathsf{ty}, \mathsf{tm})$ has **extensional identity types**, or **equality types** if it is equipped with operations

$$\begin{aligned}
&\mathsf{Eq} : (A : \mathsf{ty})(x, y : \mathsf{tm}(A)) \to \mathsf{ty},\\
&\mathsf{refl} : \mathsf{tm}(\mathsf{Eq}(A, x, x)),\\
&\mathsf{reflect} : (p : \mathsf{tm}(\mathsf{Eq}(A, x, y))) \to (x = y).
\end{aligned}$$ ⌟

Definition 3.2.4. A CwF $\mathcal{C}$ has **extensional identity types**, or **equality types**, if for every object $\Gamma \in \mathcal{C}$, we have operations

$$\begin{aligned}
&\mathsf{Eq} : (A : \mathsf{Ty}(\Gamma))(x, y : \mathsf{Tm}(\Gamma, A)) \to \mathsf{Ty}(\Gamma),\\
&\mathsf{refl} : (A : \mathsf{Ty}(\Gamma))(x : \mathsf{Tm}(\Gamma, A)) \to \mathsf{Tm}(\Gamma, \mathsf{Eq}(A, x, x)),\\
&\mathsf{reflect} : (p : \mathsf{Tm}(\Gamma, \mathsf{Eq}(A, x, y))) \to (x = y),
\end{aligned}$$

that are natural in Γ, meaning that given any morphism $(f : \Delta \to \Gamma) \in \mathcal{C}$, we have

$$\begin{aligned}
&\mathsf{Eq}(A, x, y)[f] = \mathsf{Eq}(A[f], x[f], y[f]),\\
&\mathsf{refl}(A, x)[f] = \mathsf{refl}(A[f], x[f]).
\end{aligned}$$ ⌟

3.2.3 Dependent products

Definition 3.2.5. A family $(\mathsf{ty}, \mathsf{tm})$ has **dependent products**, or **dependent function types**, or Π**-types**, if it is equipped with operations

$$\begin{aligned}
&\Pi : (A : \mathsf{ty})(B : \mathsf{tm}(A) \to \mathsf{ty}) \to \mathsf{ty},\\
&\mathsf{lam} : ((a : \mathsf{tm}(A)) \to \mathsf{tm}(B(a)) \cong \mathsf{tm}(\Pi(A, B)) : \mathsf{app}.
\end{aligned}$$ ⌟

Definition 3.2.6. A family $(\mathsf{ty}, \mathsf{tm})$ has **dependent products with arities in a family** $(\mathsf{ty}', \mathsf{tm}')$ if it is equipped with operations

$$\begin{aligned}
&\Pi' : (A : \mathsf{ty}')(B : \mathsf{tm}'(A) \to \mathsf{ty}) \to \mathsf{ty},\\
&\mathsf{lam} : ((a : \mathsf{tm}'(A)) \to \mathsf{tm}(B(a)) \cong \mathsf{tm}(\Pi'(A, B)) : \mathsf{app}.
\end{aligned}$$ ⌟

3.2.4 Correspondences with classes of categories

CwFs with some selection of type formers among Σ-types, Π-types and equality types correspond to important classes of categories: type theories with specific type formers are the internal language of some classes of categories.

There are multiple ways to state these correspondences. At the very least, we want a way to interpret type theory into the categorical structures; i.e. we want to construct models of type theory from these categories. Constructing these models requires solving coherence problems (Curien 1993; Hofmann 1994; Lumsdaine and Warren 2015). Going further, one may ask for some kind of equivalence between models of type theory and the categories. Clairambault and Dybjer (2014) construct a biequivalence between bicategory of CwFs with Σ, Π and Eq and the bicategory of locally cartesian closed categories.

- CwFs with Σ-types and equality types correspond to categories with all finite limits (finitely complete categories). Indeed, all finite products can be written using non-dependent Σ-types, and an equalizer can be written using both Σ- and equality types:
$$(a : A) \times (f(a) = g(a)).$$
- CwFs with Σ-types, equality types and Π-types correspond to locally cartesian closed categories.
- CwFs with just Σ-types correspond to clans: categories with a class of maps that is closed under finite compositions (the nullary case corresponds to $\mathbf{1}$-types and the binary case corresponds to Σ-types), and under pullbacks against arbitrary maps (this corresponds to substitution).
- CwFs without any additional structure correspond to categories with a class of maps closed under pullbacks against arbitrary maps.

3.3 CwFs with first-order dependent products

3.3.1 Definition

We now introduce $(\Sigma, \Pi_{\mathsf{rep}})$-CwFs, or CwFs with first-order dependent products, or categories with families and representable families.

They ought to correspond to categories with representable maps, analogously to the correspondences of section 3.2.4.

In Σ-CwFs, the types are zeroth-order types: there are no function types. In (Σ, Π)-CwFs, we have all higher-order types. The purpose of $(\Sigma, \Pi_{\mathsf{rep}})$-CwFs, is to have categorical semantics for a type theory with first-order types (hence second-order quantification when working over arbitrary contexts, which can include first-order types). For this purpose, we stratify the types into zeroth-order types (called representable types) and the first-order types (just called types).

Definition 3.3.1. A $(\Sigma, \Pi_{\mathsf{rep}})$**-CwF** is a CwF $\mathcal{C}$ together with:

- A second presheaf $\mathsf{Ty}_{\mathsf{rep}}$ with a natural transformation $\mathsf{Ty}_{\mathsf{rep}} \to \mathsf{Ty}$ (left implicit). The elements of $\mathsf{Ty}_{\mathsf{rep}}$ are called **representable types**.
- Both $(\mathcal{C}, \mathsf{Ty})$ and $(\mathcal{C}, \mathsf{Ty}_{\mathsf{rep}})$ have Σ- and $\mathbf{1}$- types (which are not necessarily preserved by the map $\mathsf{Ty}_{\mathsf{rep}} \to \mathsf{Ty}$).
- $(\mathcal{C}, \mathsf{Ty})$ has Π-types with arities in $\mathsf{Ty}_{\mathsf{rep}}$. ⌟

The category of $(\Sigma, \Pi_{\mathsf{rep}})$-CwFs is denoted by $\mathbf{CwF}_{\Sigma,\Pi_{\mathsf{rep}}}$.
There is a fully faithful functor

$$\mathbf{CwF}_{\Sigma} \to \mathbf{CwF}_{\Sigma,\Pi_{\mathsf{rep}}}$$

that equips a Σ-CwF with an empty presheaf of representable types. There is also a fully faithful functor

$$\mathbf{CwF}_{\Sigma,\Pi} \to \mathbf{CwF}_{\Sigma,\Pi_{\mathsf{rep}}}$$

that sends a (Σ, Π)-CwF to a $(\Sigma, \Pi_{\mathsf{rep}})$-CwF whose types are all representable (the natural transformation $\mathsf{Ty}_{\mathsf{rep}} \to \mathsf{Ty}$ is the identity).

Both of these functors admit left adjoints. As a consequence of these two functors being fully faithful, instead of studying Σ-, $(\Sigma, \Pi_{\mathsf{rep}})$- and (Σ, Π)- CwFs separately, we can just study $(\Sigma, \Pi_{\mathsf{rep}})$-CwFs and transfer some definitions and results to Σ- and (Σ, Π)-CwFs.

3.3.2 Pseudo-morphisms

The category $\mathbf{CwF}_{\Sigma,\Pi_{\mathsf{rep}}}$ is a 1-category, but categories with representable maps are usually seen as objects of a 2-category. Perhaps more familiar, finitely complete categories are the objects of a 2-category **Lex**.

We will need a way to describe the 2-categorical structure at the level of $\mathbf{CwF}_{\Sigma,\Pi_{\mathsf{rep}}}$. We will do so using 1-categorical tools: equipping $(\Sigma, \Pi_{\mathsf{rep}})$-CwFs with arrow objects $\mathbf{Arr}(-)$ (extending categories of arrows), so that 2-cells can be represented by strict morphisms $\mathcal{C} \to \mathbf{Arr}(\mathcal{D})$.

For now, we define a $(2,1)$-category of $(\Sigma, \Pi_{\mathsf{rep}})$-CwFs, pseudo-morphisms and invertible 2-cells.

Definition 3.3.2. Let $\mathcal{C}, \mathcal{D}$ be $(\Sigma, \Pi_{\mathsf{rep}})$-CwFs. A **pseudo-morphism** $F : \mathcal{C} \to_{\mathsf{ps}} \mathcal{D}$ consists of a functor $F : \mathcal{C} \to \mathcal{D}$ with actions on representable types, types and terms such that:

- The actions on representable types and types commute with the natural transformations $\mathcal{C}.\mathsf{Ty}_{\mathsf{rep}} \to \mathcal{C}.\mathsf{Ty}$ and $\mathcal{D}.\mathsf{Ty}_{\mathsf{rep}} \to \mathcal{D}.\mathsf{Ty}$.
- The terminal object is preserved up to isomorphism: for any $\Delta \in \mathcal{C}$, the canonical map
$$\mathcal{D}(\Delta, F(1_{\mathcal{C}})) \to \mathcal{D}(\Delta, 1_{\mathcal{D}})$$
is an isomorphism.
- The context extensions are preserved up to isomorphism: for any $\Delta \in \mathcal{C}$, the canonical map
$$\langle F(\boldsymbol{p}_A), F(\boldsymbol{q}_A)\rangle : \mathcal{D}(\Delta, F(\Gamma.A)) \to \mathcal{D}(\Delta, F(\Gamma).F(A))$$
is an isomorphism.
- The $\mathbf{1}$-type (and the representable $\mathbf{1}$-type) is preserved up to isomorphism: for any $\Delta \in \mathcal{C}$, the canonical map
$$\mathsf{tt} : \mathcal{D}.\mathsf{Tm}(\Delta, F(\mathbf{1})) \to \mathcal{D}.\mathsf{Tm}(\Delta, \mathbf{1})$$
is an isomorphism.

- The Σ-types (and the representable Σ-types) are preserved up to isomorphism: over any $\gamma : \mathcal{D}(\Delta, F(\Gamma))$, the canonical map

$$\langle F(\mathsf{fst}), F(\mathsf{snd})\rangle : \mathcal{D}.\mathsf{Tm}(\Delta, F(\Sigma(X,Y))[\gamma]) \to \mathcal{D}.\mathsf{Tm}(\Delta, \Sigma(F(X)[\gamma], F(Y)[\gamma^+])).$$

  is an isomorphism.

- The Π-types are preserved up to isomorphism: over any $\gamma : \mathcal{D}(\Delta, F(\Gamma))$, the canonical map

$$F(\mathsf{app}) : \mathcal{D}.\mathsf{Tm}(\Delta, F(\Pi(X,Y))[\gamma]) \to \mathcal{D}.\mathsf{Tm}(\Delta, \Pi(F(X)[\gamma], F(Y)[\gamma^+])).$$

  is an isomorphism. ⌟

We obtain notions of pseudo-morphisms of Σ- and (Σ,Π)- CwFs as special cases of the above definition.

Proposition 3.3.3. *The preservation of the* **1***- and* Σ*- types is redundant in the definition of pseudo-morphism.*

Proof. The preservation of Σ-types can be reduced to the preservation of context extension, thanks to the isomorphisms $\Gamma.\Sigma(A,B) \cong (\Gamma.A).B$. The case of the **1**-type follows from the isomorphisms $\Gamma.\mathbf{1} \cong \Gamma$. □

Proposition 3.3.4. *Let* $\mathcal{C}$ *be a* $(\Sigma, \Pi_{\mathsf{rep}})$*-CwF. Then the Yoneda embedding* $よ : \mathcal{C} \to \widehat{\mathcal{C}}$ *has the structure of a pseudo-morphism of* $(\Sigma, \Pi_{\mathsf{rep}})$*-CwFs. Furthermore, its action on terms is bijective.*

Proof. We use the modal internal language of $\mathbf{Psh}(\mathcal{C})$, and identify the (Σ,Π)-structure of $\widehat{\mathcal{C}}$ with the (Σ,Π)-CwF structure of $\Box\mathbf{Set}^{\mathfrak{c}}$.

We first define the action of $よ$ on types. Let $(\Gamma \vdash A\ \mathsf{type}) \in \mathcal{C}$ be a type. We can see it as an element $A :: \Box(よ(\Gamma) \to \mathcal{C}.\mathsf{Ty})$. We pose $よ(A) \triangleq \lambda\gamma \mapsto \mathcal{C}.\mathsf{Tm}(A(\gamma))$. Functoriality follows from the associativity of composition. The action on representable types is the same as the action on types. The action on terms is then the identity. In particular, it is bijective.

The preservation of the empty and extended contexts follows from the fact that $よ$ preserves limits that exist in the base category; in this case we use the preservation of the terminal object and of pullbacks.

By proposition 3.3.3, we do not need to check the preservation for **1**- and Σ- types.

In the case of Π-types, we have a natural transformation $\gamma :: \Box(X \Rightarrow よ(\Gamma))$, elements $A :: \Box(よ(\Gamma) \to \mathcal{C}.\mathsf{Ty}_{\mathsf{rep}})$ and $B :: \Box(よ(\Gamma) \to \mathcal{C}.\mathsf{Ty})$, and we need to prove that the canonical map from

$$\Box((x : X) \to \mathcal{C}.\mathsf{Tm}(\Pi(A(\gamma(x)), \lambda a \mapsto B(\gamma(x)))))$$

to

$$\Box((x : X) \to ((a : \mathcal{C}.\mathsf{Tm}(A(\gamma(x)))) \to \mathcal{C}.\mathsf{Tm}(B(\gamma(x), a))))$$

is bijective, which follows directly from the universal property of Π-types. □

Definition 3.3.5. Let $\mathcal{C}$ be a CwF and $(\gamma : \Gamma \vdash A(\gamma)\ \mathsf{type}) \in \mathcal{C}$, $(\gamma : \Gamma \vdash B(\gamma)\ \mathsf{type}) \in \mathcal{C}$ be two types. A **type isomorphism** $\alpha : A \cong B$ consists of terms $(\gamma : \Gamma, a : A(\gamma) \vdash f(\gamma, a) : B(\gamma)) \in \mathcal{C}$ and $(\gamma : \Gamma, b : B(\gamma) \vdash g(\gamma, b) : A(\gamma)) \in \mathcal{C}$ such that $(\gamma : \Gamma, a : A(\gamma) \vdash g(\gamma, f(\gamma, a)) = a) \in \mathcal{C}$ and $(\gamma : \Gamma, b : B(\gamma) \vdash f(\gamma, g(\gamma, b)) = b) \in \mathcal{C}$. ⌟

Definition 3.3.6. Let $F, G : \mathcal{C} \to_{\mathsf{ps}} \mathcal{D}$ be two parallel pseudo-morphisms of $(\Sigma, \Pi_{\mathsf{rep}})$-CwFs. An **invertible 2-cell** $\alpha : F \cong G$ consists of a natural isomorphism $\alpha : F \cong G$ along with:

- For every type $(\Gamma \vdash A\ \mathsf{type}) \in \mathcal{C}$, a type isomorphism $\beta_A : F(A) \cong G(A)[\alpha_\Gamma]$, such that for any $f : \Delta \to \Gamma$, we have $\beta_{A[f]} = \beta_A[F(f)]$, where

$$\beta_{A[f]} : F(A[f]) \cong G(A[f])[\alpha_\Delta],$$

  which simplifies to $\beta_{A[f]} : F(A)[F(f)] \cong G(A)[\alpha_\Gamma][F(f)]$.

- For every term $(\Gamma \vdash a : A) \in \mathcal{C}$, an equality $\beta_A(F(a)) = G(a)[\alpha_\Gamma]$. ⌟

Lemma 3.3.7. *There is a unique invertible 2-cell extending any natural isomorphism $\alpha : F \cong G$ between the underlying functors of pseudo-morphisms.*

Proof. We first prove uniqueness.

The type isomorphisms need to satisfy the following equality

$$(\gamma a : F(\Gamma.A) \vdash \beta_{A[\boldsymbol{p}_A]}(\gamma a, F(\boldsymbol{q}_A)(\gamma a)) = G(\boldsymbol{q}_A)[\alpha_{\Gamma.A}](\gamma a)).$$

By naturality, this simplifies to

$$(\gamma a : F(\Gamma.A) \vdash \beta_A[F(\boldsymbol{p}_A)](\gamma a, F(\boldsymbol{q}_A)(\gamma a)) = G(\boldsymbol{q}_A)[\alpha_{\Gamma.A}](\gamma a)),$$

i.e.

$$(\gamma a : F(\Gamma.A) \vdash \beta_A(F(\boldsymbol{p}_A)(\gamma a), F(\boldsymbol{q}_A)(\gamma a)) = G(\boldsymbol{q}_A)[\alpha_{\Gamma.A}](\gamma a)).$$

Because F preserves context extensions, this uniquely characterizes β_A.

For existence, the above provides a candidate definition of the type isomorphisms β_A, which can be shown to satisfy the desired equalities. □

Construction 3.3.8. Let $F : \mathcal{C} \to_{\mathsf{ps}} \mathcal{D}$ be a pseudo-morphism of $(\Sigma, \Pi_{\mathsf{rep}})$-CwFs. We define a $(\Sigma, \Pi_{\mathsf{rep}})$-CwF $\mathsf{Glue}_{\cong}(F)$, called the **iso-gluing** of F.

- An object of $\mathsf{Glue}_{\cong}(F)$ is a triple $(\Gamma_1, \Gamma_2, \alpha)$ where $\Gamma_1 \in \mathcal{C}$, $\Gamma_2 \in \mathcal{D}$ and $\alpha : F(\Gamma_1) \cong \Gamma_2$.

- A morphism from $(\Delta_1, \Delta_2, \beta)$ to $(\Gamma_1, \Gamma_2, \alpha)$ consists of morphisms $f_1 \in \mathcal{C}(\Delta_1, \Gamma_1)$ and $f_2 \in \mathcal{D}(\Delta_2, \Gamma_2)$ such that the following square commutes:

$$\begin{array}{ccc} F(\Delta_1) & \xrightarrow{\beta} & \Delta_2 \\ {\scriptstyle F(f_1)}\downarrow & & \downarrow{\scriptstyle f_2} \\ F(\Gamma_1) & \xrightarrow{\alpha} & \Gamma_2. \end{array}$$

- A type over $(\Gamma_1, \Gamma_2, \alpha)$ is a triple (A_1, A_2, β) consisting of types $(\Gamma_1 \vdash A_1\ \mathsf{type}) \in \mathcal{C}$, $(\Gamma_2 \vdash A_2\ \mathsf{type}) \in \mathcal{D}$ and of a type isomorphism $\beta : F(A_1) \cong A_2[\alpha]$.

- The representable types are defined in the same way as types.

- A term of type (A_1, A_2, β) is a term of type A_1 in $\mathcal{C}$. Note that there is then a unique term a_2 of type A_2 such that $\beta(F(a_1)) = a_2[\alpha]$.

- The extension of a context $(\Gamma_1, \Gamma_2, \alpha)$ by a type (A_1, A_2, β) is a context $(\Gamma_1.A_1, \Gamma_2.A_2, \alpha.\beta)$ where $(\alpha.\beta)(\gamma, a) \triangleq \langle \alpha(\gamma), \beta(\gamma, a)\rangle$.
- The **1**-, Σ- and Π- types are defined using the fact that they preserve type isomorphisms. We omit the details. ⌟

Proposition 3.3.9. *The projection map $\pi_1 : \mathsf{Glue}_{\cong}(F) \to \mathcal{C}$ is a strict $(\Sigma, \Pi_{\mathsf{rep}})$-CwF morphism and satisfies the right lifting property with respect to $\{I^{\mathsf{ty}}, I^{\mathsf{ty}_{\mathsf{rep}}}, I^{\mathsf{tm}}, E^{\mathsf{tm}}\}$.*

Proof. By definition of $\mathsf{Glue}_{\cong}(F)$, the action of π_1 on terms is bijective, which implies that it satisfies the right lifting property with respect to $\{I^{\mathsf{tm}}, E^{\mathsf{tm}}\}$.

For the right lifting property with respect to I^{ty}, take an object $(\Gamma_1, \Gamma_2, \alpha) \in \mathsf{Glue}_{\cong}(F)$ and a type $(\Gamma_1 \vdash A_1\ \mathsf{type}) \in \mathcal{C}$. We need to find A_2 and β so as to form a type (A_1, A_2, β) of $\mathsf{Glue}_{\cong}(F)$. We can simply pose $A_2 \triangleq F(A_1)[\alpha^{-1}]$ and let β be the identity isomorphism.

The case of $I^{\mathsf{ty}_{\mathsf{rep}}}$ is analogous. □

Proposition 3.3.10. *Let $F : \mathcal{C} \to_{\mathsf{ps}} \mathcal{D}$ be a pseudo-morphism of $(\Sigma, \Pi_{\mathsf{rep}})$-CwFs. If $\mathcal{C}$ is cofibrant with respect to $\{I^{\mathsf{ty}}, I^{\mathsf{ty}_{\mathsf{rep}}}, I^{\mathsf{tm}}, E^{\mathsf{tm}}\}$, then there merely exists a strict $(\Sigma, \Pi_{\mathsf{rep}})$-CwF morphism $F' : \mathcal{C} \to \mathcal{D}$ along with an isomorphism $\alpha : F \cong F'$.*

Proof. By proposition 3.3.9, there exists a section of $\pi_1 : \mathsf{Glue}_{\cong}(F) \to \mathcal{C}$. This section can be decomposed into $F' : \mathcal{C} \to \mathcal{D}$ and $\alpha : F \cong F'$. □

For applications, the mere existence of a strict morphism is often not enough.

We also prove some results involving the second projection map.

Proposition 3.3.11. *The projection map $\pi_2 : \mathsf{Glue}_{\cong}(F) \to \mathcal{D}$ is a strict morphism of $(\Sigma, \Pi_{\mathsf{rep}})$-CwFs.*

Proof. Straightforward. □

Proposition 3.3.12. *If $F : \mathcal{C} \to \mathcal{D}$ is essentially surjective on objects, types and representable types, and bijective on terms, then the projection map $\pi_2 : \mathsf{Glue}_{\cong}(F) \to \mathcal{D}$ satisfies the left lifting property with respect to $\{I^{\mathsf{ty}}, I^{\mathsf{ty}_{\mathsf{rep}}}, I^{\mathsf{tm}}, E^{\mathsf{tm}}\}$.*

Proof. The left lifting property with respect to I^{ty} corresponds to F being essentially surjective on types. The left lifting property with respect to $I^{\mathsf{ty}_{\mathsf{rep}}}$ corresponds to F being essentially surjective on representable types. The left lifting property with respect to I^{tm} and E^{tm} corresponds to F being bijective on terms.

In each case we also need to use the fact that F is essentially surjective on objects. □

In other words, when F is an equivalence in the 2-category of $(\Sigma, \Pi_{\mathsf{rep}})$-CwFs, pseudo-morphisms and natural transformations, then the iso-gluing provides a presentation of F as a span of trivial fibrations. This generalizes an analogous result for equivalences of categories.

3.3.3 CwFs with base types

We introduce a notion of CwFs with a chosen non-functorial family of base types (and base representable types). We want to work with CwFs that are presented by a collection of generating types, generating terms and equations between terms. Because they don't include any equations between types, their types should be freely generated by the generating types and the type formers. This is similar to the objects of a contextual CwF, which are freely generated by the empty context and context extensions. However a choice of base types has to be included as additional structure.

Definition 3.3.13. A $(\Sigma, \Pi_{\mathsf{rep}})$-CwF with **base types** is a $(\Sigma, \Pi_{\mathsf{rep}})$-CwF $\mathcal{C}$ along with the data of families $\mathcal{C}.\mathsf{BTy} : \mathcal{C}.\mathsf{Ob} \to \mathsf{Set}$ and $\mathcal{C}.\mathsf{BTy}_{\mathsf{rep}} : \mathcal{C}.\mathsf{Ob} \to \mathsf{Set}$ (note the absence of a functorial action: base types are not stable under substitution) together with maps $\mathcal{C}.\mathsf{BTy}(\Gamma) \to \mathcal{C}.\mathsf{Ty}(\Gamma)$ and $\mathcal{C}.\mathsf{BTy}_{\mathsf{rep}}(\Gamma) \to \mathcal{C}.\mathsf{Ty}_{\mathsf{rep}}(\Gamma)$. ⌟

Elements of $\mathcal{C}.\mathsf{BTy}(-)$ are called base non-representable types, and elements of $\mathcal{C}.\mathsf{BTy}_{\mathsf{rep}}(-)$ are called base representable types. Elements of the disjoint union of $\mathcal{C}.\mathsf{BTy}$ and $\mathcal{C}.\mathsf{BTy}_{\mathsf{rep}}$ are called base types.

There are maps

$$I^{\mathsf{bty}} : \mathsf{Free}(\mathbf{\Gamma} \vdash) \to \mathsf{Free}(\mathbf{\Gamma} \vdash \boldsymbol{A}\ \mathsf{type}_{\mathsf{base}}),$$
$$I^{\mathsf{bty}_{\mathsf{rep}}} : \mathsf{Free}(\mathbf{\Gamma} \vdash) \to \mathsf{Free}(\mathbf{\Gamma} \vdash \boldsymbol{A}\ \mathsf{type}_{\mathsf{base,rep}})$$

that are the generic extension of a $(\Sigma, \Pi_{\mathsf{rep}})$-CwF with base types by a new base type or a new base representable type.

We write $\mathbf{CwF}^{\mathsf{base}}_{\Sigma,\Pi_{\mathsf{rep}}}$ for the 1-category of $(\Sigma, \Pi_{\mathsf{rep}})$-CwFs with base types. The forgetful functor $\mathbf{CwF}^{\mathsf{base}}_{\Sigma,\Pi_{\mathsf{rep}}} \to \mathbf{CwF}_{\Sigma,\Pi_{\mathsf{rep}}}$ has both a left adjoint and a right adjoint. The image of the left adjoint consists of the $(\Sigma, \Pi_{\mathsf{rep}})$-CwF without any base types, while the image of the right adjoint consists of the $(\Sigma, \Pi_{\mathsf{rep}})$-CwFs in which all types are base types (i.e. such that the maps $\mathcal{C}.\mathsf{BTy}(\Gamma) \to \mathcal{C}.\mathsf{Ty}(\Gamma)$ and $\mathcal{C}.\mathsf{BTy}_{\mathsf{rep}}(\Gamma) \to \mathcal{C}.\mathsf{Ty}_{\mathsf{rep}}(\Gamma)$ are bijective).

Definition 3.3.14. Let $\mathcal{C}$ be a $(\Sigma, \Pi_{\mathsf{rep}})$-CwF with base types.

- A **basic non-representable type** is a type of the form $(\Gamma \vdash A[\sigma]\ \mathsf{type}) \in \mathcal{C}$ for a base non-representable type $(\partial A \vdash A\ \mathsf{type}_{\mathsf{base}}) \in \mathcal{C}$ and $(\Gamma \vdash \sigma : \partial A) \in \mathcal{C}$.
- A **basic representable type** is a type of the form $(\Gamma \vdash A[\sigma]\ \mathsf{type}) \in \mathcal{C}$ for a base representable type $(\partial A \vdash A\ \mathsf{type}_{\mathsf{base,rep}}) \in \mathcal{C}$ and $(\Gamma \vdash \sigma : \partial A) \in \mathcal{C}$.
- The **canonical representable types** are inductively generated by the basic representable types and the type formers $\mathbf{1}$ and Σ.
- The **canonical types** are inductively generated by the basic non-representable types, the canonical representable types and the type formers $\mathbf{1}$, Σ and Π_{rep}. ⌟

Definition 3.3.15. A $(\Sigma, \Pi_{\mathsf{rep}})$-CwF $\mathcal{C}$ with base types is said to be **type-presented** if it is a left object for the orthogonal factorization system generated by $\{I^{\mathsf{bty}}, I^{\mathsf{bty}_{\mathsf{rep}}}, I^{\mathsf{tm}}\}$. ⌟

Proposition 3.3.16. *The type-presented replacement* $\mathsf{typres}(\mathcal{C})$ *of a* $(\Sigma, \Pi_{\mathsf{rep}})$*-CwF with base types* $\mathcal{C}$ *admits the following explicit description:*

- *an object of* $\mathsf{typres}(\mathcal{C})$ *is a closed telescope of canonical types.*
- *the morphism* $\mathsf{typres}(\mathcal{C}) \to \mathcal{C}$ *has bijective actions on substitution, terms and base types.*
- *a representable type of* $\mathsf{typres}(\mathcal{C})$ *is a canonical representable type.*
- *a type of* $\mathsf{typres}(\mathcal{C})$ *is a canonical type.*
- *the type operations are the constructors of the inductive family defining the canonical representable types and canonical types.*
- *the rest of the structure is inherited from* $\mathcal{C}$*.*

Proof. Write $\mathcal{C}_{\mathsf{tp}}$ for candidate type-presented replacement from the statement. By definition, the map $\mathcal{C}_{\mathsf{tp}} \to \mathcal{C}$ is a right type-presented map. Thus it suffices to prove that $\mathcal{C}_{\mathsf{tp}}$ is type-presented.

Let $F : \mathcal{D} \to \mathcal{E}$ be a right type-presented map and $G : \mathcal{C}_{\mathsf{tp}} \to \mathcal{E}$ be any morphism of $(\Sigma, \Pi_{\mathsf{rep}})$-CwFs with base types. We write F^{-1} for the inverses of the actions of F on base types and on terms. Any factorization $H : \mathcal{C}_{\mathsf{tp}} \to \mathcal{D}$ of G through F must satisfy the equations:

$$
\begin{aligned}
&H(\varepsilon) = 1_{\mathcal{D}},\\
&H(\Delta.A) = H(\Delta).H(A),\\
&H(\langle\rangle) = \langle\rangle,\\
&H(\langle f, a\rangle) = \langle H(f), H(a)\rangle,\\
&H(A) = F^{-1}(G(A)), && \text{(for a base type } A\text{)}\\
&H(a) = F^{-1}(G(a)),\\
&H(A[\sigma]) = H(A)[H(\sigma)], && \text{(for a basic type } A[\sigma]\text{)}\\
&H(\Pi(A, B)) = \Pi(H(A), H(B)),\\
&H(\Sigma(A, B)) = \Sigma(H(A), H(B)),\\
&H(\mathbf{1}) = \mathbf{1},\\
&H(\mathbf{1}_{\mathsf{rep}}) = \mathbf{1}_{\mathsf{rep}},\\
&H(\Sigma_{\mathsf{rep}}(A, B)) = \Sigma_{\mathsf{rep}}(H(A), H(B)),
\end{aligned}
$$

By induction over telescopes, canonical representable types and canonical types, we can construct $H : \mathcal{C}_{\mathsf{tp}} \to \mathcal{D}$ satisfying these equations. Thus $\mathcal{C}_{\mathsf{tp}}$ is type-presented, as needed. □

Using type-presented $(\Sigma, \Pi_{\mathsf{rep}})$-CwFs, we can prove a strengthening of proposition 3.3.10.

Proposition 3.3.17. *Let $\mathcal{C}$ be a type-presented $(\Sigma, \Pi_{\mathsf{rep}})$-CwF and $F : \mathcal{C} \to_{\mathsf{ps}} \mathcal{D}$ be a pseudomorphism of $(\Sigma, \Pi_{\mathsf{rep}})$-CwFs. Then there exists a unique strict morphism $F' : \mathcal{C} \to \mathcal{D}$ along with an isomorphism $\alpha : F \cong F'$ such that for every base type $(\partial A \vdash A\ \mathsf{type}) \in \mathcal{C}$, $F'(A) = F(A)[\alpha_{\partial A}]$ and $\alpha(A) = \mathsf{id}$.*

Proof. We equip $\mathsf{Glue}_{\cong}(F)$ (which was constructed in Construction 3.3.8) with a choice of base types: a base type over $(\Gamma_1, \Gamma_2, \alpha)$ is a type $A_1 : \mathcal{C}.\mathsf{Ty}(\Gamma_1)$. The corresponding type over $(\Gamma_1, \Gamma_2, \alpha)$ is the triple $(A_1, F(A_1)[\alpha^{-1}], \mathsf{id})$.

Now the projection $\pi : \mathsf{Glue}_{\cong}(F) \to \mathcal{C}$ has bijective actions on base types and on terms. Since $\mathcal{C}$ is type-presented, it admits a unique section, which can be decomposed into $F' : \mathcal{C} \to \mathcal{D}$ along with an isomorphism $\alpha : F \cong F'$. The preservation of base types says that for every base type $(\partial A \vdash A\ \mathsf{type}) \in \mathcal{C}$, $F'(A) = F(A)[\alpha_{\partial A}]$ and $\alpha(A) = \mathsf{id}$. □

Proposition 3.3.17 is more convenient than proposition 3.3.10, thanks to the unicity of the strict replacement.

Chapter 4

First-order generalized algebraic theories

In this chapter, we study *generalized algebraic theories*, abbreviated as GATs, and their functorial semantics in Σ-CwFs. GATs, which were introduced by Cartmell (1986), generalize (multisorted) algebraic theories by allowing for dependent sorts. Examples of dependent sorts are morphisms in a category, which depend over pairs of objects, or terms in a CwF, which depend on a context and a type.

Generalized algebraic theories have the same expressive power as *essentially algebraic theories* (EATs), which are an alternative generalization of algebraic theory allowing for partial operations. In an EAT of categories, the composition of morphisms is seen as a partial operation on pairs of arbitrary morphisms, only defined when the target of the first morphism matches with the source of the second morphism. GATs and EATs having the same "expressive power" means here that a category is the category of algebras of a GAT if and only if it is the category of algebras of an EAT. Gabriel-Ulmer duality says that we can recognize these categories of algebras as the locally finitely presentable categories.

However GATs and EATs are not interchangeable: a GAT equips its category of algebras with additional structure: a (cofibrations, trivial fibrations) weak factorization system induced by the sort dependency. This is important when considering homotopical structures. See for example the work of Frey (2023), on the extension of Gabriel-Ulmer duality to GATs, and the work of Henry (2020) on the languages of model categories.

There are at least two approaches to the semantics of algebraic theories and their generalizations:

Functorial semantics A theory is an object $\mathcal{T}$ in a 2-category $\mathbf{E}$ (a doctrine). The different semantic notions are obtained by evaluating $\mathbf{E}(\mathcal{T}, -)$ at various objects of $\mathbf{E}$. Typically, $\mathbf{E}$ is a 2-category of structured categories, structure preserving functors and natural transformations, such as cartesian categories (in the case of multisorted algebraic theories), finitely complete categories (in the case of essentially algebraic theories), or, for our purposes, structured CwFs. When the category $\mathbf{Set}$ is an object of $\mathbf{E}$, the category of algebras is defined as $\mathbf{E}(\mathcal{T}, \mathbf{Set})$, which is the category of structure preserving functors from $\mathcal{T}$ to $\mathbf{Set}$. For any other object $\mathcal{X} \in \mathbf{E}$, the category $\mathbf{E}(\mathcal{T}, \mathcal{X})$ can be seen as the category of internal $\mathcal{T}$-algebras in $\mathcal{X}$.

Semantics based on signatures The semantic notions (algebras, morphisms, etc.) are derived from a syntactic notion of signature.

In the case of multisorted algebraic theories, a signature can simply be given by sets of sorts, operations and equations. Each operation has an arity, where an arity consists of a finite list of sorts for the inputs and a sort for the output of the operation. Each equation requires the data of two terms built from the operations and a finite number of variables. For example, the algebraic theory of monoids has a single sort O, an operation e of arity $() \to O$ (the neutral element), an operation m of arity $(O, O) \to O$ (the multiplication operation), and equations $m(x, e) = x$, $m(e, x) = x$ and $m(m(x, y), z) = m(x, m(y, z))$.

For generalized algebraic theories, formal definitions of signatures are somewhat more complex, as it is no longer possible to stratify sorts, operations and equations: the specification of a dependent sort may only be well-formed if some operations or equations are already present.

Generalized algebraic theories are inherently dependently typed, and any notion of signature has to involve dependent types. Cartmell originally defined GATs as collections of symbols equipped with typing rules satisfying well-formedness conditions.

Kaposi, Kovács, and Altenkirch (2019) propose a more modern approach to the specification of GATs. They define QIIT-signatures (signatures for quotient inductive-inductive types), which are presentation of GATs, as the contexts of a type theory, called the theory of signatures (ToS). The different semantic notions are then defined by induction over these signatures, namely by constructing some models of the theory of signatures. More details on this approach can be found in the PhD thesis of Kovács (2023).

Both approaches have advantages:

- Functorial semantics in a 2-category $\mathbf{E}$ are generally very compositional, as one can make use of the 2-categorical structure of $\mathbf{E}$. For instance, one gets essentially for free that given a morphism $F : \mathcal{T}_1 \to \mathcal{T}_2$ between theories, there is a functor $F^* : \mathbf{Alg}_{\mathcal{T}_1} \to \mathbf{Alg}_{\mathcal{T}_2}$ between their categories of algebras, defined simply as the precomposition functor $(- \circ F) : \mathbf{E}(\mathcal{T}_1, \mathbf{Set}) \to \mathbf{E}(\mathcal{T}_2, \mathbf{Set})$. Functorial semantics also provides a presentation-independent notion of theory: different presentations of a theory may correspond to isomorphic or equivalent objects in the 2-category.

- For an explicitly given signature, it is possible to directly compute the definitions of algebras, morphisms, etc. In the case of QIIT-signatures, these computations involve induction on the syntactic signatures. The resulting definitions are very close to how one would write the unfolded definition of the category of algebras of a concrete theory, so that the computed category of algebras has a chance to be isomorphic, rather than just equivalent, to the explicitly defined category of algebras. This is important for computations, as one can then easily transport structures between the computed and explicit categories of algebras.

A general heuristic is that working with functorial semantics is more convenient when working with abstract theories, while signatures are more convenient when working with concrete theories (that are presented by a specific signature).

The approach adopted in this thesis is to use functorial semantics, but also develop some tools that allow for explicit computation of the semantic notions. Generalized algebraic theories will be given functorial semantics in Σ-CwFs. However, instead of working in the 2-category of Σ-CwFs and pseudo-morphisms, we will work in the

1-category of Σ-CwFs and strict Σ-CwF morphisms. This allows for the use of colimits with 1-categorical universal properties, which are determined up to isomorphism rather than up to equivalence, and are thus better suited for some computations.

We implicitly rely on the fact (that we don't prove or assume; it is only used for intuition) that the 1-category of Σ-CwF and strict morphisms has a model structure that presents the 2-category of Σ-CwFs and pseudo-morphisms. In other words, the 2-categorical structure can already be described at the 1-categorical level. For example, instead of working with 2-cells $\alpha : F \Rightarrow G$ between morphisms $F, G : \mathcal{T} \to \mathcal{C}$, we will consider dependent morphisms $\alpha : \mathcal{T} \to \mathbf{Arr}_{\mathcal{C}}[F, G]$ into an arrow-object.

A GAT will be identified with its classifying Σ-CwF $\mathcal{T}$:

- The types of $\mathcal{T}$ correspond to the (derived) sorts of the theory. For example, in the theory $\mathcal{T}_{\mathbf{Cat}}$ of categories, we have base types

$$(\vdash \mathsf{Ob}\ \mathsf{type}),$$
$$(x, y : \mathsf{Ob} \vdash \mathsf{Hom}(x, y)\ \mathsf{type}),$$
$$(x, y : \mathsf{Ob}, f, g : \mathsf{Hom}(x, y) \vdash \mathsf{EqHom}(f, g)\ \mathsf{type}),$$

  The other types are derived from these, e.g. over the context $(x, y : \mathsf{Ob}, f : \mathsf{Hom}(x, y))$, we have a type of sections of f:

$$(s : \mathsf{Hom}(y, x)) \times \mathsf{EqHom}(f \circ s, \mathsf{id}).$$

- The terms of $\mathcal{T}$ correspond to the (derived) operations of the theory. The theory of categories has two generating terms id and $(- \circ -)$. Other terms include for instance n-ary composition over a context consisting of a sequence of n composable morphisms.
- Two terms of $\mathcal{T}$ being equal corresponds to the existence of a (derived) equation between two operations.
- Syntactic signatures for GATs correspond to Σ-CwFs that are presented by collections of generating types, generating terms and generating equations between terms. They satisfy a universal property in the 1-category of Σ-CwFs and strict morphisms.

4.1 Definition and examples

Definition 4.1.1. A **generalized algebraic theory**, or **GAT**, is a type-presented Σ-CwF.

A **GAT morphism** is a strict Σ-CwF morphism (it does not have to preserve the base types). We write **GAT** for the category of GATs. ⌟

Restricting GATs to *type-presented* Σ-CwFs is needed so that the semantics can be developed by looking at strict morphisms $(\mathcal{T} \to -)$ rather than pseudo-morphisms $(\mathcal{T} \to_{\mathsf{ps}} -)$. Indeed, as shown in proposition 3.3.17, any pseudo-morphism $(\mathcal{T} \to_{\mathsf{ps}} -)$ can uniquely be replaced by a strict morphism. The restriction to type-presented Σ-CwFs is similar to the restriction of Lawvere theories to categories whose underlying set of objects is exactly $\mathbb{N}$.

A type-presented GAT is typically an $\{I^{\mathsf{ty}}, I^{\mathsf{tm}}, E^{\mathsf{tm}}\}$-cellular Σ-CwF, i.e. a Σ-CwF obtained by repeatedly extending the initial Σ-CwF by a new type, a new term, or a

new equality between terms. Indeed, the base types can then be read from the cellular presentation, so as to obtain a type-presented Σ-CwF. More concretely, any $\{I^{\mathsf{ty}}, I^{\mathsf{tm}}, E^{\mathsf{tm}}\}$-cellular Σ-CwF can be seen as a $\{I^{\mathsf{bty}}, I^{\mathsf{tm}}, E^{\mathsf{tm}}\}$-cellular Σ-CwF with base types, which is type-presented. Equations between base types are also allowed by our definition of GAT, but they are never used in concrete presentations.

When presenting a GAT, instead of explicitly giving its construction as a $\{I^{\mathsf{ty}}, I^{\mathsf{tm}}, E^{\mathsf{tm}}\}$-cellular object, we write a signature listing its generating types, terms and equalities between terms. The notion of signature is kept informal for now, but we will see formal notions of finite signatures in section 4.4.

Let's consider some examples:

Example 4.1.2. The theory $\mathcal{T}_{\mathbf{Set}}$ of sets is presented by a single generating type

$$X : \mathsf{Ty}.$$

The above presentation should be interpreted as a definition of $\mathcal{T}_{\mathbf{Set}}$ as the extension of the initial Σ-CwF $\mathbf{0}_{\mathbf{CwF}_\Sigma}$ by a new type in the empty context:

$$\mathcal{T}_{\mathbf{Set}} \triangleq \mathbf{0}_{\mathbf{CwF}_\Sigma}[\vdash \underline{X}\ \mathsf{type}]. \qquad \lrcorner$$

We call $\mathcal{T}_{\mathbf{Set}}$ the theory of sets because its set-valued algebras are exactly sets, but it might have been better to call it the theory of types, since its internal algebras in a Σ-CwF $\mathcal{E}$ (strict morphisms $\mathcal{T}_{\mathbf{Set}} \to \mathcal{E}$) are exactly closed types of $\mathcal{E}$. This follows from the universal property of $\mathcal{T}_{\mathbf{Set}}$. The only base type of $\mathcal{T}_{\mathbf{Set}}$ is the generator $\underline{X}$.

Example 4.1.3. The theory $\mathcal{T}_{\mathbf{Prop}}$ of propositions is the extension of $\mathcal{T}_{\mathbf{Set}}$ by the equation

$$(x, y : X) \to (x = y).$$

This means that $\mathcal{T}_{\mathbf{Prop}}$ is defined as the following extension of $\mathcal{T}_{\mathbf{Set}}$

$$\mathcal{T}_{\mathbf{Prop}} \triangleq \mathcal{T}_{\mathbf{Set}}[x, y : \underline{X} \vdash x = y]. \qquad \lrcorner$$

Example 4.1.4. The theory $\mathcal{T}_{\mathbf{Fam}}$ of families is the extension of $\mathcal{T}_{\mathbf{Set}}$ by a new type

$$Y : X \to \mathsf{Ty}.$$

In other words, it is the following extension of $\mathcal{T}_{\mathbf{Set}}$:

$$\mathcal{T}_{\mathbf{Fam}} \triangleq \mathcal{T}_{\mathbf{Set}}[x : \underline{X} \vdash \underline{Y}(x)\ \mathsf{type}]. \qquad \lrcorner$$

Example 4.1.5. The theory $\mathcal{T}_{E\text{-}\mathbf{Preord}}$ of E-preorders is presented by the following signature:

$$\begin{aligned}
&\mathsf{Ob} : \mathsf{Ty},\\
&_ \leq _ : \mathsf{Ob} \to \mathsf{Ob} \to \mathsf{Ty},\\
&\mathsf{refl} : x \leq x,\\
&\mathsf{trans} : x \leq y \to y \leq z \to x \leq z.
\end{aligned} \qquad \lrcorner$$

An E-preorder is almost a preorder, but without truncation of the order relation.

The above signature means that $\mathcal{T}_{E\text{-}\mathbf{Preord}}$ is defined as the following sequence of cellular extensions:

$$\begin{aligned}
&\mathcal{T}_{E\text{-}\mathbf{Preord}_0} \triangleq \mathbf{0}_{\mathbf{CwF}_\Sigma}[\vdash \underline{\mathsf{Ob}}\ \mathsf{type}],\\
&\mathcal{T}_{E\text{-}\mathbf{Preord}_1} \triangleq \mathcal{T}_{E\text{-}\mathbf{Preord}_0}[x, y : \underline{\mathsf{Ob}} \vdash x \leqq y\ \mathsf{type}],\\
&\mathcal{T}_{E\text{-}\mathbf{Preord}_2} \triangleq \mathcal{T}_{E\text{-}\mathbf{Preord}_1}[x : \underline{\mathsf{Ob}} \vdash \underline{\mathsf{refl}}(x) : x \leqq x],\\
&\mathcal{T}_{E\text{-}\mathbf{Preord}} \triangleq \mathcal{T}_{E\text{-}\mathbf{Preord}_2}[x, y, z : \underline{\mathsf{Ob}}, f : x \leqq y, g : y \leqq z \vdash \underline{\mathsf{trans}}(f, g) : x \leqq z].
\end{aligned}$$

It has two base types: $(\vdash \underline{\mathsf{Ob}}\ \mathsf{type}) \in \mathcal{T}_{E\text{-}\mathbf{Preord}}$ and $(x, y : \underline{\mathsf{Ob}} \vdash x \leqq y\ \mathsf{type}) \in \mathcal{T}_{E\text{-}\mathbf{Preord}}$.

Example 4.1.6. The theory $\mathcal{T}_{\mathbf{Preord}}$ of preorders is the extension of $\mathcal{T}_{E\text{-}\mathbf{Preord}}$ by the equation:

$$(f, g : x \leq y) \to (f = g).$$

⌟

Example 4.1.7. The theory $\mathcal{T}_{\mathbf{Poset}}$ of posets is the extension of $\mathcal{T}_{\mathbf{Preord}}$ by the equation:

$$(x \leq y) \to (y \leq x) \to (x = y).$$

⌟

Example 4.1.8. The theory $\mathcal{T}_{E\text{-}\mathbf{Cat}}$ of E-categories is presented by the following signature:

$$\begin{aligned}
&\mathsf{Ob} : \mathsf{Ty},\\
&\mathsf{Hom} : \mathsf{Ob} \to \mathsf{Ob} \to \mathsf{Ty},\\
&\mathsf{EqHom} : \mathsf{Hom}(x, y) \to \mathsf{Hom}(x, y) \to \mathsf{Ty},\\
&\mathsf{id} : \mathsf{Hom}(x, x),\\
&_ \circ _ : \mathsf{Hom}(y, z) \to \mathsf{Hom}(x, y) \to \mathsf{Hom}(x, z),\\
&\mathsf{idl} : \mathsf{EqHom}(\mathsf{id} \circ f, f),\\
&\mathsf{idr} : \mathsf{EqHom}(f \circ \mathsf{id}, f),\\
&\mathsf{assoc} : \mathsf{EqHom}((f \circ g) \circ h, f \circ (g \circ h)),\\
&\mathsf{refl} : \mathsf{EqHom}(f, f),\\
&\mathsf{trans} : \mathsf{EqHom}(f, g) \to \mathsf{EqHom}(g, h) \to \mathsf{EqHom}(f, h),\\
&\mathsf{sym} : \mathsf{EqHom}(f, g) \to \mathsf{EqHom}(g, f),\\
&_ \circ _ : \mathsf{EqHom}(f, g) \to \mathsf{EqHom}(k, l) \to \mathsf{EqHom}(f \circ k, g \circ l).
\end{aligned}$$

Note that this signature does not include any equation. ⌟

Example 4.1.9. The theory $\mathcal{T}_{\mathbf{Cat}}$ of categories is the extension of $\mathcal{T}_{E\text{-}\mathbf{Cat}}$ by the following reflection rule:

$$\begin{aligned}
&\mathsf{EqHom}(f, g) \to (f = g),\\
&(p : \mathsf{EqHom}(f, g)) \to (p = \mathsf{refl}).
\end{aligned}$$

⌟

Example 4.1.10. There is a GAT $\mathcal{T}_{\mathbf{CwF}}$ of CwFs, extending the GAT $\mathcal{T}_{\mathbf{Cat}}$ by two new sorts

$$\begin{aligned}
&\mathsf{ty} : \mathsf{Ob} \to \mathsf{Ty},\\
&\mathsf{tm} : (\Gamma : \mathsf{Ob}) \to \mathsf{ty}(\Gamma) \to \mathsf{Ty},
\end{aligned}$$

and the following operations and equations:

$$
\begin{aligned}
&\diamond : \mathsf{Ob},\\
&\varepsilon : \mathsf{Hom}(\Gamma, \diamond),\\
&- : (f : \mathsf{Hom}(\Gamma, \diamond)) \to (f = \varepsilon),\\
&-[-] : (A : \mathsf{ty}(\Gamma)) \to \mathsf{Hom}(\Delta, \Gamma) \to \mathsf{ty}(\Delta),\\
&- : (A : \mathsf{ty}(\Gamma)) \to A[\mathsf{id}] = A,\\
&- : (A : \mathsf{ty}(\Gamma)) \to A[f \circ g] = A[f][g],\\
&-[-] : (a : \mathsf{tm}(\Gamma, A)) \to (f : \mathsf{Hom}(\Delta, \Gamma)) \to \mathsf{tm}(\Delta, A[f]),\\
&- : (a : \mathsf{tm}(\Gamma, A)) \to a[\mathsf{id}] = a,\\
&- : (a : \mathsf{tm}(\Gamma, A)) \to a[f \circ g] = a[f][g],\\
&(- \triangleright -) : (\Gamma : \mathsf{Ob})(A : \mathsf{ty}(\Gamma)) \to \mathsf{Ob},\\
&\langle -, - \rangle : ((f : \mathsf{Hom}(\Delta, \Gamma)) \times \mathsf{tm}(\Delta, A[f])) \cong \mathsf{Hom}(\Delta, \Gamma \triangleright A),\\
&- : (\langle f, a \rangle \circ g) = \langle f \circ g, a[g] \rangle.
\end{aligned}
$$

For some purposes, we may also include a sort of term equalities:

$$
\begin{aligned}
&\mathsf{Eq}_{\mathsf{tm}} : (x, y : \mathsf{tm}(\Gamma, A)) \to \mathsf{Ty},\\
&(x : \mathsf{tm}(\Gamma, A)) \to \mathsf{Eq}_{\mathsf{tm}}(x, x),\\
&(p : \mathsf{Eq}_{\mathsf{tm}}(x, y)) \to (x = y),\\
&(p, q : \mathsf{Eq}_{\mathsf{tm}}(x, y)) \to (p = q).
\end{aligned}
$$

Indeed, some useful GAT morphisms $\mathcal{T} \to \mathcal{T}_{\mathbf{CwF}}$ can only be written if term equalities are included. On the other hand, including this sort is not compatible with homotopical semantics, as it forces a strict notion of identification between terms. ⌟

Example 4.1.11. There are GATs $\mathcal{T}_{\mathbf{CwF}_\Sigma}$, $\mathcal{T}_{\mathbf{CwF}_{\Sigma,\Pi_{\mathsf{rep}}}}$, $\mathcal{T}_{\mathbf{CwF}_{\Sigma,\Pi}}$, etc. extending $\mathcal{T}_{\mathbf{CwF}}$. For example, $\mathcal{T}_{\mathbf{CwF}_\Sigma}$ extends $\mathcal{T}_{\mathbf{CwF}}$ with the following operations and equations:

$$
\begin{aligned}
&\top : \mathsf{ty}(\Gamma),\\
&- : \top[f] = \top,\\
&\mathsf{tt} : \mathsf{tm}(\Gamma, \top),\\
&- : \mathsf{tt}[f] = \mathsf{tt},\\
&- : (x : \mathsf{tm}(\Gamma, \top)) \to (x = \mathsf{tt}),\\
&\Sigma : (A : \mathsf{ty}(\Gamma))(B : \mathsf{ty}(\Gamma \triangleright A)) \to \mathsf{ty}(\Gamma),\\
&- : \Sigma(A, B)[f] = \Sigma(A[f], B[\langle f \circ \boldsymbol{p}, \boldsymbol{q} \rangle]),\\
&\mathsf{pair} : (a : \mathsf{tm}(\Gamma, A)) \times (b : \mathsf{tm}(\Gamma, B[\langle \mathsf{id}, a \rangle])) \cong \mathsf{tm}(\Gamma, \Sigma(A, B)),\\
&- : \mathsf{pair}(a, b)[f] = \mathsf{pair}(a[f], b[f]).
\end{aligned}
$$

At this point, it becomes very tedious to list all operations and equations; in particular the new operations all come together with a naturality equation. We will see in section 5.2 that these complex GATs can be computed from simpler SOGATs. ⌟

4.2 Functorial semantics

We start by defining, for any GAT $\mathcal{T}$, its category of algebras $\mathbf{Alg}_{\mathcal{T}}$.

4.2.1 Categories of algebras

Let $\mathcal{T}$ be a fixed GAT.

Definition 4.2.1. A **$\mathcal{T}$-algebra** is a strict Σ-CwF morphism $\mathcal{T} \to \mathbf{Set}$. ⌟

Thus, given a $\mathcal{T}$-algebra $\mathcal{M} : \mathcal{T} \to \mathbf{Set}$, we have:

- For every object $\Gamma \in \mathcal{T}$, a set $\mathcal{M}_\Gamma$.
- For every type $(\Gamma \vdash A \ \mathsf{type}) \in \mathcal{T}$, a family $\mathcal{M}_A : \mathcal{M}_\Gamma \to \mathsf{Set}$.
- For every term $(\Gamma \vdash a : A) \in \mathcal{T}$, a dependent function $\mathcal{M}_a : (\gamma : \mathcal{M}_\Gamma) \to \mathcal{M}_A(\gamma)$.
- Subject to the equations that hold in $\mathcal{T}$.

When $\mathcal{T}$ is presented by a signature, we only need to consider this data for the generators of $\mathcal{T}$.

Example 4.2.2. By the universal property of $\mathcal{T}_{\mathbf{Set}}$, the data of a strict Σ-CwF morphism $\mathcal{T}_{\mathbf{Set}} \to \mathbf{Set}$ corresponds exactly to the data of a closed type of **Set**, i.e. of a set. The same holds for our other example GATs, e.g. the data of a strict Σ-CwF $\mathcal{T}_{\mathbf{Cat}} \to \mathbf{Set}$ corresponds exactly to the data of a category. In such cases, we will use $[\mathcal{M}] : \mathcal{T}_{\mathbf{X}} \to \mathbf{Set}$ for the (unique) morphism corresponding to an object $\mathcal{M} \in \mathbf{X}$. The object $\mathcal{M}$ can be recovered by applying $[\mathcal{M}]$ to the generating sorts of $\mathcal{T}_{\mathbf{X}}$. For example, given a category $\mathcal{C}$, then its set of objects is $[\mathcal{C}]_{\mathsf{Ob}}$, its family of morphisms is $[\mathcal{C}]_{\mathsf{Hom}}$, etc. ⌟

Remark that the displayed category **Func** (over $\mathbf{Set} \times \mathbf{Set}$) of functions extends canonically to a displayed Σ-CwF over $\mathbf{Set} \times \mathbf{Set}$. A displayed type of **Func** over a function $f : \Gamma_1 \to \Gamma_2$ and two families $A_1 : \Gamma_1 \to \mathsf{Set}$ and $A_2 : \Gamma_2 \to \mathsf{Set}$ is a dependent function $g : (\gamma : \Gamma_1) \to A_1(\gamma) \to A_2(f(\gamma))$. A displayed term of that type over $a_1 : (\gamma : \Gamma_1) \to A_1(\gamma)$ and $a_2 : (\gamma : \Gamma_2) \to A_2(\gamma)$ is a proof of $(\gamma : \Gamma_1) \to g(\gamma, a_1(\gamma)) = a_2(f(\gamma))$.

Definition 4.2.3. A **morphism** between $\mathcal{T}$-algebras $\mathcal{M}, \mathcal{N} : \mathcal{T} \to \mathbf{Set}$ is a dependent Σ-CwF morphism $F : \mathcal{T} \to \mathbf{Func}[\mathcal{M}, \mathcal{N}]$. ⌟

Given a $\mathcal{T}$-algebra morphism $F : \mathcal{T} \to \mathbf{Func}[\mathcal{M}, \mathcal{N}]$, we have:

- For every object $\Gamma \in \mathcal{T}$, a function $F_\Gamma : \mathcal{M}_\Gamma \to \mathcal{N}_\Gamma$.
- For every type $(\Gamma \vdash A \ \mathsf{type}) \in \mathcal{T}$, a dependent function
$$F_A : (\gamma : \mathcal{M}_\Gamma)(a : \mathcal{M}_A(\gamma)) \to \mathcal{N}_A(F_\Gamma(\gamma)),$$
which is the action of F on the sort A.
- For every term $(\Gamma \vdash a : A) \in \mathcal{T}$, $\gamma : \mathcal{M}_\Gamma$, we have $F_A(\mathcal{M}_a(\gamma)) = \mathcal{N}_a(F_\Gamma(\gamma))$.
- Note that the interpretation of terms is propositional, so any equation between terms in $\mathcal{T}$ holds automatically in $\mathbf{Func}[\mathcal{M}, \mathcal{N}]$.
- Equations corresponding to the preservation of the CwF structure hold. In particular,
$$F_{\Gamma.A}(\gamma, a) = (F_\Gamma(\gamma), F_A(\gamma, a)),$$
$$F_{\Sigma(A,B)}(\gamma, (a, b)) = (F_A(\gamma, a), F_B((\gamma, a), b));$$
the action of F on derived sorts can be derived from its action on basic sorts.

Example 4.2.4. Given two sets X, Y identified with morphisms $[X], [Y] : \mathcal{T}_{\mathbf{Set}} \to \mathbf{Set}$, the data of a dependent Σ-CwF morphism $F : \mathcal{T}_{\mathbf{Set}} \to \mathbf{Func}[[X], [Y]]$ is isomorphic to the data of a closed type of **Func** displayed over X and Y, i.e. a function from X to Y.

Similarly, the data of a $\mathcal{T}_{\mathbf{Cat}}$-algebra morphism is isomorphic to the data of a functor between categories. ⌟

We have dependent Σ-CwF morphisms

$$\mathsf{Id} : (X : \mathbf{Set}) \to \mathbf{Func}[X, X],$$
$$\mathsf{Comp} : (X, Y, Z : \mathbf{Set}, F : \mathbf{Func}[Y, Z], G : \mathbf{Func}[X, Y]) \to \mathbf{Func}[X, Z]$$

satisfying the categorical laws. Here $(X, Y, Z : \mathbf{Set}, F : \mathbf{Func}[Y, Z], G : \mathbf{Func}[X, Y])$ is a notation that should be interpreted in the CwF of Σ-CwFs and displayed Σ-CwFs, it corresponds to the category of composable pairs of functions, also definable as the pullback $\mathbf{Func} \times_{\mathbf{Set}} \mathbf{Func}$.

This implies that the Σ-CwFs **Set** and **Func** are the Σ-CwFs of objects and morphisms of an internal category in $\mathbf{CwF}_\Sigma$. As a consequence, the whole internal category can be externalized: the $\mathcal{T}$-algebras and $\mathcal{T}$-algebra morphisms are the objects and morphisms of a 1-category $\mathbf{Alg}_\mathcal{T}$. Given a $\mathcal{T}$-algebra $\mathcal{M} : \mathcal{T} \to \mathbf{Set}$, the identity morphism is given by the composition $(\mathsf{Id} \circ \mathcal{M}) : \mathcal{T} \to \mathbf{Func}[\mathcal{M}, \mathcal{M}]$. Given two composable $\mathcal{T}$-algebra morphisms $F : \mathcal{T} \to \mathbf{Func}[\mathcal{B}, \mathcal{C}]$ and $G : \mathcal{T} \to \mathbf{Func}[\mathcal{A}, \mathcal{B}]$, their composition is defined as the composition $(\mathsf{Comp} \circ \langle F, G \rangle) : \mathcal{T} \to \mathbf{Func}[\mathcal{A}, \mathcal{C}]$.

Example 4.2.5. The category $\mathbf{Alg}_{\mathcal{T}_{\mathbf{Set}}}$ is isomorphic to **Set**, the category $\mathbf{Alg}_{\mathcal{T}_{\mathbf{Cat}}}$ is isomorphic to **Cat**. ⌟

4.2.2 The reflective embedding of algebras into presheaves

We show that category of $\mathcal{T}$-algebras is equivalent to the category of functors $\mathcal{T} \to \mathbf{Set}$ preserving some finite limits corresponding to the terminal object and context extensions of $\mathcal{T}$. This means that $\mathbf{Alg}_\mathcal{T}$ is the category of models of a finite limit sketch and is therefore locally finitely presentable (see Adamek and Rosicky (1994)). This implies in particular that $\mathbf{Alg}_\mathcal{T}$ is complete and cocomplete.

First note that there is a forgetful functor $\mathbf{Alg}_\mathcal{T} \to \mathbf{Psh}(\mathcal{T}^{\mathsf{op}})$. Indeed a $\mathcal{T}$-algebra consists of a functor $\mathcal{T} \to \mathbf{Set}$ with additional structure, and a $\mathcal{T}$-algebra morphism consists of a natural transformation $\mathcal{T} \to \mathbf{Func}[\mathcal{M}, \mathcal{N}]$ with additional structure.

Proposition 4.2.6. *The functor* $\mathbf{Alg}_\mathcal{T} \to \mathbf{Psh}(\mathcal{T}^{\mathsf{op}})$ *is fully faithful.*

Proof. The action of the inclusion $\mathbf{Alg}_\mathcal{T} \to \mathbf{Psh}(\mathcal{T}^{\mathsf{op}})$ on morphisms forgets the action of a dependent Σ-CwF morphism $\mathcal{T} \to \mathbf{Func}[X, Y]$ on types and terms. Thus we have to show that any dependent functor $\mathcal{T} \to \mathbf{Func}[X, Y]$ can uniquely be extended to a dependent Σ-CwF morphism.

Given any dependent functor $F : \mathcal{T} \to \mathbf{Func}[X, Y]$ and type $(\Gamma \vdash A\ \mathsf{type}) \in \mathcal{T}$, we have a commuting square

$$\begin{array}{ccc} X(\Gamma.A) & \xrightarrow{F(\Gamma.A)} & Y(\Gamma.A) \\ {\scriptstyle X(\mathbf{p}_A)}\downarrow & & \downarrow{\scriptstyle Y(\mathbf{p}_A)} \\ X(\Gamma) & \xrightarrow[F(\Gamma)]{} & Y(\Gamma), \end{array}$$

where $X(\Gamma.A) = (\gamma : X(\Gamma)) \times X(A, \gamma)$ and $Y(\Gamma.A) = (\gamma : Y(\Gamma)) \times Y(A, \gamma)$.

Any action of F on types has to satisfy the equation $F(\Gamma.A) = \langle F(\Gamma), F(A)\rangle$, corresponding to preservation of context extensions. Conversely, this equation determines an action of F on types that preserves context extensions, namely

$$F(A, \gamma, a) = \pi_2(F(\Gamma.A, (\gamma, a))).$$

The action on terms is then uniquely determined since F preserves sections. Therefore any dependent functor $\mathcal{T} \to \mathbf{Func}[X, Y]$ can uniquely be extended to a dependent Σ-CwF morphism. □

Proposition 4.2.7. *A presheaf $\mathcal{M} : \mathbf{Psh}(\mathcal{T}^{\mathsf{op}})$ is in the essential image of the inclusion $\mathbf{Alg}_{\mathcal{T}} \to \mathbf{Psh}(\mathcal{T}^{\mathsf{op}})$ if and only if $\mathcal{M}(1_{\mathcal{T}})$ is terminal and for any $(\partial A \vdash A\ \mathsf{type}) \in \mathcal{T}$, squares of the following form are pullback squares:*

$$\begin{array}{ccc} \mathcal{M}(\Gamma.A) & \xrightarrow{\mathcal{M}(\langle x, q_{A[x]}\rangle)} & \mathcal{M}(\partial A.A) \\ {\scriptstyle \mathcal{M}(p_{A[x]})}\downarrow & & \downarrow{\scriptstyle \mathcal{M}(p_A)} \\ \mathcal{M}(\Gamma) & \xrightarrow[\mathcal{M}(x)]{} & \mathcal{M}(\partial A). \end{array}$$

Proof. The condition from the statement says that $\mathcal{M} : \mathcal{T} \to \mathbf{Set}$ should preserve the terminal object and context extensions up to isomorphism. This is equivalent to $\mathcal{M}$ being a pseudo-morphism of Σ-CwFs (by proposition 3.3.3). But $\mathcal{T}$ is type-presented, so there is by proposition 3.3.17 a strict Σ-CwF morphism $\mathcal{M}' : \mathcal{T} \to \mathbf{Set}$ and an isomorphism $\mathcal{M} \cong \mathcal{M}'$. Thus $\mathcal{M}$ is in the essential image of the inclusion. □

Corollary 4.2.8. *The category $\mathbf{Alg}_{\mathcal{T}}$ is complete and cocomplete.*

Proof. See for example Adamek and Rosicky (1994). □

Proposition 4.2.9. *For every $\Gamma \in \mathcal{T}$, the representable presheaf $\yoneda(\Gamma) \in \mathbf{Psh}(\mathcal{T}^{\mathsf{op}})$ is in the essential image of the inclusion $\mathbf{Alg}_{\mathcal{T}} \to \mathbf{Psh}(\mathcal{T}^{\mathsf{op}})$.*

Proof. By proposition 4.2.7, it suffices to check that $\mathcal{T}(\Gamma, 1_{\mathcal{T}})$ is terminal and that squares of the following form are pullback squares:

$$\begin{array}{ccc} \mathcal{T}(\Gamma, \Delta.A) & \xrightarrow{(\langle x, q_{A[x]}\rangle \circ -)} & \mathcal{T}(\Gamma, \partial A.A) \\ {\scriptstyle (p_{A[x]} \circ -)}\downarrow & & \downarrow{\scriptstyle (p_A \circ -)} \\ \mathcal{T}(\Gamma, \Delta) & \xrightarrow[(x \circ -)]{} & \mathcal{T}(\Gamma, \partial A). \end{array}$$

This follows from the fact that the square

$$\begin{array}{ccc} \Delta.A & \xrightarrow{\langle x, q_{A[x]}\rangle} & \partial A.A \\ {\scriptstyle p_{A[x]}}\downarrow & & \downarrow{\scriptstyle p_A} \\ \Delta & \xrightarrow[x]{} & \partial A, \end{array}$$

is a pullback square in $\mathcal{T}$. □

4.2.3 Displayed algebras

One may want to equip the category $\mathbf{Alg}_{\mathcal{T}}$ with additional structure. Its category structure arose from the data of an internal category $(\mathbf{Set}, \mathbf{Func})$ in $\mathbf{CwF}_{\Sigma}$. Additional structure on categories of algebras can be obtained by equipping this internal category with additional structure.

We are here interested in notions of displayed algebras and sections thereof, which should be the types and terms of a Σ-CwF structure on $\mathbf{Alg}_{\mathcal{T}}$. A displayed $\mathcal{T}_{\mathbf{Set}}$-algebra should be a family, while a displayed $\mathcal{T}_{\mathbf{Cat}}$-algebra should be a displayed category.

Construction 4.2.10. We extend the displayed category **Fam** to a displayed Σ-CwF over **Set**.

- A type of **Fam** over a family $X' : X \to \mathsf{Set}$ displayed over a type $A : X \to \mathsf{Set}$ of **Set** is a family
$$A' : (x : X) \to X'(x) \to A(x) \to \mathsf{Set}.$$
- A term of **Fam** over a family $A' : (x : X) \to X'(x) \to A(x) \to \mathsf{Set}$ and displayed over a term $a : (x : X) \to A(x)$ of **Set** is a dependent function
$$a' : (x : X)(x' : X'(x)) \to A'(x, x', a(x)).$$
- The context extension of a displayed type
$$A' : (x : X) \to X'(x) \to A(x) \to \mathsf{Set}$$
is the family
$$(X'.A')(x, a) \triangleq (x' : X'(x)) \times A'(x, x', a).$$
- We omit the other components. ⌟

Construction 4.2.11. We extend the displayed category **Sect** to a displayed Σ-CwF over **Fam**.

- A type of **Sect** over a section $f : (x : X) \to X'(x)$ displayed over a type $A' : (x : X) \to X'(x) \to A(x) \to \mathsf{Set}$ of **Fam** is a dependent function
$$g : (x : X)(a : A(x)) \to A'(x, f(x), a).$$
- A term of **Fam** over a family $f : (x : X)(a : A(x)) \to A'(x, x'(x), a)$ and displayed over a term $a' : (x : X)(x' : X'(x)) \to A'(x, x', a(x))$ of **Fam** is a witness of the equality
$$(x : X) \to a'(x, f(x)) = g(x, a(x)).$$ ⌟
- The context extension of a displayed type
$$g : (x : X)(a : A(x)) \to A'(x, f(x), a)$$
is the section
$$(f.g)(x, a) \triangleq (f(x), g(x, a)).$$
- We omit the other components.

Definition 4.2.12. Let $X : \mathcal{T} \to \mathbf{Set}$ be a $\mathcal{T}$-algebra. A **displayed $\mathcal{T}$-algebra** over X is a dependent Σ-CwF morphism $\mathcal{T} \to \mathbf{Fam}[X]$. ⌟

Definition 4.2.13. Let $Y : \mathcal{T} \to \mathbf{Fam}[X]$ be a displayed $\mathcal{T}$-algebra over a base X. A **section** of Y over X is a displayed Σ-CwF morphism $\mathcal{T} \to \mathbf{Sect}[X, Y]$. ⌟

Example 4.2.14. A computation shows that displayed $\mathcal{T}_{\mathbf{Cat}}$-algebras and their sections are exactly displayed categories and their sections as defined in section 2.2. Indeed, by the universal property of $\mathcal{T}_{\mathbf{Cat}}$, a displayed $\mathcal{T}_{\mathbf{Cat}}$-algebra $\mathcal{D} : \mathcal{T} \to \mathbf{Fam}[\mathcal{C}]$ over $\mathcal{C} : \mathcal{T} \to \mathbf{Set}$ consists of:

- a family $\mathcal{D}_{\mathsf{Ob}} : \mathcal{C}_{\mathsf{Ob}} \to \mathsf{Set}$,
- a family

$$\begin{aligned}\mathcal{D}_{\mathsf{Hom}} : (x : \mathcal{C}_{\mathsf{Ob}}) \times (y : \mathcal{C}_{\mathsf{Ob}}) \\ \to (x' : \mathcal{D}_{\mathsf{Ob}}(x)) \times (y' : \mathcal{D}_{\mathsf{Ob}}(y)) \to \mathcal{C}_{\mathsf{Hom}}(x, y) \to \mathsf{Set},\end{aligned}$$

- a dependent function $\mathcal{D}_{\mathsf{id}} : (x : \mathcal{C}_{\mathsf{Ob}}) \to (x' : \mathcal{D}_{\mathsf{Ob}}(x)) \to \mathcal{D}_{\mathsf{Hom}}(\mathcal{C}_{\mathsf{id}}(x))$,
- etc. ⌟

The algebras, algebra morphisms, displayed algebras and sections thereof should form a Σ-CwF. Because we want to not only define Σ-CwFs of set-valued algebras, but also Σ-CwFs of internal algebras, we defer its definition until later.

4.2.4 Adjunction induced by GAT morphisms

Given any GAT morphism $F : \mathcal{T}_1 \to \mathcal{T}_2$, there is a functor $F^* : \mathbf{Alg}_{\mathcal{T}_2} \to \mathbf{Alg}_{\mathcal{T}_1}$, defined by precomposition with F. Indeed, the components of $\mathbf{Alg}_{\mathcal{T}_1}$ are all defined as certain morphisms $(\mathcal{T}_1 \to -)$, which can be precomposed with $F : \mathcal{T}_2 \to \mathcal{T}_1$. This is the action on morphisms of a functor $\mathbf{Alg} : \mathbf{GAT}^{\mathsf{op}} \to \mathbf{CwF}_\Sigma$.

We now check that $F^* : \mathbf{Alg}_{\mathcal{T}_2} \to \mathbf{Alg}_{\mathcal{T}_1}$ admits a left adjoint $F_! : \mathbf{Alg}_{\mathcal{T}_1} \to \mathbf{Alg}_{\mathcal{T}_2}$.

Lemma 4.2.15. *For any GAT $\mathcal{T}$ and algebra $\mathcal{M} \in \mathbf{Alg}_{\mathcal{T}}$, the coslice category $(\mathbf{Alg}_{\mathcal{T}} \backslash \mathcal{M})$ is the category of algebras of a GAT $\mathcal{T}[\backslash \mathcal{M}]$, and the projection functor $(\mathbf{Alg}_{\mathcal{T}} \backslash \mathcal{M}) \to \mathbf{Alg}_{\mathcal{T}}$ is induced by a GAT morphism $\mathcal{T} \to \mathcal{T}[\backslash \mathcal{M}]$. Moreover, the morphism $\mathcal{T} \to \mathcal{T}[\backslash \mathcal{M}]$ preserves base types and is a type-presented extension.*

Proof. We construct $\mathcal{T}[\backslash \mathcal{M}]$ as an extension of $\mathcal{T}$ (in $\mathbf{CwF}_\Sigma^{\mathsf{base}}$).

- For every object $X \in \mathcal{T}$ and $(x : X) \in \mathcal{M}$, there is a new term $(\vdash g_x : X) \in \mathcal{T}[\backslash \mathcal{M}]$.
- For every type $(X \vdash Y \text{ type}) \in \mathcal{T}$ and $(y : Y(x)) \in \mathcal{M}$, there is a new term $(\vdash g_y : Y[g_x]) \in \mathcal{T}[\backslash \mathcal{M}]$.
- For every morphism $(f : X \to Y) \in \mathcal{T}$ and $(x : X) \in \mathcal{M}$, we have $f(g_x) = g_{\mathcal{M}_f(x)}$.
- For every term $(X \vdash f : Y) \in \mathcal{T}$ and $(x : X) \in \mathcal{M}$, we have $f(g_x) = g_{\mathcal{M}_f(x)}$.

Because we only add new (closed) terms and equations between terms, $\mathcal{T}[\backslash \mathcal{M}]$ is still type-presented, with the same base types as $\mathcal{T}$. It is clear from the universal property of $\mathcal{T}[\backslash \mathcal{M}]$ that $\mathbf{Alg}_{\mathcal{T}[\backslash \mathcal{M}]} \cong (\mathbf{Alg}_{\mathcal{T}} \backslash \mathcal{M})$. □

Theorem 4.2.16. *For any GAT morphism $F : \mathcal{T}_1 \to \mathcal{T}_2$, the functor $F^* : \mathbf{Alg}_{\mathcal{T}_2} \to \mathbf{Alg}_{\mathcal{T}_1}$ has a left adjoint $F_! : \mathbf{Alg}_{\mathcal{T}_1} \to \mathbf{Alg}_{\mathcal{T}_2}$.*

Proof. It suffices to prove that for every $\mathcal{T}_1$-algebra $\mathcal{M}$, the comma category $(\mathcal{M} \downarrow F^*)$, whose objects are pairs of $\mathcal{N} : \mathbf{Alg}_{\mathcal{T}_2}$ and $g : \mathcal{M} \to F^*(\mathcal{N})$, has an initial object. Note that the comma category $(\mathcal{M} \downarrow F^*)$ can be computed as the following pullback:

$$\begin{array}{ccc} (\mathcal{M} \downarrow F^*) & \longrightarrow & (\mathbf{Alg}_{\mathcal{T}_1} \backslash \mathcal{M}) \\ \downarrow & \lrcorner & \downarrow \\ \mathbf{Alg}_{\mathcal{T}_2} & \xrightarrow{F^*} & \mathbf{Alg}_{\mathcal{T}_1}. \end{array}$$

By lemma 4.2.15 the coslice category $(\mathbf{Alg}_{\mathcal{T}_1} \backslash \mathcal{M})$ is the category of algebras of a GAT $\mathcal{T}_1[\backslash \mathcal{M}]$.

The comma category $(\mathcal{M} \downarrow F^*)$ is therefore the category of algebras of the GAT $\mathcal{T}_3$ computed as the following pushout:

$$\begin{array}{ccc} \mathcal{T}_1 & \xrightarrow{F} & \mathcal{T}_2 \\ \downarrow & \ulcorner & \downarrow \\ \mathcal{T}_1[\backslash \mathcal{M}] & \longrightarrow & \mathcal{T}_3. \end{array}$$

The Σ-CwF $\mathcal{T}_3$ is type-presented because it is a type-presented extension of the GAT $\mathcal{T}_2$. Indeed, the Σ-CwF morphism $\mathcal{T}_1 \to \mathcal{T}_1[\backslash \mathcal{M}]$ is a type-presented extension and type-presented extensions are preserved by pushouts. □

4.3 Internal algebras

We have now defined the category of set-valued algebras of a GAT $\mathcal{T}$. Thanks to functorial semantics, we may also consider algebras valued into any other $\mathcal{E} \in \mathbf{CwF}_\Sigma$, that is functors $\mathcal{T} \to \mathcal{E}$. We are interested in particular in functors $\mathcal{T}_1 \to \mathbf{Alg}_{\mathcal{T}_2}$ valued in other categories of algebras, and in functors $\mathcal{T}_1 \to \mathbf{Psh}(\mathcal{C})$ valued in presheaf categories.

4.3.1 Definitions

Definition 4.3.1. Let $\mathcal{T}$ be a GAT and $\mathcal{E}$ be a Σ-CwF. An **internal $\mathcal{T}$-algebra in $\mathcal{E}$** is a Σ-CwF morphism $\mathcal{T} \to \mathcal{E}$. ⌟

Example 4.3.2. An internal $\mathcal{T}_{\mathbf{Set}}$-algebra in $\mathcal{E}$ is a closed type of $\mathcal{E}$.

An internal $\mathcal{T}_{\mathbf{Preord}}$-algebra in $\mathcal{E}$ is an internal preorder in $\mathcal{E}$, consisting of:

- A closed type $(\vdash \mathsf{Ob}\ \mathsf{type}) \in \mathcal{E}$;
- A type $(x, y : \mathsf{Ob} \vdash x \leq y\ \mathsf{type}) \in \mathcal{E}$;
- A term $(x : \mathsf{Ob} \vdash \mathsf{refl}(x) : x \leq x) \in \mathcal{E}$;
- A term $(x, y, z : \mathsf{Ob}, f : y \leq z, g : x \leq y \vdash \mathsf{trans}(f, g) : x \leq z) \in \mathcal{E}$;
- A term equality $(x, y : \mathsf{Ob}, f, g : x \leq y \vdash f = g) \in \mathcal{E}$. ⌟

Definition 4.3.3. We define a dependent functor

$$\mathbf{Arr} : (\mathcal{E} : \mathbf{CwF}_\Sigma) \to \mathbf{DispCwF}_\Sigma[\mathcal{E} \times \mathcal{E}].$$

⌟

Construction. Take $\mathcal{E} : \mathbf{CwF}_\Sigma$. The underlying displayed category of $\mathbf{Arr}(\mathcal{E})$ is the arrow displayed category of $\mathcal{E}$. The rest of the components are defined analogously to the components of **Func**:

- A displayed type of $\mathbf{Arr}(\mathcal{E})$ over $(\alpha_X : X_1 \to X_2) \in \mathcal{E}$, $(X_1 \vdash A_1\ \mathsf{type}) \in \mathcal{E}$ and $(X_2 \vdash A_2\ \mathsf{type}) \in \mathcal{E}$ is a term

$$(x : X_1, a : A_1(x) \vdash \alpha_A(x, a) : A_2(\alpha_X(x)) \in \mathcal{E}.$$

- A displayed term of type α_A as above over $(X_1 \vdash a_1 : A_1) \in \mathcal{E}$ and $(X_2 \vdash a_2 : A_2)$ is a witness of the equality

$$(x : X_1 \vdash \alpha_A(x, a_1(x)) = a_2(\alpha_X(x))) \in \mathcal{E}.$$

- We omit the other components.

We obtain functoriality for free from the fact that this construction is induced by a GAT morphism $\mathcal{T}_{\mathbf{DispCwF}_\Sigma} \to \mathcal{T}_{\mathbf{CwF}_\Sigma}$. □

Definition 4.3.4. A morphism of internal $\mathcal{T}$-algebras between $\mathcal{M}, \mathcal{N} : \mathcal{T} \to \mathcal{E}$ is a Σ-CwF morphism $F : \mathcal{T} \to \mathbf{Arr}(\mathcal{E})[\mathcal{M}, \mathcal{N}]$. ⌟

Definition 4.3.5. We define a dependent functor $\mathbf{Fam} : (\mathcal{E} : \mathbf{CwF}_\Sigma) \to \mathbf{DispCwF}_\Sigma[\mathcal{E}]$. ⌟

Construction. Take $\mathcal{E} : \mathbf{CwF}_\Sigma$.

- A displayed object of $\mathbf{Fam}(\mathcal{E})$ over $X \in \mathcal{E}$ is a type

$$(x : X \vdash X'(x)\ \mathsf{type}) \in \mathcal{E}.$$

- A displayed morphism of $\mathbf{Fam}(\mathcal{E})$ over $f : X \to Y$, X' and Y' is a term

$$(x : X, x' : X'(x) \vdash f'(x) : Y'(f(x))) \in \mathcal{E}.$$

- A displayed type of $\mathbf{Fam}(\mathcal{E})$ over $(X \vdash A\ \mathsf{type})$ and X' is a type

$$(x : X, x' : X'(x), a : A(x) \vdash A'(x, x', a)\ \mathsf{type}) \in \mathcal{E}.$$

- A displayed term of type A' as above over $(X \vdash a : A)$ is a term

$$(x : X, x' : X'(x) \vdash a'(x, x', a) : A'(x, x', a(x))) \in \mathcal{E}.$$

- The extension of a context X' by a type A' is the type

$$(x : X, a : A(x) \vdash (x' : X'(x)) \times (a' : A'(x, x', a))\ \mathsf{type}) \in \mathcal{E}.$$

- We omit the rest of the components.

We obtain functoriality for free from the fact that this construction is induced by a GAT morphism $\mathcal{T}_{\mathbf{DispCwF}_\Sigma} \to \mathcal{T}_{\mathbf{CwF}_\Sigma}$. □

Definition 4.3.6. An internal displayed $\mathcal{T}$-algebra over $\mathcal{M} : \mathcal{T} \to \mathcal{E}$ is a Σ-CwF morphism $\mathcal{M}' : \mathcal{T} \to \mathbf{Fam}(\mathcal{E})[\mathcal{M}]$. ⌟

Definition 4.3.7. We define a dependent functor $\mathbf{Sect} : (\mathcal{E} : \mathbf{CwF}_\Sigma) \to \mathbf{DispCwF}_\Sigma[\mathbf{Fam}(\mathcal{E})]$. ⌟

Proof. Take $\mathcal{E} : \mathbf{CwF}_\Sigma$.

- A displayed object of $\mathbf{Sect}(\mathcal{E})$ over $X \in \mathcal{E}$ and $X' \in \mathbf{Fam}(\mathcal{E})[X]$ is a term
$$(x : X \vdash \alpha_X(x) : X'(x)) \in \mathcal{E}.$$
- A displayed morphism of $\mathbf{Sect}(\mathcal{E})$ over $f : X \to Y$, f', α_X and α_Y is a witness of the equality
$$(x : X \vdash f'(x, \alpha_X(x)) = \alpha_Y(f(x))) \in \mathcal{E}.$$
- A displayed type of $\mathbf{Sect}(\mathcal{E})$ over $(X \vdash A\ \mathsf{type})$, X', A' and α_X is a term
$$(x : X, a : A \vdash \alpha_A(x, a) : A'(x, \alpha_X(x), a) \in \mathcal{E}.$$
- A displayed term of type α_A as above over $(X \vdash a : A)$ and a' is a witness of the equality
$$(x : X \vdash a'(x, \alpha_X(x)) = \alpha_A(x, a(x))) \in \mathcal{E}.$$
- We omit the rest of the components.

As before, we obtain functoriality for free from the fact that this construction is induced by a GAT morphism $\mathcal{T}_{\mathbf{DispCwF}_\Sigma} \to \mathcal{T}_{\mathbf{CwF}_\Sigma}$. □

Definition 4.3.8. A section of an internal displayed $\mathcal{T}$-algebra $\mathcal{M}' : \mathcal{T} \to \mathbf{Fam}(\mathcal{E})[\mathcal{M}]$ is a Σ-CwF morphism $S : \mathcal{T} \to \mathbf{Sect}(\mathcal{E})[\mathcal{M}']$. ⌟

4.3.2 CwFs of internal algebras

Our goal should now be to define a functor

$$\mathbf{Alg} : \mathbf{GAT}^{\mathrm{op}} \times \mathbf{CwF}_\Sigma \to \mathbf{CwF}_\Sigma,$$

where $\mathbf{Alg}(\mathcal{T}, \mathcal{E})$ the Σ-CwF of internal $\mathcal{T}$-algebras in $\mathcal{E}$.

Unfortunately all of the approaches that I know of require an extremely large amount of easy but tedious constructions, so I will only give a discussion of possible approaches for the construction and outline the main difficulties.

In addition to internal algebras, one also wants to construct the tensor product of GATs: the tensor product $(- \otimes \mathcal{T})$ should be left adjoint to $\mathbf{Alg}(\mathcal{T}, -)$; in particular we would have isomorphisms

$$(\mathcal{T}_1 \otimes \mathcal{T}_2 \to \mathcal{E}) \cong (\mathcal{T}_1 \to \mathbf{Alg}(\mathcal{T}_2, \mathcal{E})),$$

which further internalize to

$$\mathbf{Alg}(\mathcal{T}_1 \otimes \mathcal{T}_2, \mathcal{E}) \cong \mathbf{Alg}(\mathcal{T}_1, \mathbf{Alg}(\mathcal{T}_2, \mathcal{E})).$$

Moreover, the tensor product and the internal algebra functor should be part of a closed symmetric monoidal category structure on GATs. This suggests two approaches:

- Start with internal algebra functor and derive the tensor product as a left adjoint.

- Start with the tensor product and derive the internal algebra functor as a right adjoint.

In either case, one can use the fact that $(\mathcal{T} \otimes -)$ ought to preserve colimits and $\mathbf{Alg}(-, \mathcal{E})$ ought to send colimits to limits, together with cellular presentations of GATs.

We have defined the category structure of $\mathbf{Alg}_{\mathcal{T}}$ from an internal category in $\mathbf{CwF}_\Sigma$. Similarly, we should expect to define the Σ-CwF structure of $\mathbf{Alg}(\mathcal{T}, \mathcal{E})$ from an internal Σ-CwF in $\mathbf{CwF}_\Sigma$. The CwFs and displayed CwFs $\mathcal{E}$, $\mathbf{Arr}(\mathcal{E})$, $\mathbf{Fam}(\mathcal{E})$ and $\mathbf{Sect}(\mathcal{E})$ should be the interpretations of the sorts of this internal Σ-CwF.

There is unfortunately a first issue with this approach: some of the Σ-CwF operations can only be interpreted as pseudo-morphisms of Σ-CwFs, instead of strict morphisms. Consider the functor $\Sigma : \mathbf{Fam} \to \mathbf{Set}$ that sends a family to its total space, and which should be used to interpret context extensions. This functor cannot be extended to a strict Σ-CwF morphism, because it does not preserve context extensions strictly: the sets

$$((x : X) \times (y : Y(x))) \times ((x' : X'(x)) \times Y'(x, y, x'))$$

and

$$((x : X) \times (x' : X'(x))) \times ((y : Y(x)) \times Y'(x, y, x'))$$

are isomorphic but not strictly equal. This functor is however a pseudo-morphism of Σ-CwFs.

Interpreting the Σ-CwF operations as pseudo-morphisms suffices, since we can later strictify any pseudo-morphism with source $\mathcal{T}$. This however means that we cannot easily rely on general results on the externalization of internal algebras. Generally, although $\mathbf{Alg}_{\mathcal{T}}(\mathcal{E})$ should morally be defined as the externalization at $\mathcal{T}$ of an internal Σ-CwF $\mu(\mathcal{E})$ in $\mathbf{CwF}_\Sigma$, an explicit definition of $\mu(\mathcal{E})$ as an algebraic structure requires a bit too much effort; it is simpler to define all of the components of $\mathbf{Alg}_{\mathcal{T}}(\mathcal{E})$ directly.

Remark 4.3.9. An internal category in **Cat** is called a double category. A double category has objects, horizontal morphisms, vertical morphisms, and squares involving horizontal and vertical morphisms.

Accordingly, an internal Σ-CwF in $\mathbf{CwF}_\Sigma$ could be called a double Σ-CwF. A double Σ-CwF has an underlying double category, as well as horizontal and vertical types, horizontal and vertical terms, and more exotic sorts (horizontal morphisms between vertical types, double types, double terms, etc.), for a total of 16 sorts.

CwFs are already a rather large algebraic structure, but double CwFs are so large that listing all the sorts already spans one page, and unfolding the whole structure by hand is not feasible.

We can estimate the number of operations and equations in a presentation of double Σ-CwFs. We will later define the tensor product of GATs. Internal $\mathcal{T}_{\mathbf{CwF}_\Sigma}$-algebras in $\mathbf{Alg}_{\mathcal{T}_{\mathbf{CwF}_\Sigma}}$ are in bijection with algebras of $(\mathcal{T}_{\mathbf{CwF}_\Sigma} \otimes \mathcal{T}_{\mathbf{CwF}_\Sigma})$. When two GATs $\mathcal{T}_1$ and $\mathcal{T}_2$ have s_1, s_2 generating sorts, o_1, o_2 generating operations and e_1, e_2 generating equations, then their tensor product $(\mathcal{T}_1 \otimes \mathcal{T}_2)$ has $s_1 s_2$ generating sorts, $o_1 s_2 + s_1 o_2$ generating operations and $e_1 s_2 + o_1 o_2 + s_1 e_2$ generating equations. The theory $\mathcal{T}_{\mathbf{CwF}_\Sigma}$ is presented using 4 sorts, 17 operations and 17 equations, so double Σ-CwFs can be presented with 16 sorts, 136 operations and 425 equations.

This structure corresponds approximately to the large table that is constructed by Kaposi, Kovács, and Altenkirch (2019), also for the purpose of defining the CwF of algebras of any GAT (described by a QIIT-signature). ⌟

Remark 4.3.10. The construction of $\mathbf{Alg}(\mathcal{T}, \mathcal{E})$ is closely related to the construction of the linear arrow $\mathcal{C} \multimap \mathcal{D}$ in clans with strictly commuting limits by Moeneclaey (2022).

Moeneclaey uses clans with strictly commuting limits so as to obtain a closed symmetric monoidal structure over the 1-category of clans with strictly commuting limits, whereas they are morally interested in the 2-category of clans. This is similar to our situation: we use the 1-category of type-presented Σ-CwFs as a presentation of the 2-category of Σ-CwFs and pseudo-morphisms. ⌟

We will need the following fact.

Proposition 4.3.11 (Equivalence between internal models and presheaves of models)**.** *For any category $\mathcal{C}$ and GAT $\mathcal{T}$, there is an equivalence of categories*

$$\mathbf{Alg}_{\mathcal{T}}(\mathbf{Psh}(\mathcal{C})) \cong \mathbf{Cat}(\mathcal{C}^{\mathsf{op}}, \mathbf{Alg}_{\mathcal{T}}),$$

between internal $\mathcal{T}$-algebras in $\mathbf{Psh}(\mathcal{C})$, and functors from $\mathcal{C}^{\mathsf{op}}$ to $\mathcal{T}$-algebras.

Proof. This is well-known for sketches or finite limits theories, see for example Johnstone 2002, p. D1.2.14. □

Proposition 4.3.11 would normally be derived from the fact that the tensor product of GATs is symmetric. Indeed, for any category $\mathcal{C}$, there is a GAT $\mathcal{T}_{\mathbf{Psh}(\mathcal{C})}$ that classifies presheaves over $\mathcal{C}$. Then the result is an equivalence

$$\mathbf{Alg}_{\mathcal{T}_{\mathbf{Psh}(\mathcal{C})}}(\mathbf{Alg}_{\mathcal{T}}(\mathbf{Set})) \cong \mathbf{Alg}_{\mathcal{T}}(\mathbf{Alg}_{\mathcal{T}_{\mathbf{Psh}(\mathcal{C})}}(\mathbf{Set})),$$

which would follow from the isomorphism $\mathcal{T}_{\mathbf{Psh}(\mathcal{C})} \otimes \mathcal{T} \cong \mathcal{T} \otimes \mathcal{T}_{\mathbf{Psh}(\mathcal{C})}$.

When working in the multimodal internal language of a presheaves, there is a further equivalence

$$\mathbf{Alg}_{\mathcal{T}}(\mathbf{Psh}(\mathcal{C})) \cong \Box(\mathbf{Alg}_{\Delta\mathcal{T}}(\mathbf{Set}^{\mathfrak{c}}))$$

between internal $\mathcal{T}$-algebras and $\Delta\mathcal{T}$-algebras in the internal language, where $\Delta\mathcal{T}$ is the internal GAT at mode $\mathfrak{c}$ corresponding to the external GAT $\mathcal{T}$.

4.4 Finitely generated algebras

Let $\mathcal{T}$ be a GAT. Proposition 4.2.9 implies that the Yoneda embedding $\yoneda : \mathcal{T}^{\mathsf{op}} \to \mathbf{Psh}(\mathcal{T}^{\mathsf{op}})$ induces a fully faithful functor $H : \mathcal{T} \to \mathbf{Alg}_{\mathcal{T}}^{\mathsf{op}}$ such that morphisms $H(\Gamma) \to \mathcal{M}$ are in natural bijection with elements $(\gamma : \Gamma) \in \mathcal{M}$. In other words, $H(\Gamma)$ is the algebra freely generated by an element of the sort Γ.

The goal of this section is to construct a stricter functor $\mathbf{0}_{\mathcal{T}}[-] : \mathcal{T} \to \mathbf{Alg}_{\mathcal{T}}^{\mathsf{op}}$, whose image also consists of the finitely generated algebras of $\mathcal{T}$. By "stricter", we mean that $\mathbf{0}_{\mathcal{T}}[-]$ is not only a functor, but also a strict Σ-CwF morphism. For a presented GAT $\mathcal{T}$, an object Γ of $\mathcal{T}$ can then be seen as a syntactic signature presenting the finitely generated algebra $\mathbf{0}_{\mathcal{T}}[\Gamma]$. Because $\mathbf{0}_{\mathcal{T}}[-]$ is a Σ-CwF morphism, $\mathbf{0}_{\mathcal{T}}[\Gamma]$ can be computed by induction over the "signature" Γ.

Definition 4.4.1. A **extension** of a algebra $\mathcal{M} : \mathbf{Alg}_{\mathcal{T}}$ is a map of the form

$$\mathcal{M} \to \mathcal{M}[\boldsymbol{x} : A(\sigma)]$$

where $(\partial A \vdash A\ \mathsf{type})$ is a type of $\mathcal{T}$ and $(\sigma : \partial A) \in \mathcal{M}$.

Concretely, the extension $\mathcal{M} \to \mathcal{M}[\boldsymbol{x} : A(\sigma)]$ is defined as the chosen pushout

$$\begin{array}{ccc} H(\partial A) & \xrightarrow{\langle\sigma\rangle} & \mathcal{M} \\ \downarrow & \ulcorner & \downarrow \\ H(\partial A.A) & \xrightarrow{\langle\sigma,\boldsymbol{x}\rangle} & \mathcal{M}[\boldsymbol{x} : A(\sigma)], \end{array}$$

which satisfies the necessary universal property. In particular, a extension depends only on $(\partial A, A, \sigma)$, not on the choice of the map $\mathcal{M} \to \mathcal{M}[\boldsymbol{x} : A(\sigma)]$. ⌟

Construction 4.4.2. We equip the category $\mathbf{Alg}_{\mathcal{T}}^{\mathsf{op}}$ with the structure of a Σ-CwF.

- The types over $\mathcal{M} \in \mathbf{Alg}_{\mathcal{T}}$ are the extensions of $\mathcal{M}$. The restriction of an extension $\mathcal{M} \to \mathcal{M}[\boldsymbol{x} : A(\sigma)]$ along a morphism $F : \mathcal{M} \to \mathcal{N}$ is the extension $\mathcal{N} \to \mathcal{N}[\boldsymbol{x} : A(F(\sigma))]$. These restrictions are strictly functorial, since these extensions are all computed as pushouts of $H(\partial A) \to H(\partial A.A)$.
- The terms of type $i : \mathcal{M} \to \mathcal{M}[\boldsymbol{x} : A(\sigma)]$ are the elements $(x : A(\sigma)) \in \mathcal{M}$ (or, equivalently, the retractions of i). The restriction of $(x : A(\sigma)) \in \mathcal{M}$ along $F : \mathcal{M} \to \mathcal{N}$ is the element $(F(x) : A(F(\sigma)) \in \mathcal{N}$. These restrictions are strictly functorial.
- The extension of a context by a type $\mathcal{M} \to \mathcal{M}[\boldsymbol{x} : A(\sigma)]$ is the codomain $\mathcal{M}[\boldsymbol{x} : A(\sigma)]$, as witnessed by its universal property.
- The $\mathbf{1}$- and Σ- types are defined using the $\mathbf{1}$- and Σ-types of $\mathcal{T}$.
  - The $\mathbf{1}$-type is given by the extension $\mathcal{M} \to \mathcal{M}[\boldsymbol{x} : \mathbf{1}]$.
  - The dependent sum of $\mathcal{M} \to \mathcal{M}[\boldsymbol{a} : A(\sigma)] \to \mathcal{M}[\boldsymbol{a} : A(\sigma)][\boldsymbol{b} : B(\tau(\boldsymbol{a}))]$ is the extension $\mathcal{M} \to \mathcal{M}[\boldsymbol{x} : (a : A(\sigma)) \times B(\tau(a))]$. ⌟

Proposition 4.4.3. *The functor $H : \mathcal{T} \to \mathbf{Alg}_{\mathcal{T}}^{\mathsf{op}}$ extends to a pseudo-morphism of Σ-CwFs.*

Proof. We define its action on types: a type $(\partial A \vdash A\ \mathsf{type})$ is mapped to the extension

$$H(\partial A) \to H(\partial A)[\boldsymbol{x} : A],$$

i.e. the pushout of $H(\partial A) \to H(\partial A.A)$ along $\mathsf{id} : H(\partial A) \to H(\partial A)$.

The weak preservation of the terminal object follows from the universal property of $H(1_{\mathcal{T}})$.

We have to check that context extensions are weakly preserved, which follows from the fact that $H(\partial A)[\boldsymbol{x} : A(\sigma)]$ satisfies the universal property of $H(\partial A.A)$.

The action on terms is then uniquely determined by the action on morphisms. □

Proposition 4.4.4. *The pseudo-morphism $H : \mathcal{T} \to \mathbf{Alg}_{\mathcal{T}}^{\mathsf{op}}$ is essentially surjective on types and bijective on terms.*

Proof. As H is fully faithful, it is bijective on terms, since terms can be identified with sections of projection maps.

Take $\Gamma \in \mathcal{T}$ and an extension $H(\Gamma) \to H(\Gamma)[\boldsymbol{x} : A(\sigma)]$. The element $(\sigma : \partial A) \in H(\Gamma)$ corresponds to $(\gamma : \Gamma \vdash \sigma(\gamma) : \partial A) \in \mathcal{T}$. Thus we can consider the type $(\gamma : \Gamma \vdash A(\sigma(\gamma))\ \mathsf{type}) \in \mathcal{T}$ and its image by H, which is the pushout of $H(\Gamma) \to H(\Gamma.A[\sigma])$ along $\mathsf{id} : H(\Gamma) \to H(\Gamma)$. Since it has the universal property of $H(\Gamma)[\boldsymbol{x} : A(\sigma)]$, it is isomorphic to it. Thus the actions of H on types are essentially surjective. □

By proposition 3.3.17, since $\mathcal{T}$ is type-presented, there is a strict Σ-CwF morphism $\mathbf{0}_{\mathcal{T}}[-] : \mathcal{T} \to \mathbf{Alg}_{\mathcal{T}}^{\mathsf{op}}$ that is naturally isomorphic to H. Then $\mathbf{0}_{\mathcal{T}}[-]$ is also fully faithful, essentially surjective on types and bijective on terms.

Example 4.4.5. We can consider some objects of $\mathcal{T}_{\mathbf{Cat}}$, for example

$$
\begin{aligned}
&(x : \mathsf{Ob}),\\
&(x, y : \mathsf{Ob}),\\
&(x, y : \mathsf{Ob}, f : \mathsf{Hom}(x, y)),\\
&(x, y : \mathsf{Ob}, f : \mathsf{Hom}(x, y), g : \mathsf{Hom}(y, x), p : \mathsf{EqHom}(g \circ f, \mathsf{id})).
\end{aligned}
$$

Then they are sent to freely generated categories that would typically be described by the following diagrams:

$$
\begin{aligned}
&\{x\},\\
&\{x \quad y\},\\
&\{x \to y\},\\
&\left\{ \quad x \xrightarrow{f} y \xrightarrow{g} x. \quad \overset{\curvearrowright}{\mathsf{id}} \quad \right\}
\end{aligned}
$$

⌟

4.5 Trivial fibrations

We write $I_{\mathcal{T}}$ for the set of all extensions of $\mathcal{T}$-algebras (defined in definition 4.4.1).

Definition 4.5.1. A morphism $F : \mathcal{M} \to \mathcal{N}$ of algebras is a **trivial fibration** if for every type $(\partial A \vdash A\ \mathsf{type}) \in \mathcal{T}$, the following lifting condition is satisfied: for every boundary $(\sigma : \partial A) \in \mathcal{M}$ and element $(a : Y(F(\sigma))) \in \mathcal{N}$, there merely exists an element $(a_0 : Y(\sigma)) \in \mathcal{M}$ such that $F(a_0) = a$. ⌟

The trivial fibrations are the maps in the right class of a weak factorization system generated by $I_{\mathcal{T}}$. The maps in the left class are called **cofibrations**.

We call split trivial fibrations the maps in the right class of the algebraic weak factorization system generated by $I_{\mathcal{T}}$, and algebraic cofibrations the maps in the left class.

If $\mathcal{T}$ is type-presented, we write $I_{\mathcal{T}}^{\mathsf{base}}$ for the subset of $I_{\mathcal{T}}$ spanned by the base types. For example, $I_{\mathcal{T}_{\mathbf{Cat}}}^{\mathsf{base}}$ consists of the functors

$$
\begin{aligned}
&I_{\mathsf{Ob}} : \{\} \to \{x\},\\
&I_{\mathsf{Hom}} : \{x, y\} \to \{x \to y\},\\
&I_{\mathsf{EqHom}} : \{x \rightrightarrows y\} \to \{x \to y\},
\end{aligned}
$$

which are the generic extensions by a new object, a new morphism, or a new morphism equality.

Lemma 4.5.2. *Assume that $\mathcal{T}$ is type-presented. Then the extensions coincide with the finite $I_{\mathcal{T}}^{\mathsf{base}}$-cellular maps.*

Proof. A finite $I_{\mathcal{T}}^{\mathsf{base}}$-cellular map is a finite composition of pushouts of maps in $I_{\mathcal{T}}^{\mathsf{base}}$. A type of $\mathcal{T}$ is an iterated dependent sum of substitutions of base types. There is a correspondence between these two decompositions. The correspondence becomes bijective if we only allow left-nested iterated dependent sums. □

Corollary 4.5.3. *A morphism between $\mathcal{T}$-algebras is a trivial fibration if and only if it satisfies the right lifting property with respect to $I_{\mathcal{T}}^{\mathsf{base}}$. A morphism between $\mathcal{T}$-algebras is a split trivial fibration if and only if it equipped with a right lifting structure with respect to $I_{\mathcal{T}}^{\mathsf{base}}$.* □

4.6 Congruences and fibrant congruences

In $\mathbf{Set} = \mathbf{Alg}_{\mathcal{T}_{\mathbf{Set}}}$, a map $f : X \to Y$ is a trivial fibration (surjective) if and only if it is the quotient inclusion for the equivalence relation that identifies $x_1, x_2 : X$ when $f(x_1) = f(x_2)$, i.e. the kernel of f. In this section we study a similar characterization of trivial fibrations in categories of algebras of arbitrary GATs.

Quotients in cocomplete categories are generally computed as coequalizers of diagrams $\mathcal{R} \rightrightarrows \mathcal{C}$, where $\mathcal{R}$ is a subobject of $\mathcal{C} \times \mathcal{C}$. For general GATs, computing quotients can be complex, since identifying two elements can allow the formation of new elements that were not present in the original algebra.

Some examples can be found by looking at categories; general quotients of categories have been studied by Bednarczyk, Borzyszkowski, Pawlowski, and Barr (1999). Consider the quotient $\mathbf{q} : \mathcal{X} \to \mathbf{Q}(\mathcal{X})$ of the category $\mathcal{X} \triangleq \{x \xrightarrow{f} y \quad a \xrightarrow{g} b\}$ by the relation that identifies y and a. In the quotient, we can form the composition $\mathbf{q}(g) \circ \mathbf{q}(f) : \mathbf{q}(x) \to \mathbf{q}(b)$ of f and g, yet there is no morphism $x \to y$ in the original category.

In other words, the quotient inclusion $\mathbf{q}$ is not a trivial fibration, since the morphism $\mathbf{q}(g) \circ \mathbf{q}(f)$ does not have a preimage. In order to obtain a trivial fibration, we would need to have, at the very least, an isomorphism $y \cong a$ in the original category, corresponding to a lift of the identity $\mathsf{id} : \mathbf{q}(y) \cong \mathbf{q}(a)$. More generally, we will see that the projection $\mathsf{Hom} \to \mathsf{Ob} \times \mathsf{Ob}$, when equipped with the structure of a setoid morphism induced by the chosen equivalence relations, should be a setoid fibration.

Construction 4.6.1. By setoid we mean a set equipped with a (proof-irrelevant) equivalence relation. The category **Setoid** is equipped with the structure of a Σ-CwF, displayed over the Σ-CwF of sets: indeed **Setoid** is the category of algebras of a GAT $\mathcal{T}_{\mathbf{Setoid}}$. ⌟

Definition 4.6.2. A **congruence** on an algebra $\mathcal{M}$ is a dependent Σ-CwF morphism $\widetilde{\mathcal{M}} : \mathcal{T} \to \mathbf{Setoid}[\mathcal{M}]$. ⌟

If X is an object or type of $\mathcal{T}$, we write $(x \sim x') \in \widetilde{\mathcal{M}}$ when x and x' are related elements of $\mathcal{M}_X$ at the equivalence relation $\widetilde{\mathcal{M}}_X$.

Example 4.6.3. Consider the theory of categories and a base category $\mathcal{C}$. Using the universal property of $\mathcal{T}_{\mathbf{Cat}}$, a congruence $\widetilde{\mathcal{C}}$ can be decomposed into the following components:

- An equivalence relation $(- \sim -) \in \widetilde{\mathcal{C}}$ on objects of $\mathcal{C}$.
- A displayed equivalence relation $(- \sim -) \in \widetilde{\mathcal{C}}$ on morphisms of $\mathcal{C}$ displayed twice over the equivalence relation on objects.

  This means that morphisms $(f : x \to y)$ and $(g : x' \to y')$ can be compared as long as $(x \sim x') \in \widetilde{\mathcal{C}}$ and $(y \sim y') \in \widetilde{\mathcal{C}}$.

- Such that the equivalence relations preserve identities and compositions:

$$(x \sim x') \in \widetilde{\mathcal{C}} \to (\mathsf{id}(x) \sim \mathsf{id}(x')) \in \widetilde{\mathcal{C}},$$
$$(f \sim f') \wedge (g \sim g') \to ((f \circ g) \sim (f' \circ g')) \in \widetilde{\mathcal{C}},$$

  where $(f \sim f') \wedge (g \sim g')$ presupposes the relevant relations between the sources and targets of the morphisms. ⌟

Given two congruences $\widetilde{\mathcal{M}}$ and $\overline{\mathcal{M}}$ over the same base algebra $\mathcal{M}$, we write $\widetilde{\mathcal{M}} \subseteq \overline{\mathcal{M}}$ when there is an implication $(y_1 \sim y_2) \in \widetilde{\mathcal{M}}_Y \implies (y_1 \sim y_2) \in \overline{\mathcal{M}}_Y$ for every sort $(X \vdash Y\ \mathsf{type}) \in \mathcal{T}$, or equivalently when there is a natural transformation $\widetilde{\mathcal{M}} \Rightarrow \overline{\mathcal{M}}$ in $\mathcal{T} \to \mathbf{Setoid}$ lying over the identity natural transformation on $\mathcal{M}$.

Definition 4.6.4. Let $F : \mathcal{M} \to \mathcal{N}$ be a morphism of algebras. The kernel of F is the congruence $\ker_F : \mathcal{T} \to \mathbf{Setoid}$ over $\mathcal{M}$ defined by:

$$((x_1 \sim x_2) \in \ker_F) \iff F(x_1) = F(x_2)$$

for every $X \in \mathcal{T}$ and

$$((y_1 \sim y_2) \in \ker_F) \iff F(y_1) = F(y_2)$$

for every $(X \vdash Y\ \mathsf{type}) \in \mathcal{T}$. ⌟

Definition 4.6.5. A **quotient** of a congruence $\widetilde{\mathcal{M}}$ is an algebra $\mathbf{Q}$ along with a morphism $\mathbf{q} : \mathcal{M} \to \mathbf{Q}$ such that $\widetilde{\mathcal{M}} \subseteq \ker_{\mathbf{q}}$, and satisfying the following universal property: for every other morphism $F : \mathcal{M} \to \mathcal{N}$ such that $\widetilde{\mathcal{M}} \subseteq \ker_F$, there is a unique morphism $\widetilde{F} : \mathbf{Q} \to \mathcal{N}$ such that $F = \widetilde{F} \circ \mathbf{q}$. ⌟

Proposition 4.6.6. *A quotient of a congruence $\widetilde{\mathcal{M}}$ is exactly a coequalizer of*

$$(\mathcal{P} \circ \widetilde{\mathcal{M}}) \longrightarrow \mathcal{M} \times \mathcal{M} \rightrightarrows \mathcal{M},$$

where $\mathcal{P} : \mathbf{Setoid} \to \mathbf{Set}$ is the Σ-CwF pseudo-morphism that sends $\widetilde{X}$ to

$$(x_1 : X) \times (x_2 : X) \times ((x_1 \sim x_2) \in \widetilde{X}),$$

and $(\mathcal{P} \circ \widetilde{\mathcal{M}})$ is strictified using proposition 3.3.17.

Proof. It follows from the observation that for any morphism $F : \mathcal{M} \to \mathcal{N}$, we have $\widetilde{\mathcal{M}} \subseteq \ker_F$ if and only if $(\mathcal{P} \circ \widetilde{\mathcal{M}}) \rightrightarrows \mathcal{M} \xrightarrow{F} \mathcal{N}$ commutes. □

In particular, since $\mathbf{Alg}_{\mathcal{T}}$ is cocomplete, quotients always exist.

Definition 4.6.7. A dependent setoid $\widetilde{Y}$ over a base setoid $\widetilde{X}$ is **fibrant** if for every $(x_1 \sim x_2) \in \widetilde{X}$ and $y_1 \in Y(x_1)$, there exists some $y_2 \in Y(x_2)$ such that $(y_1 \sim y_2) \in \widetilde{Y}$. ⌟

The fibrant dependent setoid determine the types of a subCwF $\mathbf{Setoid}_{\mathsf{fib}}$ of the CwF **Setoid** of setoids.

Proposition 4.6.8. *The quotient functor $Q : \mathbf{Setoid}_{\mathsf{fib}} \to \mathbf{Set}$ extends to a pseudo-morphism of Σ-CwFs.*

Proof. It suffices to show that Q preserves the terminal object and context extensions up to isomorphism. The preservation of the terminal object is immediate. The preservation of context extensions boils down to the fact that Q preserves pullback squares when one of the maps is a fibration. Take a pullback square

$$\begin{array}{ccc} B & \longrightarrow & Y \\ \downarrow & \lrcorner & \downarrow p \\ A & \xrightarrow{f} & X, \end{array}$$

in **Setoid**, such that the map $p : Y \to X$ is a fibration.

There is a canonical map $g : Q(B) \to Q(A) \times_{Q(x)} Q(Y)$, specified by $g([(a, y)]) = ([a], [y])$. Our goal is to prove that it is an isomorphism.

We define the inverse map $g^{-1} : Q(A) \times_{Q(x)} Q(Y) \to Q(B)$ using the universal properties of the quotients $Q(A)$ and $Q(B)$. Take a pair $([a], [y])$ in $Q(A) \times_{Q(x)} Q(Y)$. We have $[f(a)] = [p(y)]$, i.e. $f(a) = q(y)$ in the setoid X. Since $p : Y \twoheadrightarrow X$ is a fibration, there merely exists an element $y' : Y$ such that $p(y') = f(a)$ and $y \sim y'$. We then merely have an element $[(a, y')]$ of $Q(B)$. Thanks to the definition of $Q(B)$ as a quotient, this element does not depend on the choices of a, y and y' in their equivalence classes. Thus this determines a map $g^{-1} : Q(A) \times_{Q(x)} Q(Y) \to Q(B)$.

It is then straightforward to prove that g^{-1} is an inverse of g, using the universal properties of the quotients $Q(A)$, $Q(Y)$ and $Q(B)$. □

It can be shown that the fibrancy assumption is necessary.

Proposition 4.6.9. *The quotient functor Q : **Setoid** $\to$ **Set** does not extend to a pseudo-morphism of Σ-CwFs.*

Proof. We show that Q does not preserve pullbacks. Consider the cospan $\{\bullet \quad \bullet\} \to \{\bullet \sim \bullet\} \leftarrow \{\bullet \quad \bullet\}$ in **Setoid**. The quotient of its pullback has two elements, but the pullback of the quotients has four elements. □

Definition 4.6.10. A congruence $\widetilde{\mathcal{M}}$ is said to be **fibrant** if it factors through the inclusion $\mathbf{Setoid}_{\mathsf{fib}} \hookrightarrow \mathbf{Setoid}$, or equivalently if for every type $(X \vdash Y\ \mathsf{type}) \in \mathcal{T}$, the dependent setoid $\widetilde{\mathcal{M}}_Y$ is fibrant over $\widetilde{\mathcal{M}}_X$. ⌟

Lemma 4.6.11. *Given a fibrant congruence $\widetilde{\mathcal{M}}$, the strictification of the pseudo-morphism $(Q \circ \widetilde{\mathcal{M}}) : \mathcal{T} \to$ **Set** is a quotient $\mathbf{Q}(\widetilde{\mathcal{M}})$ of $\widetilde{\mathcal{M}}$. The quotient inclusion $\mathbf{q} : \mathcal{M} \to \mathbf{Q}(\widetilde{\mathcal{M}})$ is defined pointwise as the quotient inclusion $\mathbf{q}_X : \mathcal{M}_X \to Q(\widetilde{\mathcal{M}}_X)$.*

Proof. We verify the universal property of the quotient.

Let $F : \mathcal{M} \to \mathcal{N}$ be a morphism such that $\widetilde{\mathcal{M}} \subseteq \mathcal{N}$. By the universal property of the quotient set $Q(\widetilde{\mathcal{M}}_X)$, there is a unique map $\widetilde{F}_X : Q(\widetilde{\mathcal{M}}_X) \to \mathcal{N}_X$ such that $\widetilde{F}_X \circ \mathbf{q}_X = F_x$. The universal properties of the quotient sets implies that this assignment is natural in X, thus defining a morphism $\widetilde{F} : \mathbf{Q}(\widetilde{\mathcal{M}}) \to \mathcal{N}$ such that $\widetilde{F} \circ \mathbf{q} = F$. The unicity of the components $\widetilde{F}_X$ imply that this morphism is the unique factorization of F through $\mathbf{q}$.

Thus $\mathbf{Q}(\widetilde{\mathcal{M}})$ is a quotient of $\widetilde{\mathcal{M}}$. □

Corollary 4.6.12. *If $\widetilde{\mathcal{M}}$ is a fibrant congruence, then the quotient inclusion $\mathbf{q} : \mathcal{M} \to \mathbf{Q}(\widetilde{\mathcal{M}})$ is a trivial fibration in $\mathbf{Alg}_{\mathcal{T}}$, and $\widetilde{\mathcal{M}} = \ker_{\mathbf{q}}$.*

Proof. By lemma 4.6.11, every component of $\mathbf{q}$ is a quotient inclusion at the level of sets, and is therefore surjective. □

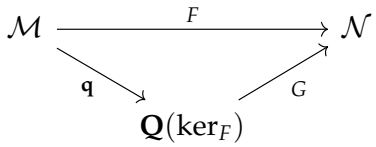
Conversely, we can prove that all trivial fibrations arise in this way.

Proposition 4.6.13. *If $F : \mathcal{M} \to \mathcal{N}$ is a trivial fibration in $\mathbf{Alg}_{\mathcal{T}}$, then $\ker_F$ is a fibrant congruence and $\mathcal{N}$ is the quotient of $\ker_F$.*

Proof. We first prove the fibrancy of $\ker_F$.

Take a type $(X \vdash Y \text{ type}) \in \mathcal{T}$. Take $(x_1, x_2 : X) \in \mathcal{M}$ such that $F(x_1) = F(x_2)$ and take some $(y_1 : Y(x_1)) \in \mathcal{M}$. Since $(F(y_1) : Y(F(x_2))) \in \mathcal{N}$ and F is a trivial fibration, there exists some $(y_2 : Y(x_2)) \in \mathcal{M}$ such that $F(y_2) = F(y_1)$. This proves the fibrancy of $\ker_F$.

We can then consider the quotient $\mathbf{Q}(\ker_F)$. Its universal property implies that the map F factors through the quotient inclusion.



Since both F and $\mathbf{q}$ are trivial fibrations, the map G is also a trivial fibration, surjective on every sort. To prove that it is an isomorphism, it suffices to prove that it is injective on every sort.

Take a type $(X \vdash Y \text{ type}) \in \mathcal{T}$. Take elements $(x : X) \in \mathbf{Q}(\ker_F)$ and $(y_1, y_2 : Y(x) \in \mathbf{Q}(\ker_F)$ such that $G(y_1) = G(y_2)$. Since $\mathbf{q}$ is a trivial fibration, there exists lifts $(x' : X) \in \mathcal{M}$, $(y_1', y_2' : Y(x')) \in \mathcal{M}$ such that $\mathbf{q}(x') = x$, $\mathbf{q}(y_1') = y_1$ and $\mathbf{q}(y_2') = y_2$. We then have $F(y_1') = F(y_2')$, i.e. $(y_1' \sim y_2') \in \ker_F$. Therefore, $\mathbf{q}(y_1') = \mathbf{q}(y_2')$, i.e. $y_1 = y_2$, as needed. □

Note that without assuming the axiom of choice, it is not possible to use this result to construct a trivial fibration that is split. If the goal is to construct a split trivial fibration, it may be useful to consider **Setoid**-valued algebras without taking their quotients.

Chapter 5

Second-order generalized algebraic theories

We now introduce second-order generalized algebraic theories (SOGATs) and their functorial semantics in $\mathbf{CwF}_{\Sigma,\Pi_{\mathsf{rep}}}$.

The semantics of SOGATs are more complicated than the semantics of GATs. An algebra is defined not as a morphism from $\mathcal{T}$ to **Set**, but as a category $\mathcal{C}$ (with a terminal object) together with a morphism from $\mathcal{T}$ to $\mathbf{Psh}(\mathcal{C})$ with its $(\Sigma,\Pi_{\mathsf{rep}})$-CwF structure.

This is fine so far, but when defining e.g. algebra morphisms between algebras with underlying categories $\mathcal{C}$ and $\mathcal{D}$, one needs to have a $(\Sigma,\Pi_{\mathsf{rep}})$-CwF displayed over $\mathbf{Psh}(\mathcal{C}) \times \mathbf{Psh}(\mathcal{D})$. The $(\Sigma,\Pi_{\mathsf{rep}})$-CwF structure of $\mathbf{Psh}(\mathcal{C})$ is already not trivial, so we would rather avoid having to define ad-hoc and complex $(\Sigma,\Pi_{\mathsf{rep}})$-CwF structures.

We will observe that all the $(\Sigma,\Pi_{\mathsf{rep}})$-CwF constructions we need actually arise from a single general construction. For any Σ-CwF $\mathcal{E}$ with an internal category $\mathbb{C}$, there is an external $(\Sigma,\Pi_{\mathsf{rep}})$-CwF $\mathsf{psh}_{\mathcal{E}}(\mathbb{C})$ of *internal presheaves over* $\mathbb{C}$. When $\mathcal{E} = \mathbf{Set}$, we recover the construction of $\mathbf{Psh}(\mathcal{C})$ for an external category $\mathcal{C}$. When $\mathcal{E} = \mathbf{Func}$, an internal category in $\mathcal{E}$ is exactly an external functor F, and $\mathsf{psh}_{\mathbf{Func}}(F)$ is exactly the $(\Sigma,\Pi_{\mathsf{rep}})$-CwF we need to define algebras morphisms lying over F. Using other Σ-CwFs, such as **Fam** and **Sect**, we obtain other components of the semantics of SOGATs.

Importantly, this construction $\mathsf{psh}_{\_}(-)$, which turns Σ-CwFs into $(\Sigma,\Pi_{\mathsf{rep}})$-CwFs, has a left adjoint. The left adjoint turns $(\Sigma,\Pi_{\mathsf{rep}})$-CwFs into Σ-CwFs, and restricts to a functor that turns SOGATs into GATs. We use this translation from SOGATs to GATs to define the semantics of SOGATs from the semantics of GATs. Past or future semantic constructions at the level of GATs can then be applied to SOGATs.

5.1 Definition and examples

Definition 5.1.1. A **second-order generalized algebraic theory**, or **SOGAT**, is a type-presented $(\Sigma,\Pi_{\mathsf{rep}})$-CwF.

A morphism between SOGATs is a morphism in $\mathbf{CwF}_{\Sigma,\Pi_{\mathsf{rep}}}$. ⌟

SOGATs are defined just like GATs, but it is now possible to use the Π-types to express second-order operations.

Example 5.1.2. The SOGAT $\mathcal{T}_{\mathsf{Fam}_{\mathsf{rep}}}$ of families with representable elements is presented

by the following signature:

$$\begin{aligned}&\mathsf{ty} : \mathsf{Ty},\\ &\mathsf{tm} : \mathsf{ty} \to \mathsf{Ty}_{\mathsf{rep}}\end{aligned}$$

This means that the $(\Sigma, \Pi_{\mathsf{rep}})$-CwF $\mathcal{T}_{\mathsf{Fam}_{\mathsf{rep}}}$ is defined as the following extension of $\mathbf{0}_{\Sigma,\Pi_{\mathsf{rep}}}$:

$$\mathcal{T}_{\mathsf{Fam}_{\mathsf{rep}}} = \mathbf{0}_{\Sigma,\Pi_{\mathsf{rep}}}[\vdash \underline{\mathsf{ty}}\ \mathsf{type}][A : \underline{\mathsf{ty}} \vdash \underline{\mathsf{tm}}(A)\ \mathsf{type}_{\mathsf{rep}}].$$ ⌟

Example 5.1.3. The SOGAT $\mathcal{T}_\Sigma$ of families with Σ-types is the extension of $\mathcal{T}_{\mathsf{Fam}_{\mathsf{rep}}}$ by the following signature:

$$\begin{aligned}&\mathbf{1} : \mathsf{ty},\\ &\mathsf{tt} : \mathsf{tm}(\mathbf{1}),\\ &(x, y : \mathsf{tm}(\mathbf{1})) \to (x = y),\\ &\Sigma : (A : \mathsf{ty})(B : \mathsf{tm}(A) \to \mathsf{ty}) \to \mathsf{ty},\\ &\mathsf{pair} : (a : \mathsf{tm}(A))(b : \mathsf{tm}(B(a))) \to \mathsf{tm}(\Sigma(A, B)),\\ &\mathsf{fst} : \mathsf{tm}(\Sigma(A, B)) \to \mathsf{tm}(A),\\ &\mathsf{snd} : (p : \mathsf{tm}(\Sigma(A, B))) \to \mathsf{tm}(B(\mathsf{fst}(p))),\\ &\mathsf{fst}(\mathsf{pair}(a, b)) = a,\\ &\mathsf{snd}(\mathsf{pair}(a, b)) = b,\\ &\mathsf{pair}(\mathsf{fst}(p), \mathsf{snd}(p)) = p.\end{aligned}$$ ⌟

The SOGAT $\mathcal{T}_\Sigma$ includes a second-order operation: Σ, whose second-argument is a dependent type. When presenting $\mathcal{T}_\Sigma$ as an iterated extension of $\mathcal{T}_{\mathsf{Fam}_{\mathsf{rep}}}$, we need to use Π-types to specify the generator $\underline{\Sigma}$:

$$\mathcal{T}_\Sigma = \mathcal{T}_{\mathsf{Fam}_{\mathsf{rep}}}[A : \underline{\mathsf{ty}}, B : \Pi(\underline{\mathsf{tm}}(A), \underline{\mathsf{ty}}) \vdash \underline{\Sigma}(A, B) : \underline{\mathsf{ty}}][\cdots].$$

Writing this Π-type is only allowed because the sort $\underline{\mathsf{tm}}$ is representable.

Example 5.1.4. The SOGAT $\mathcal{T}_{\Sigma,\Pi}$ of families with Σ- and Π-types is the extension of $\mathcal{T}_\Sigma$ by the following signature:

$$\begin{aligned}&\Pi : (A : \mathsf{ty})(B : \mathsf{tm}(A) \to \mathsf{ty}) \to \mathsf{ty},\\ &\mathsf{app} : \mathsf{tm}(\Pi(A, B)) \cong ((a : \mathsf{tm}(A)) \to \mathsf{tm}(B(a))) : \mathsf{lam}.\end{aligned}$$ ⌟

Example 5.1.5. There is a SOGAT $\mathcal{T}_{\Sigma,\Pi_{\mathsf{rep}}}$ of $(\Sigma, \Pi_{\mathsf{rep}})$-families. It is an extension of $\mathcal{T}_\Sigma$.

There is a morphism $\mathcal{T}_{\Sigma,\Pi_{\mathsf{rep}}} \to \mathcal{T}_{\Sigma,\Pi}$ which sends $\mathsf{ty}_{\mathsf{rep}}$ to ty, corresponding to the fact that any (Σ, Π)-family is a $(\Sigma, \Pi_{\mathsf{rep}})$-family in which all types are representable. ⌟

Example 5.1.6. We can define a SOGAT $\mathcal{T}_{\mathsf{MLTT}}$ whose models are models of (one variant) of Martin-Löf type theory. Here we consider MLTT with a hierarchy of universes closed under Σ-, Π-, $\mathbf{1}$-, Bool-, Empty-, Id- and W- types.

We first define a SOGAT $\mathcal{T}_{\mathcal{U}}$ of cumulative families with universes: its components are indexed by natural numbers corresponding to universe levels.

$$\begin{aligned}&\mathsf{ty}_n : \mathsf{Ty},\\ &\mathsf{tm}_n : \mathsf{ty}_n \to \mathsf{Ty}_{\mathsf{rep}},\\ &\mathcal{U}_n : \mathsf{ty}_{n+1},\\ &\mathsf{El}_n : \mathsf{tm}_{n+1}(\mathcal{U}_n) \cong \mathsf{ty}_n,\\ &\mathsf{Lift}_n : \mathsf{ty}_n \to \mathsf{ty}_{n+1},\\ &\mathsf{lift}_n : (A : \mathsf{ty}_n) \to \mathsf{tm}_n(A) \cong \mathsf{tm}_{n+1}(\mathsf{Lift}_n(A)).\end{aligned}$$

We can then define an extension $\mathcal{T}_{\mathcal{U},\Pi}$ of $\mathcal{T}_{\mathcal{U}}$:

$$\begin{aligned}&\Pi_n : (A : \mathsf{ty}_n)(B : \mathsf{tm}_n(A) \to \mathsf{ty}_n) \to \mathsf{ty}_n,\\ &\mathsf{app}_n : \mathsf{tm}_n(\Pi(A,B)) \cong ((a : \mathsf{tm}_n(A)) \to \mathsf{tm}_n(B(a))) : \mathsf{lam}_n.\end{aligned}$$

Note that for any $n < \omega$, there is $\mathcal{T}_{\mathsf{Fam}_{\mathsf{rep}}} \to \mathcal{T}_{\mathcal{U}}$ selecting the family $(\mathsf{ty}_n, \mathsf{tm}_n)$. The SOGAT $\mathcal{T}_{\mathcal{U},\Pi}$ could be defined as a certain colimit that glues together ω-many copies of $\mathcal{T}_{\Pi}$ over the ω-many underlying families of $\mathcal{T}_{\mathcal{U}}$. The same method could be used to define the other negative type structures (Σ- and **1**- types) in a way that is independent of the universe levels; only the universes and the lifting operations change the universe levels. This does not work directly for positive types (inductive types), because their elimination principle needs to target arbitrary universe levels.

The other type-formers are also specified by SOGATs extending $\mathcal{T}_{\mathcal{U}}$ (expect for W-types, which are defined as an extension of $\mathcal{T}_{\mathcal{U},\Pi}$).

Empty**-types** We define $\mathcal{T}_{\mathcal{U},\mathsf{Empty}}$ as the extension of $\mathcal{T}_{\mathcal{U}}$ specified by the following signature:

$$\begin{aligned}&\mathsf{Empty}_n : \mathsf{ty}_n,\\ &\mathsf{absurd} : (P : \mathsf{tm}(\mathsf{Empty}_n) \to \mathsf{ty}_m) \to (x : \mathsf{tm}(\mathsf{Empty}_n)) \to \mathsf{tm}_m(P(x)).\end{aligned}$$

Bool**-types**

$$\begin{aligned}&\mathsf{Bool}_n : \mathsf{ty}_n,\\ &\mathsf{true} : \mathsf{tm}(\mathsf{Bool}_n),\\ &\mathsf{false} : \mathsf{tm}(\mathsf{Bool}_n),\\ &\mathsf{elimBool} : (P : \mathsf{tm}(\mathsf{Bool}_n) \to \mathsf{ty}_m) \to \mathsf{tm}(P(\mathsf{true})) \to \mathsf{tm}(P(\mathsf{false}))\\ &\quad \to (b : \mathsf{tm}(\mathsf{Bool})) \to \mathsf{tm}(P(b)),\\ &- : \mathsf{elimBool}(P,t,f,\mathsf{true}) = t,\\ &- : \mathsf{elimBool}(P,t,f,\mathsf{false}) = f.\end{aligned}$$

Id**-types**

$$\begin{aligned}&\mathsf{Id}_n : (A : \mathsf{ty}_n)(x,y : \mathsf{tm}(A)) \to \mathsf{ty}_n,\\ &\mathsf{refl} : (A : \mathsf{ty}_n)(x : \mathsf{tm}(A)) \to \mathsf{tm}(\mathsf{Id}_n(A,x,x)),\\ &\mathsf{J} : (A : \mathsf{ty}_n)(x : \mathsf{tm}(A))(P : (y : \mathsf{tm}(A))(p : \mathsf{tm}(\mathsf{Id}(A,x,y))) \to \mathsf{ty}_m)\\ &\quad \to (d : \mathsf{tm}(P(x,\mathsf{refl}(A,x))))(y : \mathsf{tm}(A))(p : \mathsf{tm}(\mathsf{Id}(A,x,y)))\\ &\quad \to \mathsf{tm}(P(y,p)),\\ &- : \mathsf{J}(A,x,P,d,x,\mathsf{refl}(A,x)) = d.\end{aligned}$$

W**-types** The specification of W-types stands out among other inductive types, because they require the presence of Π-types.

$$\begin{aligned}&W_n : (A : \mathsf{ty}_n)(B : \mathsf{tm}(A) \to \mathsf{ty}_n) \to \mathsf{ty}_n,\\ &\mathsf{sup} : (a : \mathsf{tm}(A))(f : \mathsf{tm}(\Pi(B(a), \lambda - \mapsto W(A,B)))) \to \mathsf{tm}(W_n(A,B)),\\ &\mathsf{elimW} : (P : \mathsf{tm}(W(A,B)) \to \mathsf{ty}_m)\\ &\quad \to (r : (a : \mathsf{tm}(A))(f : \mathsf{tm}(\Pi(B(a), \lambda - \mapsto W(A,B))))\\ &\qquad (f' : \mathsf{tm}(\Pi(B(a), \lambda b \mapsto P(\mathsf{app}(f,b))))) \to \mathsf{tm}(P(\mathsf{sup}(a,f))))\\ &\quad \to (w : \mathsf{tm}(W(A,B))) \to \mathsf{tm}(P(w)),\\ &- : \mathsf{elimW}(P,r,\mathsf{sup}(a,f)) = r(a,f,\mathsf{lam}(\lambda b \mapsto \mathsf{elimW}(P,r,\mathsf{app}(f,b)))).\end{aligned}$$

Finally, $\mathcal{T}_{\mathsf{MLTT}}$ is defined as a colimit of $\mathcal{T}_{\mathcal{U},\Pi}$, $\mathcal{T}_{\mathcal{U},\Sigma}$, $\mathcal{T}_{\mathcal{U},\mathbf{1}}$, $\mathcal{T}_{\mathcal{U},\mathsf{Empty}}$, $\mathcal{T}_{\mathcal{U},\mathsf{Bool}}$, $\mathcal{T}_{\mathcal{U},\mathsf{Id}}$ and $\mathcal{T}_{\mathcal{U},\Pi,W}$ over $\mathcal{T}_{\mathcal{U}}$. Equivalently, $\mathcal{T}_{\mathsf{MLTT}}$ is presented by a signature that is the concatenation of the signatures of the subtheories. ⌟

5.2 Functorial semantics and reduction to GATs

Definition 5.2.1. An **algebra** of a SOGAT $\mathcal{T}$ consists of a category $\mathcal{C}$, with a terminal object $1_{\mathcal{C}}$, and a strict $(\Sigma, \Pi_{\mathsf{rep}})$-CwF morphism $\mathbb{C} : \mathcal{T} \to \mathbf{Psh}(\mathcal{C})$. ⌟

Given an algebra $(\mathcal{C}, \mathbb{C})$, we will use the following notations. For an object $X \in \mathcal{T}$, we write $(\Gamma \vdash x : X) \in \mathcal{C}$ to indicate that x is an element of $\mathbb{C}_X(\Gamma)$. For a type $(X \vdash Y\ \mathsf{type}) \in \mathcal{T}$ and $(\Gamma \vdash x : X) \in \mathcal{C}$, we write $(\Gamma \vdash y : Y(x)) \in \mathcal{C}$ to indicate that y is an element of $\mathbb{C}_Y(\Gamma, x)$.

We defer the definition of morphism until later. We want to exhibit the algebras of a SOGAT $\mathcal{T}$ as the algebras of a (first-order) GAT $\mathcal{T}^{\mathsf{fo}}$. For example, we know that the category of algebras of $\mathcal{T}_{\mathbf{Fam}_{\mathsf{rep}}}$ is **CwF**, which is also the category of algebras of a GAT $\mathcal{T}_{\mathbf{CwF}}$. Among other things, this will automatically provide a complete and cocomplete category of algebras of any SOGAT.

We write $\mathcal{T}_{\mathbf{Cat}_1}$ for the GAT of categories with a chosen terminal object. We want to find a GAT $\mathcal{T}^{\mathsf{fo}}$ extending $\mathcal{T}_{\mathbf{Cat}_1}$, i.e. with a strict Σ-CwF morphism $\mathcal{T}_{\mathbf{Cat}_1} \to \mathcal{T}^{\mathsf{fo}}$. Take any category (with a terminal object) $\mathcal{C} \in \mathbf{Cat}_1$, which can equivalently be seen as $[\mathcal{C}] : \mathcal{T}_{\mathbf{Cat}_1} \to \mathbf{Set}$. Then $\mathcal{T}^{\mathsf{fo}}$ should have the property that Σ-CwF morphisms $\mathcal{T}^{\mathsf{fo}} \to \mathbf{Set}$ extending $[\mathcal{C}] : \mathcal{T}_{\mathbf{Cat}_1} \to \mathbf{Set}$ correspond to $(\Sigma, \Pi_{\mathsf{rep}})$-CwF morphisms $\mathcal{T} \to \mathbf{Psh}(\mathcal{C})$.

Looking at morphisms $\mathcal{T}^{\mathsf{fo}} \to \mathbf{Set}$ is not enough to characterize $\mathcal{T}^{\mathsf{fo}}$; we should look at morphisms $\mathcal{T}^{\mathsf{fo}} \to \mathcal{E}$ for an arbitrary Σ-CwF $\mathcal{E}$. Take a Σ-CwF $\mathcal{E}$ and an internal category with a terminal object $\mathbb{C}$ in $\mathcal{E}$, corresponding to a Σ-CwF morphism $[\mathbb{C}] : \mathcal{T}_{\mathbf{Cat}_1} \to \mathcal{E}$. We want a correspondence between Σ-CwF morphisms $\mathcal{T}^{\mathsf{fo}} \to \mathcal{E}$ extending $\mathbb{C}$ and $(\Sigma, \Pi_{\mathsf{rep}})$-CwF morphisms $\mathcal{T} \to \mathsf{psh}_{\mathcal{E}}(\mathbb{C})$, where $\mathsf{psh}_{\mathcal{E}}(\mathbb{C})$ generalizes the construction of presheaves to internal categories. For any external category $\mathcal{C}$, we should have $\mathsf{psh}_{\mathbf{Set}}(\mathcal{C}) = \mathbf{Psh}(\mathcal{C})$. The construction of $\mathsf{psh}_{\mathcal{E}}(\mathbb{C})$ is detailed below.

Construction 5.2.2. Given any Σ-CwF $\mathcal{E}$ with base types together with an internal category with a terminal object $\mathbb{C}$ (which can be identified with a strict Σ-CwF morphism $[\mathbb{C}] : \mathcal{T}_{\mathbf{Cat}_1} \to \mathcal{E}$), we construct a $(\Sigma, \Pi_{\mathsf{rep}})$-CwF with base types $\mathsf{psh}_{\mathcal{E}}(\mathbb{C})$ of **internal presheaves over** $\mathbb{C}$.

- An object of $\mathsf{psh}_{\mathcal{E}}(\mathbb{C})$ is an internal presheaf Γ over $\mathbb{C}$ in $\mathcal{E}$, consisting of:
  - a type $(x : \mathbb{C}.\mathsf{Ob} \vdash \Gamma(x)\ \mathsf{type}) \in \mathcal{E}$;
  - a term $(f : \mathbb{C}.\mathsf{Hom}(x, y), \gamma : \Gamma(y) \vdash \gamma[f] : \Gamma(x)) \in \mathcal{E}$;
  - an equality $(x : \mathbb{C}.\mathsf{Ob}, \gamma : \Gamma(x) \vdash \gamma[\mathsf{id}] = \gamma) \in \mathcal{E}$;
  - an equality $(f : \mathbb{C}.\mathsf{Hom}(y, z), g : \mathbb{C}.\mathsf{Hom}(x, y), \gamma : \Gamma(z) \vdash \gamma[f][g] = \gamma[f \circ g]) \in \mathcal{E}$.
- A morphism of $\mathsf{psh}_{\mathcal{E}}(\mathbb{C})$ from Γ to Δ consists of:
  - a term $(x : \mathbb{C}.\mathsf{Ob}, \gamma : \Gamma(x) \vdash \delta(x, \gamma) : \Delta(x)) \in \mathcal{E}$;
  - an equality $(f : \mathbb{C}.\mathsf{Hom}(x, y), \gamma : \Gamma(y) \vdash \delta(y, \gamma)[f] = \delta(x, \gamma[f])) \in \mathcal{E}$.
- A type of $\mathsf{psh}_{\mathcal{E}}(\mathbb{C})$ over Γ is an internal dependent presheaf over Γ, consisting of:

  - a type $(x : \mathbb{C}.\mathsf{Ob}, \gamma : \Gamma(x) \vdash A(x,\gamma)\ \mathsf{type}) \in \mathcal{E}$;
  - a term $(f : \mathbb{C}.\mathsf{Hom}(x,y), a : A(y,\gamma) \vdash a[f] : A(x,\gamma[f])) \in \mathcal{E}$;
  - an equality $(x : \mathbb{C}.\mathsf{Ob}, a : A(x,\gamma) \vdash a[\mathsf{id}] = a) \in \mathcal{E}$;
  - an equality $(f : \mathbb{C}.\mathsf{Hom}(y,z), g : \mathbb{C}.\mathsf{Hom}(x,y), a : A(z,\gamma) \vdash a[f][g] = a[f \circ g]) \in \mathcal{E}$.

- A representable type of $\mathsf{psh}_{\mathcal{E}}(\mathbb{C})$ over Γ is a type A over Γ together with:
  - a term $(x : \mathbb{C}.\mathsf{Ob}, \gamma : \Gamma(x) \vdash (x.A[\gamma]) : \mathbb{C}.\mathsf{Ob}) \in \mathcal{E}$;
  - a term $(f : \mathbb{C}.\mathsf{Hom}(x,y), a : A(x,\gamma[f]) \vdash \langle f, a\rangle : \mathbb{C}.\mathsf{Hom}(x, y.A[\gamma])) \in \mathcal{E}$;
  - a term $(x : \mathbb{C}.\mathsf{Ob}, \gamma : \Gamma(x) \vdash \boldsymbol{p}_{A[\gamma]} : \mathbb{C}.\mathsf{Hom}(x.A[\gamma], x)) \in \mathcal{E}$;
  - a term $(x : \mathbb{C}.\mathsf{Ob}, \gamma : \Gamma(x) \vdash \boldsymbol{q}_{A[\gamma]} : A(\gamma[\boldsymbol{p}_{A[\gamma]}])) \in \mathcal{E}$;
  - the following equalities in $\mathcal{E}$:
$$\begin{aligned}&\langle f, a\rangle \circ g = \langle f \circ g, a[g]\rangle,\\ &\boldsymbol{p}_{A[\gamma]} \circ \langle f, a\rangle = f,\\ &\boldsymbol{q}_{A[\gamma]}[\langle f, a\rangle] = a,\end{aligned}$$
    over suitable contexts.

- A term of $\mathsf{psh}_{\mathcal{E}}(\mathbb{C})$ of type A over Γ consists of:
  - a term $(x : \mathbb{C}.\mathsf{Ob}, \gamma : \Gamma(x) \vdash a(x,\gamma) : A(x,\gamma)) \in \mathcal{E}$;
  - an equality $(f : \mathbb{C}.\mathsf{Hom}(x,y), \gamma : \Gamma(y) \vdash a(y,\gamma)[f] = a(x,\gamma[f])) \in \mathcal{E}$.

- A base type of $\mathsf{psh}_{\mathcal{E}}(\mathbb{C})$ is a type A such that $(x : \mathbb{C}.\mathsf{Ob}, \gamma : \Gamma(x) \vdash A(x,\gamma)\ \mathsf{type}) \in \mathcal{E}$ is a base type of $\mathcal{E}$.

- The empty context, context extensions, $\mathbf{1}$-type and Σ-types are defined in the same way as in the $(\Sigma, \Pi_{\mathsf{rep}})$-CwF structure of external presheaf categories.

- The definition of the first-order Π-type makes use of the representability of the domain:
$$\begin{aligned}&\Pi_{\mathsf{rep}}(\Gamma, A, B)(x,\gamma) = B(x.A[\gamma], (\gamma[\boldsymbol{p}_{A[\gamma]}], \boldsymbol{q}_{A[\gamma]})),\\ &\Pi_{\mathsf{rep}}(\Gamma, A, B)(f : y \to x, \gamma, b) = b[f^+ : y.A[\gamma[f]] \to x.A[\gamma]],\\ &\mathsf{app}(\Gamma, A, B, b, a)(x,\gamma) = b(x,\gamma)[\langle a(x,\gamma)\rangle : x \to x.A[\gamma]],\\ &\mathsf{lam}(\Gamma, A, B, b)(x,\gamma) = b(x.A[\gamma], (\gamma[\boldsymbol{p}_{A[\gamma]}], \boldsymbol{q}_{A[\gamma]})).\end{aligned}$$
  Note that the usual definition of Π-types in a presheaf category quantifies on objects and morphisms, and therefore cannot be expressed internally to an arbitrary Σ-CwF. ⌟

Lemma 5.2.3. *The above construction determines a functor* $\mathsf{psh} : (\mathcal{T}_{\mathbf{Cat}_1} / \mathbf{CwF}_{\Sigma}^{\mathsf{base}}) \to \mathbf{CwF}_{\Sigma,\Pi_{\mathsf{rep}}}^{\mathsf{base}}$. *Moreover, this functor is induced by a GAT morphism from the GAT of* $(\Sigma, \Pi_{\mathsf{rep}})$*-CwFs with base types into the GAT of* Σ*-CwFs with base types with an internal category with a terminal object.*

Proof. The existence of the GAT morphism follows from the fact that all components of $\mathsf{psh}_{\mathcal{E}}(\mathbb{C})$ have been defined as finite collections of types, terms and term equalities from $\mathcal{E}$. Note that we need to include term equalities in the GAT of Σ-CwFs with an internal category. □

Corollary 5.2.4. *The functor* $\mathsf{psh} : (\mathcal{T}_{\mathbf{Cat}_1} / \mathbf{CwF}_\Sigma^{\mathsf{base}}) \to \mathbf{CwF}_{\Sigma,\Pi_{\mathsf{rep}}}^{\mathsf{base}}$ *has a left adjoint.* □

Because $(\mathcal{T}_{\mathbf{Cat}_1} / \mathbf{CwF}_\Sigma^{\mathsf{typres}})$ and $\mathbf{CwF}_{\Sigma,\Pi_{\mathsf{rep}}}^{\mathsf{typres}}$ are coreflective subcategories of $(\mathcal{T}_{\mathbf{Cat}_1} / \mathbf{CwF}_\Sigma^{\mathsf{base}})$ and $\mathbf{CwF}_{\Sigma,\Pi_{\mathsf{rep}}}^{\mathsf{base}}$, the adjunction restricts to type-presented CwFs.

Corollary 5.2.5. *The composite functor*

$$(\mathcal{T}_{\mathbf{Cat}_1} / \mathbf{CwF}_\Sigma^{\mathsf{typres}}) \xrightarrow{\mathsf{psh}} \mathbf{CwF}_{\Sigma,\Pi_{\mathsf{rep}}}^{\mathsf{base}} \xrightarrow{\mathsf{typres}(-)} \mathbf{CwF}_{\Sigma,\Pi_{\mathsf{rep}}}^{\mathsf{typres}}$$

has a left adjoint. □

For any SOGAT $\mathcal{T}$, we write $\mathcal{T}^{\mathsf{fo}}$ for the GAT obtained by applying the left adjoint of corollary 5.2.5 to $\mathcal{T}$. The assignment $\mathcal{T} \mapsto \mathcal{T}^{\mathsf{fo}}$ is functorial, even though morphisms of SOGATs and GATs are not required to preserve the base types.

Definition 5.2.6. The Σ-CwF $\mathbf{Alg}_\mathcal{T}$ of algebras of a SOGAT $\mathcal{T}$ is the Σ-CwF $\mathbf{Alg}_{\mathcal{T}^{\mathsf{fo}}}$ of algebras of the GAT $\mathcal{T}^{\mathsf{fo}}$. ⌟

We can use the universal property of $\mathcal{T}^{\mathsf{fo}}$ to compute characterizations of the components of $\mathbf{Alg}_\mathcal{T}$ using $(\Sigma, \Pi_{\mathsf{rep}})$-CwF morphisms out of $\mathcal{T}$.

- A $\mathcal{T}$-algebra consists of a category $\mathcal{C}$ with a terminal object and a $(\Sigma, \Pi_{\mathsf{rep}})$-CwF morphism $\mathbb{C} : \mathcal{T} \to \mathsf{psh}_{\mathbf{Set}}(\mathcal{C})$.
- A $\mathcal{T}$-algebra morphism from $(\mathcal{C}, \mathbb{C})$ to $(\mathcal{D}, \mathbb{D})$ consists of a functor $F : \mathcal{C} \to \mathcal{D}$ preserving the terminal object and a dependent $(\Sigma, \Pi_{\mathsf{rep}})$-CwF morphism $\mathbb{F} : \mathcal{T} \to \mathsf{psh}_{\mathbf{Func}}(F)[\mathbb{C}, \mathbb{D}]$.

  Since the data of a displayed term of **Func** is propositional, only the displayed types of **Func** carry proof-relevant data, and all equalities between displayed terms (or morphisms) of **Func** hold automatically. This simplifies the understanding of $\mathsf{psh}_{\mathbf{Func}}$.

  A dependent $(\Sigma, \Pi_{\mathsf{rep}})$-CwF morphism $\mathbb{F} : \mathcal{T} \to \mathsf{psh}_{\mathbf{Func}}(F)[\mathbb{C}, \mathbb{D}]$ consists of:

  - For every object $X \in \mathcal{T}$, an internal presheaf $\mathbb{F}_X$ in **Func** over the presheaves $\mathbb{C}_X$ and $\mathbb{D}_X$. This can be decomposed into a dependent function (a type of **Func**)

    $$\mathbb{F}_X : (\Gamma : \mathcal{C}.\mathsf{Ob}) \to \mathbb{C}_X(\Gamma) \to \mathbb{D}_X(F(\Gamma))$$

    and equalities (a term of **Func**)

    $$(f : \mathcal{C}.\mathsf{Hom}(\Delta, \Gamma)) \to \mathbb{F}_X(\Gamma, x)[F(f)] = \mathbb{F}_X(\Delta, x[f]).$$

    In other words, $\mathbb{F}_X$ is a natural transformation from $\mathbb{C}_X$ to $F^*(\mathbb{D}_X)$.
  - For every type $(X \vdash Y\ \mathsf{type}) \in \mathcal{T}$, and internal dependent presheaf $\mathbb{F}_Y$ over $\mathbb{F}_X$ in **Func**, over the dependent presheaves $\mathbb{C}_Y$ and $\mathbb{D}_Y$. This can be understood as a dependent natural transformation from $\mathbb{C}_Y$ to $\mathbb{D}_Y$ over the natural transformation $\mathbb{F}_X$.
  - The rest of the data is propositional.
- A displayed $\mathcal{T}$-algebra over $(\mathcal{C}, \mathbb{C})$ consists of a displayed category $\mathcal{D}$ over $\mathcal{C}$ (with a displayed terminal object) and a dependent $(\Sigma, \Pi_{\mathsf{rep}})$-CwF morphism $\mathbb{D} : \mathcal{T} \to \mathsf{psh}_{\mathbf{Fam}}(\mathcal{D})[\mathbb{C}]$.

- A section of a displayed $\mathcal{T}$-algebra $(\mathcal{D}, \mathbb{D})$ over $(\mathcal{C}, \mathbb{D})$ consists of a section S of $\mathcal{D}$ and a dependent $(\Sigma, \Pi_{\mathsf{rep}})$-CwF morphism $\mathbb{D} : \mathcal{T} \to \mathsf{psh}_{\mathbf{Sect}}(S)[\mathbb{C}, \mathbb{D}]$.

Example 5.2.7. $\mathcal{T}_{\mathsf{Fam}_{\mathsf{rep}}}$-algebras are exactly CwFs. Indeed, by the universal property of $\mathcal{T}_{\mathsf{Fam}_{\mathsf{rep}}}$, a $(\Sigma, \Pi_{\mathsf{rep}})$-CwF morphism $\mathbb{C} : \mathcal{T}_{\mathsf{Fam}_{\mathsf{rep}}} \to \mathsf{psh}_{\mathbf{Set}}(\mathcal{C})$ is determined by a closed type $(\vdash \mathbb{C}(\mathsf{ty})\ \mathsf{type}) \in \mathsf{psh}_{\mathbf{Set}}(\mathcal{C})$ and a representable type $(A : \mathbb{C}(\mathsf{ty}) \vdash \mathbb{C}(\mathsf{tm})\ \mathsf{type}_{\mathsf{rep}}) \in \mathsf{psh}_{\mathbf{Set}}(\mathcal{C})$. ⌟

5.3 Contextual algebras

The category of CwFs has a coreflective subcategory of contextual CwFs, defined in section 3.1.2. In this section we generalize this to a notion of contextual algebras for any SOGAT.

Let $\mathcal{T}$ be a SOGAT. Given any sort $(\partial A \vdash A\ \mathsf{type}) \in \mathcal{T}$, we may consider the extension

$$I^A : \mathsf{Free}_{\mathcal{T}}(\underline{\Gamma} \vdash \underline{\sigma} : \partial A) \to \mathsf{Free}_{\mathcal{T}}(\underline{\Gamma} \vdash \underline{a} : A(\underline{\sigma})).$$

This is the generic extension of a $\mathcal{T}$-algebra by a new element of the sort A, dependent over an arbitrary context $\underline{\Gamma}$.

Given any algebra $\mathcal{C}$ and $(\Gamma \vdash \sigma : \partial S) \in \mathcal{C}$, we write $\mathcal{C}[\Gamma \vdash \underline{a} : A(\sigma)]$ for the extension of $\mathcal{C}$ by a new element of A over Γ. It can be computed as the pushout

$$\begin{array}{ccc} \mathsf{Free}_{\mathcal{T}}(\underline{\Gamma} \vdash \underline{\sigma} : \partial A) & \xrightarrow{\langle \Gamma, \sigma \rangle} & \mathcal{C} \\ \downarrow{\scriptstyle I^A} & & \downarrow \\ \mathsf{Free}_{\mathcal{T}}(\underline{\Gamma} \vdash \underline{a} : A(\underline{\sigma})) & \xrightarrow{\langle \Gamma, \sigma, \underline{a} \rangle} & \mathcal{C}[\Gamma \vdash \underline{a} : A(\sigma)]. \end{array}$$

We write $I_{\mathcal{T}}$ for the set of all extensions $\{I^A \mid (\partial A \vdash A\ \mathsf{type}) \in \mathcal{T}\}$, and $I^{\mathsf{base}}_{\mathcal{T}}$ for the subset of $I_{\mathcal{T}}$ spanned by extensions corresponding to the base types. For example, $I^{\mathsf{base}}_{\mathcal{T}_{\mathbf{Fam}_{\mathsf{rep}}}}$ consists of the CwF morphisms

$$I^{\mathsf{ty}} : \mathsf{Free}_{\mathbf{CwF}}(\underline{\Gamma} \vdash) \to \mathsf{Free}_{\mathbf{CwF}}(\underline{\Gamma} \vdash \underline{A}\ \mathsf{type}),$$
$$I^{\mathsf{tm}} : \mathsf{Free}_{\mathbf{CwF}}(\underline{\Gamma} \vdash \underline{A}\ \mathsf{type}) \to \mathsf{Free}_{\mathbf{CwF}}(\underline{\Gamma} \vdash \underline{a} : \underline{A})$$

that we have previously encountered.

We have previously defined such sets for GATs, see section 4.5. In particular we have sets $I_{\mathcal{T}^{\mathsf{fo}}}$ and $I^{\mathsf{base}}_{\mathcal{T}^{\mathsf{fo}}}$ of extensions for $\mathcal{T}^{\mathsf{fo}}$. We have inclusions $I_{\mathcal{T}} \subseteq I_{\mathcal{T}^{\mathsf{fo}}}$ and $I^{\mathsf{base}}_{\mathcal{T}} \subseteq I^{\mathsf{base}}_{\mathcal{T}^{\mathsf{fo}}}$. Indeed, the extension I^A for a sort $(\partial A \vdash A\ \mathsf{type}) \in \mathcal{T}$ coincides with the extension associated to the sort $(\Gamma : \mathsf{Ob}, \sigma : \partial A(\Gamma) \vdash A\ \mathsf{type}) \in \mathcal{T}^{\mathsf{fo}}$. The only maps that are not in the image of $I^{\mathsf{base}}_{\mathcal{T}} \subseteq I^{\mathsf{base}}_{\mathcal{T}^{\mathsf{fo}}}$ are the extensions corresponding to the sorts of the theory of categories, i.e. the generic extension of a $\mathcal{T}$-algebra by a new context, morphism, or equality between morphisms.

Definition 5.3.1. A $\mathcal{T}$-algebra morphism is said to be **contextual isomorphism** if it is a right map for the orthogonal factorization system generated by $I^{\mathsf{base}}_{\mathcal{T}}$. ⌟

The maps in the left class of that orthogonal factorization system are called **left contextual** maps. A $\mathcal{T}$-algebra $\mathcal{C}$ is said to be contextual when the map $\mathbf{0}_{\mathcal{T}} \to \mathcal{C}$ is left contextual.

The intuition behind this definition is that an algebra is contextual when its components for the sorts in $(I^{\mathsf{base}}_{\mathcal{T}^{\mathsf{fo}}} \setminus I^{\mathsf{base}}_{\mathcal{T}})$ are freely generated, i.e. when the objects and

morphisms of its underlying category are freely generated (by the operations of $\mathcal{T}^{\mathsf{fo}}$, which include empty and extended contexts, empty and extended substitutions, etc.).

Any $\mathcal{T}$-algebra has a contextual core $\mathsf{cxl}(\mathcal{C})$, which is the unique (up to isomorphism) contextual $\mathcal{T}$-algebra equipped with a contextual isomorphism $\mathsf{cxl}(\mathcal{C}) \to \mathcal{C}$. The contextual core functor $\mathsf{cxl}(-)$ is right adjoint to the embedding from contextual $\mathcal{T}$-algebras into $\mathcal{T}$-algebras. This exhibits the category $\mathbf{Alg}_{\mathcal{T}}^{\mathsf{cxl}}$ of contextual $\mathcal{T}$-algebras as a coreflective subcategory of the category $\mathbf{Alg}_{\mathcal{T}}$ of $\mathcal{T}$-algebras.

Definition 5.3.2. For any functor $F : \mathcal{C} \to \mathcal{D}$, write $\mathsf{psh}^{\cong}_{\mathbf{Func}}(F)$ for the subCwF of $\mathsf{psh}_{\mathbf{Func}}(F)$ spanned by the representable types and types whose underlying maps

$$f_Y : (\Gamma : \mathcal{C}.\mathsf{Ob}) \to (x : X_1(\Gamma)) \to Y_1(\Gamma, x) \to Y_2(\Gamma, f_X(x))$$

are families of isomorphisms

$$f_Y : (\Gamma : \mathcal{C}.\mathsf{Ob}) \to (x : X_1(\Gamma)) \to Y_1(\Gamma, x) \cong Y_2(\Gamma, f_X(x)) \qquad \lrcorner$$

Proposition 5.3.3. *A $\mathcal{T}$-algebra morphism $\mathbb{F} : \mathcal{T} \to \mathsf{psh}_{\mathbf{Func}}(F)[\mathbb{C}, \mathbb{D}]$ is a contextual isomorphism if and only if it factors through*

$$\mathsf{psh}^{\cong}_{\mathbf{Func}}(F) \hookrightarrow \mathsf{psh}_{\mathbf{Func}}(F).$$

Proof. This is clear from unfolding the definitions. □

Proposition 5.3.4. *Let $\mathbb{C} : \mathcal{T} \to \mathsf{psh}_{\mathbf{Set}}(\mathcal{C})$ be a $\mathcal{T}$-algebra, and $F : \mathcal{A} \hookrightarrow \mathcal{C}$ a subcategory of $\mathcal{C}$. Assume that for every base representable sort $(X \vdash Y\ \mathsf{type}_{\mathsf{rep}}) \in \mathcal{T}$, $\Gamma \in \mathcal{A}$ and $x \in \mathbb{C}_X(\Gamma)$ we have a chosen object $\Gamma.Y(x) \in \mathcal{A}$ such that $F(\Gamma.Y(x)) = F(\Gamma).Y(x)$.*

Then $\mathcal{A}$ can be equipped with the structure of a $\mathcal{T}$-algebra and F can be equipped with the structure of a $\mathcal{T}$-algebra morphism, in such a way that F becomes a contextual isomorphism and that the chosen objects $\Gamma.Y(x)$ become the context extensions of $\mathcal{A}$.

Proof. Consider the following pullback in $(\Sigma, \Pi_{\mathsf{rep}})$-CwF:

$$\begin{array}{ccc} \mathcal{G} & \longrightarrow & \mathsf{psh}^{\cong}_{\mathbf{Func}}(F) \\ \downarrow{\scriptstyle \pi_2} & \lrcorner & \downarrow{\scriptstyle \pi_2} \\ \mathcal{T} & \xrightarrow{\mathbb{C}} & \mathsf{psh}_{\mathbf{Set}}(\mathcal{C}). \end{array}$$

We equip $\mathcal{G}$ with the following choice of base types:

- A base non-representable type of $\mathcal{G}$ is a type lying over a base non-representable type of $\mathcal{T}$ and such that the family of isomorphisms

  $$f_Y : (\Gamma : \mathcal{A}.\mathsf{Ob}) \to (x : X_1(\Gamma)) \to Y_1(\Gamma, x) \cong Y_2(\Gamma, f_X(x))$$

  is a family of identity maps

  $$f_Y : (\Gamma : \mathcal{A}.\mathsf{Ob}) \to (x : X_1(\Gamma)) \to Y_1(\Gamma, x) = Y_2(\Gamma, f_X(x)).$$

- A base representable type of $\mathcal{G}$ is a representable type lying over a base representable type $(X \vdash Y\ \mathsf{type}_{\mathsf{rep}})$ of $\mathcal{T}$, such that the family of isomorphisms

  $$f_Y : (\Gamma : \mathcal{A}.\mathsf{Ob}) \to (x : X_1(\Gamma)) \to Y_1(\Gamma, x) \cong Y_2(\Gamma, f_X(x))$$

is a family of identity maps

$$f_Y : (\Gamma : \mathcal{A}.\mathsf{Ob}) \to (x : X_1(\Gamma)) \to Y_1(\Gamma, x) = Y_2(\Gamma, f_X(x)),$$

and such that the underlying context extensions in $\mathcal{A}$ are given by the chosen objects $\Gamma.Y(x)$.

It can be shown that the actions of $\pi_2 : \mathcal{G} \to \mathcal{T}$ are bijective on base representable types, base non-representable types and terms. Thus, since $\mathcal{T}$ is type-presented, the projection $\pi_2 : \mathcal{G} \to \mathcal{T}$ admits a unique section in $\mathbf{CwF}^{\mathsf{base}}_{\Sigma,\Pi_{\mathsf{rep}}}$. This section can be decomposed into:

- A $\mathcal{T}$-algebra structure on $\mathcal{A}$, namely the composition
$$\mathcal{T} \to \mathcal{G} \to \mathsf{psh}^{\cong}_{\mathbf{Func}}(F) \xrightarrow{\pi_1} \mathsf{psh}_{\mathbf{Set}}(\mathcal{A}).$$
- A $\mathcal{T}$-algebra morphism structure on F which is a contextual isomorphism, given by the composition
$$\mathcal{T} \to \mathcal{G} \to \mathsf{psh}^{\cong}_{\mathbf{Func}}(F) \hookrightarrow \mathsf{psh}_{\mathbf{Func}}(F).$$
- Such that the context extensions of $\mathcal{A}$ coincide with the chosen object $\Gamma.Y(x)$, due to the fact that the section preserves the base representable types. □

5.3.1 Explicit description of the contextual core

The contextual core of a CwF can be explicitly constructed using telescopes, indexed by context length. For general SOGATs, we have to account for more general context shapes.

Definition 5.3.5. The **join** of two representable types $(\partial A \vdash A\ \mathsf{type}_{\mathsf{rep}})$ and $(\partial B \vdash B\ \mathsf{type}_{\mathsf{rep}})$ is a representable type $(\partial(A * B) \vdash (A * B)\ \mathsf{type}_{\mathsf{rep}})$, where

$$\partial(A * B) \triangleq (\sigma : \partial A) \times (\tau : A(\sigma) \to \partial B),$$
$$(A * B)(\sigma, \tau) \triangleq (a : A(\sigma)) \times (b : B(\tau(a))).$$

Note that the representability of A is used in the definition of $\partial(A * B)$. ⌟

Proposition 5.3.6. *The join operation is associative up to isomorphism, and the unit representable type* $(\mathbf{1} \vdash \mathbf{1}\ \mathsf{type}_{\mathsf{rep}})$ *is neutral up to isomorphism.*

Proof. We check associativity, and leave the (left and right) neutrality of $(\mathbf{1} \vdash \mathbf{1}\ \mathsf{type}_{\mathsf{rep}})$ to the reader. Take representable types $(\partial A \vdash A\ \mathsf{type}_{\mathsf{rep}})$, $(\partial C \vdash C\ \mathsf{type}_{\mathsf{rep}})$ and $(\partial C \vdash C\ \mathsf{type}_{\mathsf{rep}})$. Then we can compute

$$\partial((A * B) * C) = ((\sigma : \partial A) \times (\tau : A(\sigma) \to \partial B)) \times (\kappa : (a : A(\sigma)) \times B(\tau(a)) \to \partial C),$$
$$((A * B) * C)((\sigma, \tau), \kappa) = ((a : A(\sigma)) \times (b : B(\tau(a)))) \times C(\kappa(a, b)),$$
$$\partial(A * (B * C)) = (\sigma : \partial A) \times ((a : A(\sigma)) \to (\tau : \partial B)) \times (\kappa : B(\tau) \to \partial C),$$
$$(A * (B * C))(\sigma, (\tau, \kappa)) = (a : A(\sigma)) \times ((b : B(\tau(a))) \times C(\kappa(a, b))).$$

Using the associativity of dependent sums and the distributivity of dependent products over dependent sums, we see that the left-nested join $((A * B) * C)$ is isomorphic to the right-nested join $(A * (B * C))$. (Left-nested and right-nested joins are two normal forms for "polynomials": left-nested joins are sums of products, while right-nested joins are Horner normal forms.) □

Definition 5.3.7. A **context shape** is a list of base representable types. Any context shape $(A_1, \dots, A_n)$ has a join $(A_1 * \cdots * A_n)$, which we choose to be left-nested, since contexts are extended on the right. ⌟

Example 5.3.8. The only representable sort of $\mathcal{T}_{\mathbf{Fam}_{\mathsf{rep}}}$ is tm. Thus its context shapes are in bijection with natural numbers, corresponding to context lengths. Their joins are the representable types

$$
\begin{aligned}
&(\mathbf{1} \vdash \mathbf{1}\ \mathsf{type}_{\mathsf{rep}}),\\
&(A_1 : \mathsf{ty} \vdash \mathsf{tm}(A_1)\ \mathsf{type}_{\mathsf{rep}}),\\
&(A_1 : \mathsf{ty}, A_2 : \mathsf{tm}(A_1) \to \mathsf{ty} \vdash (a_1 : \mathsf{tm}(A_1)) \times (a_2 : \mathsf{tm}(A_2(a_1)))\ \mathsf{type}_{\mathsf{rep}}),\\
&\dots
\end{aligned}
$$

They correspond to the telescopes of any length. Indeed, the contexts of these representable types correspond to telescopes of types, and the representable types over these contexts then correspond to telescopes of terms. ⌟

Proposition 5.3.9. *The contextual core of a $\mathcal{T}$-algebra $\mathcal{C}$ admits the following description:*

- *An object of $\mathsf{cxl}(\mathcal{C})$ is a pair (S, σ), where S is a context shape, and $(1_{\mathcal{C}} \vdash \sigma : \partial S) \in \mathcal{C}$.*
- *The fully faithful functor $1_{\mathcal{C}}.- : \mathsf{cxl}(\mathcal{C}) \to \mathcal{C}$ sends (S, σ) to the object $1_{\mathcal{C}}.S(\sigma) \in \mathcal{C}$.*

Proof. We use proposition 5.3.4 to equip $\mathsf{cxl}(\mathcal{C})$ with the structure of a $\mathcal{T}$-algebra and $1_{\mathcal{C}}.-$ with the structure of a contextual isomorphism. We have to specify chosen context extensions in $\mathsf{cxl}(\mathcal{C})$: for any $(S, \sigma) \in \mathsf{cxl}(\mathcal{C})$, base representable type $(X \vdash Y\ \mathsf{type}_{\mathsf{rep}}) \in \mathcal{T}$ and element $(1_{\mathcal{C}}.S(\sigma) \vdash x : X) \in \mathcal{C}$, we pose $(S, \sigma).Y(x) \triangleq ((S * Y), (\sigma, x))$. We check that $1_{\mathcal{C}}.(S * Y)(\sigma, x) = 1_{\mathcal{C}}.(a : S(\sigma)).Y(x(a))$. Thus the assumptions of proposition 5.3.4 are satisfied.

It remains to prove that $\mathsf{cxl}(\mathcal{C})$ is contextual; this is analogous to the proof of proposition 3.1.6. Take any right contextual map $F : \mathcal{D} \to \mathcal{E}$ and any $\mathcal{T}$-algebra morphism $G : \mathsf{cxl}(\mathcal{C}) \to \mathcal{E}$. We need to define a factorization of G through F. We define a functor $H : \mathsf{cxl}(\mathcal{C}) \to \mathcal{D}$ by induction on the structure of context shapes.

$$
\begin{aligned}
&H(1) = 1_{\mathcal{D}},\\
&H((S * A)(\sigma, \tau)) = H(S(\sigma)).A(F^{-1}(G(\tau))),\\
&H(\langle\rangle : \Gamma \to 1) = \langle\rangle,\\
&H(\langle f, a\rangle : \Gamma \to (S * A)(\sigma, \tau)) = \left\langle H(f), F^{-1}(G(a))\right\rangle.
\end{aligned}
$$

We can prove that the underlying functor of G factors as H followed by F.

We then consider the $(\Sigma, \Pi_{\mathsf{rep}})$-CwF morphism

$$\mathsf{psh}_{\mathbf{Func}}(H) \times \mathsf{psh}_{\mathbf{Func}}(F) \to \mathsf{psh}_{\mathbf{Func}}(G) \times \mathsf{psh}_{\mathbf{Func}}(F)$$

induced by the Σ-CwF morphism $(\mathbf{Func}[C, D] \times \mathbf{Func}[D, E]) \to (\mathbf{Func}[C, E] \times \mathbf{Func}[D, E])$ that sends (h, f) to $(f \circ h, f)$.

We can restrict it to a $(\Sigma, \Pi_{\mathsf{rep}})$-CwF morphism

$$\mathsf{psh}_{\mathbf{Func}}(H) \times \mathsf{psh}^{\cong}_{\mathbf{Func}}(F) \to \mathsf{psh}_{\mathbf{Func}}(G) \times \mathsf{psh}^{\cong}_{\mathbf{Func}}(F).$$

We can show that its actions on types, representable types and terms are all bijective. Using the fact that $\mathcal{T}$ is type-presented, we obtain a factorization of

$$\langle G, F\rangle : \mathsf{psh}_{\mathbf{Func}}(G) \times \mathsf{psh}^{\cong}_{\mathbf{Func}}(F)$$

through

$$\mathsf{psh}_{\mathbf{Func}}(H) \times \mathsf{psh}^{\cong}_{\mathbf{Func}}(F) \to \mathsf{psh}_{\mathbf{Func}}(G) \times \mathsf{psh}^{\cong}_{\mathbf{Func}}(F),$$

witnessing the fact that H extends to a morphism of $\mathcal{T}$-algebra that is a factorization of G through F, as needed. □

5.3.2 The GAT of contextual algebras

In this section, we describe a GAT $\mathcal{T}^{\mathsf{cxl}}$ such that its category of algebras is equivalent to the category of contextual algebras of $\mathcal{T}$. Uemura (2019) has proven that for any representable map category $\mathcal{T}$, the category of finitely continuous functors $\mathcal{T} \to \mathsf{Set}$ is equivalent to the $(2,1)$-category of democratic models of $\mathcal{T}$. Transposed to our setting, for any SOGAT $\mathcal{T}$, the GAT $\mathcal{T}^{\mathsf{cxl}}$ should be obtained obtained from $\mathcal{T}$ by forgetting its representable types. Because a GAT has to be type-presented, we also need to ensure that $\mathcal{T}^{\mathsf{cxl}}$ is type-presented.

One application of this description of $\mathcal{T}^{\mathsf{cxl}}$ will be a description of the contextual core in functorial semantics: the contextual core of a $\mathcal{T}$-algebra

$$\mathbb{C} : \mathcal{T} \to \mathsf{psh}_{\mathbf{Set}}(\mathcal{C})$$

will be the strictification of the composition

$$\mathcal{T}^{\mathsf{cxl}} \to \mathcal{T} \xrightarrow{\mathbb{C}} \mathsf{psh}_{\mathbf{Set}}(\mathcal{C}) \xrightarrow{\mathsf{eval}_1} \mathsf{Set},$$

where eval_1 evaluates a presheaf at the terminal object $1_{\mathcal{C}}$ of $\mathcal{C}$.

More generally, compositions of the form

$$\mathcal{T}^{\mathsf{cxl}} \xrightarrow{\lfloor - \rfloor} \mathcal{T} \xrightarrow{\mathbb{C}} \mathsf{psh}_{\mathcal{E}}(\mathcal{C}) \xrightarrow{\mathsf{eval}_1} \mathcal{E}$$

generalize the contextual core to $\mathcal{T}$-algebras valued in other Σ-CwFs.

Definition 5.3.10. A **base monomial type** of a SOGAT $\mathcal{T}$ is a pair (A, B), where A is a context shape and B is a base type. The corresponding type is the dependent product

$$(\sigma : \partial A, \tau : A(\sigma) \to \partial B \vdash ((a : A(\sigma)) \to B(\tau(a)))\ \mathsf{type}) \in \mathcal{T}.$$ ⌟

Definition 5.3.11. We equip $\mathcal{T}$ with a new $\mathbf{CwF}^{\mathsf{base}}_{\Sigma}$ structure, keeping the same underlying $\mathbf{CwF}_{\Sigma}$-structure, and using the base monomial types as base types. We write $\mathcal{T}^{\mathsf{cxl}}$ for the type-presented replacement of this Σ-CwF with base types. ⌟

Proposition 5.3.12. *The Σ-CwF map $\lfloor - \rfloor : \mathcal{T}^{\mathsf{cxl}} \to \mathcal{T}$ is fully faithful, bijective on terms, and split essentially surjective on objects and types.*

Proof. As $\mathcal{T}^{\mathsf{cxl}}$ is a contextual CwF, the morphism $\mathcal{T}^{\mathsf{cxl}} \to \mathcal{T}$ is fully faithful if and only if it is bijective on terms, and essentially surjective on objects if and only if it is essentially surjective on types. By definition, $\mathcal{T}^{\mathsf{cxl}}$ is bijective on base types and bijective on terms.

Since $\mathcal{T}$ is type-presented as a $(\Sigma, \Pi_{\mathsf{rep}})$-CwF, its types are obtained as the closure of the base types under the operations of a $(\Sigma, \Pi_{\mathsf{rep}})$-CwF. Because Π-types essentially

distribute over dependent sums, any type of $\mathcal{T}$ is isomorphic to a iterated dependent sum of iterated dependent products of base types. But we can show that any iterated dependent product of base types (a "monomial" type) can be written as the substitution of a base monomial type. This implies that the iterated dependent products of base types are in the image of $\mathcal{T}^{\mathsf{cxl}} \to \mathcal{T}$. Since $\mathcal{T}^{\mathsf{cxl}} \to \mathcal{T}$ preserves dependent sums, the iterated dependent sums of iterated dependent products of base types are also in the image of $\mathcal{T}^{\mathsf{cxl}} \to \mathcal{T}$. Thus the types of $\mathcal{T}$ are all in the essential image of $\mathcal{T}^{\mathsf{cxl}} \to \mathcal{T}$. □

We will write $\lceil - \rceil$ for chosen lifts of elements of $\mathcal{T}$, so that for example $\lfloor \lceil X \rceil \rfloor \cong X$ for $X \in \mathcal{T}$.

Construction 5.3.13. For any Σ-CwF $\mathcal{E}$ with an internal category $\mathbb{C}$ with a terminal object $1_{\mathbb{C}}$, we construct a Σ-CwF morphism

$$\mathsf{eval}_1 : \mathsf{psh}_{\mathcal{E}}(\mathbb{C}) \to \mathcal{E}.$$

Moreover this construction is induced by a natural transformation between the GAT morphisms $\mathcal{T}_{\mathbf{CwF}_\Sigma} \to \mathcal{T}_{\mathbf{CwF}_\Sigma \setminus \mathcal{T}_{\mathbf{Cat}_1}}$ that induce

$$(\mathcal{E}, \mathbb{C}) \mapsto \mathsf{psh}_{\mathcal{E}}(\mathbb{C})$$

and

$$(\mathcal{E}, \mathbb{C}) \mapsto \mathcal{E}.$$

In particular, it is natural and it also acts on displayed Σ-CwFs with an internal category. ⌟

Construction. The actions of eval_1 on objects, morphisms, type and terms just perform the evaluation of some components at $1_{\mathbb{C}}$. Because context extensions, $\mathbf{1}$ and Σ are defined pointwise in $\mathsf{psh}_{\mathcal{E}}(\mathbb{C})$, they are strictly preserved by evaluation at $1_{\mathbb{C}}$. □

Definition 5.3.14. We define a GAT morphism $\mathcal{T}^{\mathsf{cxl}} \to \mathcal{T}^{\mathsf{fo}}$ as the composition

$$\mathcal{T}^{\mathsf{cxl}} \xrightarrow{\lfloor - \rfloor} \mathcal{T} \xrightarrow{\eta} \mathsf{psh}_{\mathcal{T}^{\mathsf{fo}}}(\bullet) \xrightarrow{\mathsf{eval}_1} \mathcal{T}^{\mathsf{fo}}.$$

⌟

That GAT morphism induces an adjunction $L \dashv R$ between $\mathbf{Alg}_{\mathcal{T}^{\mathsf{cxl}}}$ and $\mathbf{Alg}_{\mathcal{T}}$.

Proposition 5.3.15. *The right adjoint $R : \mathbf{Alg}_{\mathcal{T}} \to \mathbf{Alg}_{\mathcal{T}^{\mathsf{cxl}}}$ admits the following explicit description: for a $\mathcal{T}$-algebra $\mathcal{C}$ classified by $\mathbb{C} : \mathcal{T} \to \mathsf{psh}_{\mathbf{Set}}(\mathcal{C})$, the $\mathcal{T}^{\mathsf{cxl}}$-algebra $R(\mathcal{C})$ is classified by*

$$\mathcal{T}^{\mathsf{cxl}} \xrightarrow{\lfloor - \rfloor} \mathcal{T} \xrightarrow{\mathbb{C}} \mathsf{psh}_{\mathbf{Set}}(\mathcal{C}) \xrightarrow{\mathsf{eval}_1} \mathbf{Set}.$$

Proposition 5.3.16. *By definition, $R(\mathcal{C})$ is classified by*

$$\mathcal{T}^{\mathsf{cxl}} \xrightarrow{\lfloor - \rfloor} \mathcal{T} \xrightarrow{\eta} \mathsf{psh}_{\mathcal{T}^{\mathsf{fo}}}(\bullet) \xrightarrow{\mathsf{eval}_1} \mathcal{T}^{\mathsf{fo}} \xrightarrow{[\mathcal{C}, \mathbb{C}]} \mathbf{Set}.$$

By naturality of eval_1*, we can rewrite this to*

$$\mathcal{T}^{\mathsf{cxl}} \xrightarrow{\lfloor - \rfloor} \mathcal{T} \xrightarrow{\eta} \mathsf{psh}_{\mathcal{T}^{\mathsf{fo}}}(\bullet) \xrightarrow{\mathsf{psh}([\mathcal{C}, \mathbb{C}])} \mathsf{psh}_{\mathbf{Set}}(\mathcal{C}) \xrightarrow{\mathsf{eval}_1} \mathbf{Set},$$

where $[\mathcal{C}, \mathbb{C}] : \mathcal{T}^{\mathsf{fo}} \to \mathbf{Set}$ is the classifier of $\mathcal{C}$ seen as a $\mathcal{T}^{\mathsf{fo}}$-algebra, and $\mathsf{psh}([\mathcal{C}, \mathbb{C}])$ is the action on morphisms of the functor $\mathsf{psh}(-)$.

Then $\mathsf{psh}([\mathcal{C},\mathbb{C}]) \circ \eta$ *is just the transpose of* $[\mathcal{C},\mathbb{C}]$ *in the adjunction* $(-)^{\mathsf{fo}} \dashv \mathsf{psh}(-)$*, i.e.* $\mathbb{C} : \mathcal{T} \to \mathsf{psh}_{\mathbf{Set}}(\mathcal{C})$.

Thus we can rewrite the above as

$$\mathcal{T}^{\mathsf{cxl}} \xrightarrow{\lfloor - \rfloor} \mathcal{T} \xrightarrow{\mathbb{C}} \mathsf{psh}_{\mathbf{Set}}(\mathcal{C}) \xrightarrow{\mathsf{eval}_1} \mathbf{Set},$$

as needed.

Proposition 5.3.17. *Let* $F : \mathcal{C} \to \mathcal{D}$ *be a morphism of* $\mathcal{T}$*-algebras. If* $\mathcal{C}$ *is contextual and* $R(F) : R(\mathcal{C}) \to R(\mathcal{D})$ *is an isomorphism, then* F *is a contextual isomorphism.*

Proof. Let $(X \vdash A\ \mathsf{type}) \in \mathcal{T}$ be a sort. Take an object of $\mathcal{C}$. Since $\mathcal{C}$ is contextual, this object can be written $1_{\mathcal{C}}.S(\sigma)$ for some $(\partial S \vdash S\ \mathsf{type}_{\mathsf{rep}}) \in \mathcal{T}$ and $(1_{\mathcal{C}} \vdash \sigma : \partial S) \in \mathcal{C}$. Also take an element $(1_{\mathcal{C}}.S(\sigma) \vdash x : X) \in \mathcal{C}$.

We want to prove that the action of F on elements of A over $1.S(\sigma)$ and x is an isomorphism. But the action of F on A at $1_{\mathcal{C}}.S(\sigma)$ is equivalently the action of F on

$$(\sigma : \partial S \vdash \Pi_{s:S(\sigma)} A(x[s])\ \mathsf{type})$$

at $1_{\mathcal{C}}$.

The action of F on a sort at $1_{\mathcal{C}}$ is exactly the action of $R(F)$ on that sort. Therefore, the action of F on elements of A is bijective. Because this holds for any sort, F is a contextual isomorphism, as needed. □

Proposition 5.3.18. *For every* $\mathcal{M} : \mathbf{Alg}_{\mathcal{T}^{\mathsf{cxl}}}$*, the unit* $\eta_{\mathcal{M}} : \mathcal{M} \to R(L(\mathcal{M}))$ *is an isomorphism and* $L(\mathcal{M})$ *is contextual.*

Proof. We prove that for every $\mathcal{M} : \mathbf{Alg}_{\mathcal{T}^{\mathsf{cxl}}}$, we can construct a universal arrow $\eta_{\mathcal{M}} : \mathcal{M} \to R(L_{\mathcal{M}})$, that this universal arrow is an isomorphism and that $L_{\mathcal{M}}$ is contextual.

Take some $\mathcal{M} : \mathbf{Alg}_{\mathcal{T}^{\mathsf{cxl}}}$. Thanks to proposition 5.3.12, we can interpret the following notation: For Γ a representable sort of $\mathcal{T}$ and A a sort over $\mathcal{T}$ over Γ, we write

$$(\gamma : \Gamma \vdash x(\gamma) : A(\gamma)) \in \mathcal{M}$$

to mean that $(x : \lceil \Pi(\Gamma, A) \rceil) \in \mathcal{M}$.

We define a category $L_{\mathcal{M}}$:

- An object of $L_{\mathcal{M}}$ is a pair (S, σ) where S is a context shape and $(\sigma : \lceil \partial S \rceil) \in \mathcal{M}$.
- A morphism from (T, τ) to (S, σ) is an element $(t : \lceil T \rceil(\tau) \vdash f(t) : \lceil S \rceil(\sigma)) \in \mathcal{M}$. Identities and compositions are defined in the expected way.

We equip $L_{\mathcal{M}}$ with the structure of a $\mathcal{T}$-algebra by constructing a pseudo-morphism $\mathbb{L} : \mathcal{T} \to \mathsf{psh}_{\mathbf{Set}}(L_{\mathcal{M}})$ (which we can then strictify).

- An element of $\mathbb{L}(X)$ at (S, σ) is an element $(\gamma : \lceil S \rceil(\sigma) \vdash x(\gamma) : \lceil X \rceil) \in \mathcal{M}$. The functorial action is precomposition with $(t : \lceil T \rceil(\tau) \vdash f(t) : \lceil S \rceil(\sigma)) \in \mathcal{M}$.
- The action of $\mathbb{L}$ on a morphism $(f : X \to Y) \in \mathcal{T}$ is postcomposition with $\lceil f \rceil : \lceil X \rceil \to \lceil Y \rceil$.
- An element of $\mathbb{L}(A)$ at (S, σ) over an element x of $\mathbb{L}(X)$ is an element $(\gamma : \lceil S \rceil(\sigma) \vdash a(\gamma) : \lceil A \rceil(x(\gamma))) \in \mathcal{M}$.

- The action of $\mathbb{L}$ on a term $(X \vdash a : A) \in \mathcal{T}$ is the dependent natural transformation that takes $(\gamma : \ulcorner S \urcorner(\sigma) \vdash x(\gamma) : \ulcorner X \urcorner) \in \mathcal{M}$ and constructs $(\gamma : \ulcorner S \urcorner(\sigma) \vdash \ulcorner a \urcorner(x(\gamma)) : \ulcorner A \urcorner(x(\gamma))) \in \mathcal{M}$.
- If A is a representable sort, then $\mathbb{L}(A)$ is (locally) representable in $\mathsf{psh}_{\mathbf{Set}}(L_{\mathcal{M}})$.
- The preservation of Π-types boils down to a bijective correspondence between elements

$$(\gamma : \ulcorner S \urcorner(\sigma), a : \ulcorner A \urcorner(x(\gamma)) \vdash b(\gamma, a) : \ulcorner B \urcorner(x(\gamma), a)) \in \mathcal{M}$$

  and

$$(\gamma : \ulcorner S \urcorner(\sigma) \vdash f(\gamma) : \ulcorner \Pi(A, B) \urcorner(x(\gamma))) \in \mathcal{M}.$$

It follows from the characterization of proposition 5.3.9 that $L_{\mathcal{M}}$ is contextual.

We then define a morphism $\eta_{\mathcal{M}} : \mathcal{M} \to R(L_{\mathcal{M}})$. It suffices to define a natural transformation from $\mathcal{M}$ to $\mathsf{eval}_1 \circ L_{\mathcal{M}} \circ \lfloor - \rfloor$ as functors from $\mathcal{T}^{\mathsf{cxl}}$ to **Set**.

Given $x : \mathcal{M}_X$, we need to construct $(\ulcorner 1 \urcorner \vdash \eta_{\mathcal{M}}(x) : \ulcorner \lfloor X \rfloor \urcorner) \in \mathcal{M}$, i.e. $(\eta_{\mathcal{M}}(x) : \ulcorner \Pi(\lfloor \ulcorner 1 \urcorner \rfloor, \ulcorner \lfloor X \rfloor \urcorner) \urcorner) \in \mathcal{M}$. But $\ulcorner \Pi(\lfloor \ulcorner 1 \urcorner \rfloor, \ulcorner \lfloor X \rfloor \urcorner) \urcorner$ is isomorphic to X, so we can simply transport x over that isomorphism. Naturality is straightforward. Because the components $\eta_{\mathcal{M}}$ are defined by transport over some isomorphisms, they are themselves isomorphisms; hence $\eta_{\mathcal{M}}$ is an isomorphism.

Finally, we can prove that the arrow $\eta_{\mathcal{M}}$ is universal. Given another arrow $\alpha : \mathcal{M} \to R(\mathcal{C})$, we have to prove that there exists a unique $F : L_{\mathcal{M}} \to \mathcal{C}$ such that $\alpha = R(F) \circ \eta_{\mathcal{M}}$. Because the components of $L_{\mathcal{M}}$ are defined using some components of $\mathcal{M}$, we can apply α to these components when defining F. We omit the details. □

Theorem 5.3.19. *The coreflective subcategory of* $\mathbf{Alg}_{\mathcal{T}}$ *determined by the adjunction* $(L \dashv R)$ *is the same as the coreflective subcategory of contextual* $\mathcal{T}$*-algebras.*

Proof. It suffices to prove that for any $\mathcal{T}$-algebra $\mathcal{C}$, the counit

$$\varepsilon : L(R(\mathcal{C})) \to \mathcal{C}$$

exhibits $L(R(\mathcal{C}))$ as the contextual core of $\mathcal{C}$.

We have already proven in proposition 5.3.18 that $L(R(\mathcal{C}))$ is contextual. Because the adjunction is coreflective, we know that $R(\varepsilon)$ is an isomorphism. But, by proposition 5.3.17, this means that ε is a contextual isomorphism. Thus $L(R(\mathcal{C}))$ is the contextual core of $\mathcal{C}$, as needed. □

Chapter 6

Relative induction principles

The goal of this chapter is to explain how to prove properties of the initial algebras of SOGATs, in particular properties of the syntax of type theory. Some of the content of this chapter is based on work which I did together with Ambrus Kaposi and Christian Sattler (2023). The relative induction principles presented in this thesis are generalized to arbitrary SOGATs, instead of a single type theory. I also give applications of the relative induction principles which were not present in the paper.

Because the initial algebra of a SOGAT $\mathcal{T}$ is the initial algebra of the GAT $\mathcal{T}^{\mathsf{fo}}$, one has an induction principle over the initial algebra $\mathbf{0}_{\mathcal{T}}$: any displayed $\mathcal{T}^{\mathsf{fo}}$-algebra over $\mathbf{0}_{\mathcal{T}}$ admits a section. Properties of the initial algebra can then be proven by induction, i.e. by obtaining sections into carefully constructed displayed $\mathcal{T}^{\mathsf{fo}}$-algebras. Directly constructing all components of these displayed $\mathcal{T}^{\mathsf{fo}}$-algebras is feasible but impractical.

Instead, we want to construct them by composing multiple smaller constructions. Having such constructions is comparable to having alternative induction principles, or to general methods that can be used to suitably strengthen induction hypotheses. Using induction principles requires providing "motives" (the induction predicates, which are the sorts of a displayed model) and "methods" (the operations of the displayed model), subject to some equations. For the syntax of type theory considered as the initial $\mathcal{T}^{\mathsf{fo}}$-algebra, these equations include the substitution laws: all methods must commute with substitution. Proving these substitution laws by hand is very tedious, so we want alternative motives and methods that don't visibly interact with substitutions.

We show how to prove metatheorems using "relative induction principles". This combines two ideas:

- A *sconing* construction can be used to prove properties of closed components of an initial algebra $\mathbf{0}_{\mathcal{T}}$. By closed components, we mean for instance the sets $\mathbf{0}_{\mathcal{T}}.\mathsf{Ty}(1)$ and $\mathbf{0}_{\mathcal{T}}.\mathsf{Tm}(1, A)$ of closed types and terms. The scone, or Sierpinski cone, of a category $\mathcal{C}$ with a terminal object is the displayed category $\mathbf{Scone}_{\mathcal{C}}$ over $\mathcal{C}$ whose displayed objects over Γ are families $\Gamma' : \mathcal{C}(1, \Gamma) \to \mathsf{Set}$.

  Suppose that we want to prove a property of closed types and terms. We may have the "motives" for induction on closed types and terms.

  $$P_{\mathsf{Ty}} : \mathbf{0}_{\mathcal{T}}.\mathsf{Ty}(1) \to \mathsf{Set},$$
  $$P_{\mathsf{Tm}} : \forall A \to P_{\mathsf{Ty}}(A) \to \mathbf{0}_{\mathcal{T}}.\mathsf{Tm}(1, A) \to \mathsf{Set},$$

  and may want to construct functions $S_{\mathsf{Ty}} : \forall A \to P_{\mathsf{Ty}}(A)$ and $S_{\mathsf{Tm}} : \forall a \to P_{\mathsf{Tm}}(S_{\mathsf{Ty}}(A), a)$. In order to use the initiality of the syntax, we need to strengthen

these motives to arbitrary types and terms. The sconing construction explains how to do this: we need to quantify over all closing substitutions $\gamma : \mathbf{0}_{\mathcal{T}}(1, \Gamma)$ and consider the closed types $A[\gamma]$ and closed terms $a[\gamma]$. The strengthened motives are as follows:

$$
\begin{aligned}
&P'_{\mathsf{Ty}} : (\Gamma' : \mathbf{0}_{\mathcal{T}}(1, \Gamma) \to \mathsf{Set}) \to (A : \mathbf{0}_{\mathcal{T}}.\mathsf{Ty}(\Gamma)) \to \mathsf{Set},\\
&P'_{\mathsf{Ty}}(\Gamma', A) = \forall(\gamma : \mathbf{0}_{\mathcal{T}}(1, \Gamma))(\gamma' : \Gamma'(\gamma)) \to P_{\mathsf{Ty}}(A[\gamma]),\\
&P'_{\mathsf{Tm}} : (\Gamma' : \mathbf{0}_{\mathcal{T}}(1, \Gamma) \to \mathsf{Set})(A' : P'_{\mathsf{Ty}}(\Gamma', A)) \to (a : \mathbf{0}_{\mathcal{T}}.\mathsf{Ty}(\Gamma, A)) \to \mathsf{Set},\\
&P'_{\mathsf{Ty}}(\Gamma', A', a) = \forall(\gamma : \mathbf{0}_{\mathcal{T}}(1, \Gamma))(\gamma' : \Gamma'(\gamma)) \to P_{\mathsf{Tm}}(A'(\gamma'), a[\gamma]).
\end{aligned}
$$

In general, the sconing construction equips $\mathbf{Scone}_{\mathbf{0}_{\mathcal{T}}}$ with the structure of a displayed $\mathcal{T}$-algebra over $\mathbf{0}_{\mathcal{T}}$, starting from the data of a displayed "higher-order model" over $\mathbf{0}_{\mathcal{T}}$. A displayed higher-order model can be seen as the motives and methods for induction over the closed components of $\mathbf{0}_{\mathcal{T}}$.

- When proving properties of the open components of an initial algebra $\mathcal{S}$, one often has access to a functor $F : \mathcal{C} \to \mathcal{S}$. This functor specifies that the result of the induction should be available in the image of F, naturally in $\mathcal{C}$. This functor is called "figure shape" by Sterling (2022). This is also closely related to the "worlds" of Twelf (Pfenning and Schürmann 1999) or the "schemas" of Beluga (Pientka 2010). For example, the functor $1_{\mathcal{S}} \to \mathcal{S}$ that selects the terminal context is used for canonicity; since canonicity is a statement about terms in the empty context. The functor $\mathbf{Ren}(\mathcal{S}) \to \mathcal{S}$ is used for normalization; due to the fact that normalization can be performed over arbitrary contexts, but normal forms and the overall normalization procedure is only stable under renamings.

  In this situation, we construct a new $\mathcal{T}$-algebra $\mathcal{S}_F$, which *relativizes* $\mathcal{S}$ with respect to F. It is defined in such a way that closed components of $\mathcal{S}_F$ correspond bijectively to open components of $\mathcal{S}$ over the image of F. At a first glance, this seems impossible. What makes this possible is that $\mathcal{S}_F$ is not defined as an external $\mathcal{T}$-algebra, but as an internal $\mathcal{T}$-algebra in $\mathbf{Psh}(\mathcal{C})$, i.e. an element of $\mathbf{Alg}_{\mathcal{T}}(\mathbf{Psh}(\mathcal{C}))$. By proposition 4.3.11, this corresponds to a functor $\mathcal{S}[F(-)] : \mathcal{C}^{\mathsf{op}} \to \mathbf{Alg}_{\mathcal{T}}$ (a "presheaf of $\mathcal{T}$-algebras"). For any $\Gamma \in \mathcal{C}$, the closed components of the $\mathcal{T}$-algebra $\mathcal{S}[F(\Gamma)]$ will coincide with the components of S over $F(\Gamma)$; we have a natural isomorphism

  $$\mathcal{S}[F(\Gamma)].\mathsf{Ty}(1) \cong \mathcal{S}.\mathsf{Ty}(F(\Gamma)).$$

  When working in the internal language to $\mathbf{Psh}(\mathcal{C})$, the quantification over Γ is implicit; the left hand side of the above natural isomorphism is just written $\mathcal{S}_F.\mathsf{Ty}(1)$, and $\mathcal{S}_F$ behaves like an ordinary $\mathcal{T}$-algebra.

  A relative induction principle for $\mathcal{S}$ with respect to F is then an induction principle for $\mathcal{S}_F$ (an universal property for $\mathcal{S}_F$ in the category of internal $\mathcal{T}$-algebras in $\mathbf{Psh}(\mathcal{C})$, or equivalently for $\mathcal{S}[F(-)]$ in the category of functors $\mathcal{C}^{\mathsf{op}} \to \mathbf{Alg}_{\mathcal{T}}$). There is also a version of the relative induction principle in the internal language of $\mathbf{Psh}(\mathcal{C})$. Once we have an induction principle for $\mathcal{S}_F$, we can prove properties of its closed components using sconing, internally to $\mathbf{Psh}(\mathcal{C})$. Because of how $\mathcal{S}_F$ is defined, this yields proofs of properties of open components of S.

6.1 Contextualization

SOGATs are related to both first-order and higher-order notions of theories. A SOGAT $\mathcal{T}$ can be turned to a first-order GAT either through the reduction of section 5.2, or by going through the forgetful functor $\mathbf{CwF}_{\Sigma,\Pi_{\mathsf{rep}}} \to \mathbf{CwF}_{\Sigma}$ (which gives, up to type-presented replacement, the GAT $\mathcal{T}^{\mathsf{cxl}}$ of contextual algebras). A SOGAT can also be seen as a higher-order theory by going though the left adjoint functor $\mathbf{CwF}_{\Sigma,\Pi_{\mathsf{rep}}} \to \mathbf{CwF}_{\Sigma,\Pi}$. This provides the following notion of higher-order model:

Definition 6.1.1. A **higher-order model** of $\mathcal{T}$ is a $(\Sigma, \Pi_{\mathsf{rep}})$-CwF morphism $\mathcal{T} \to \mathbf{Set}$, where the $(\Sigma, \Pi_{\mathsf{rep}})$-structure of **Set** is induced from its (Σ, Π)-structure. ⌟

We call them higher-order models rather than higher-order algebras, because they don't assemble into a category, or at least not into a category of algebras and homomorphisms; there is a notion of isomorphism of higher-order models, but no notion of directed morphism.

Example 6.1.2. A higher-order model of $\mathcal{T}_{\mathsf{Fam}_{\mathsf{rep}}}$ consists of a set $\mathsf{ty} : \mathsf{Set}$ and a family $\mathsf{tm} : \mathsf{ty} \to \mathsf{Set}$. Note that this can be seen as a universe (with tm as its decoding function), although it is not equipped with any additional structure.

A higher-order model of $\mathcal{T}_{\Sigma}$ is a family closed under **1**- and Σ- types, i.e. with

$$\begin{aligned}
&\mathbf{1} : \mathsf{ty},\\
&\mathsf{tm}(\mathbf{1}) \cong \{\star\},\\
&\Sigma : (A : \mathsf{ty})(B : \mathsf{tm}(A) \to \mathsf{ty}) \to \mathsf{ty},\\
&\mathsf{tm}(\Sigma(A,B)) \cong (a : \mathsf{tm}(A)) \times \mathsf{tm}(B(a)).
\end{aligned}$$ ⌟

Their main purpose is to serve as intermediate steps in the constructions of "first-order" algebras. The $\mathcal{T}$-algebras have an underlying category of contexts, while the higher-order models do not have any explicit notion of context. Because the following construction adds explicit contexts to a higher-order model, we call it the *contextualization* of a higher-order model. There actually are multiple variants of the contextualization: the *telescopic contextualization* produces a contextual algebra, whereas the **Set**-*contextualization* produces an algebra whose base category is the category of sets.

Remark 6.1.3. The name *contextualization* can be confusing, as it is not directly related to contextual CwFs and the operation of taking the contextual core. Perhaps a better name for this operation can be found in the future, but I have not found one yet, so I decided to stick to the name that was used in our paper (Bocquet, Kaposi, and Sattler 2023).

Candidates for alternative names include first-order-ification, algebraization. ⌟

Construction 6.1.4. Let $\mathbb{M} : \mathcal{T} \to \mathbf{Set}$ be a higher-order model. The **telescopic contextualization** of $\mathbb{M}$ is the contextual $\mathcal{T}$-algebra $\mathsf{Cxl}_{\mathsf{Tele}}(\mathbb{M})$ specified by the composite Σ-CwF morphism

$$\mathcal{T}^{\mathsf{cxl}} \xrightarrow{\sim} \mathcal{T} \xrightarrow{\mathbb{M}} \mathbf{Set}.$$ ⌟

Construction 6.1.5. Let $\mathbb{M} : \mathcal{T} \to \mathbf{Set}$ be a higher-order model. The **Set-contextualization** of $\mathbb{M}$ is the $\mathcal{T}$-algebra $\mathsf{Cxl}_{\mathbf{Set}}(\mathbb{M})$ specified by the strictification of the composite $(\Sigma, \Pi_{\mathsf{rep}})$-CwF pseudo-morphism

$$\mathcal{T} \xrightarrow{\mathbb{M}} \mathbf{Set} \xrightarrow{\mathsf{よ}} \mathbf{Psh}(\mathbf{Set}).$$ ⌟

Proposition 6.1.6. *The telescopic contextualization is the contextual core of the* **Set**-*contextualization:*

$$\mathsf{Cxl}_{\mathsf{Tele}}(\mathbb{M}) \cong \mathsf{cxl}(\mathsf{Cxl}_{\mathbf{Set}}(\mathbb{M})).$$

Proof. By theorem 5.3.19, the contextual core of $\mathsf{Cxl}_{\mathbf{Set}}(\mathbb{M})$ is classified by the strictification of the composite Σ-CwF morphism

$$\mathcal{T}^{\mathsf{cxl}} \xrightarrow{\sim} \mathcal{T} \xrightarrow{\mathbb{M}} \mathbf{Set} \xrightarrow{\text{よ}} \mathbf{Psh}(\mathbf{Set}) \xrightarrow{\mathsf{eval}_1} \mathbf{Set}.$$

But $\mathsf{eval}_1 \circ \text{よ} \cong \mathsf{Id}$, so this is the Σ-CwF morphism classifying the telescopic contextualization. □

The telescopic contextualization can be generalized by looking at

$$\mathcal{T}^{\mathsf{cxl}} \xrightarrow{\sim} \mathcal{T} \xrightarrow{\mathbb{M}} \mathcal{E} \xrightarrow{\mathcal{E}(1_{\mathcal{E}},-)} \mathbf{Set}$$

for any (Σ,Π)-CwF $\mathcal{E}$. We won't need to use this generalization; in cases where it could possibly be used, we will rather use Construction 6.1.5 in the internal language of $\mathcal{E}$. This assumes that the internal language of $\mathcal{E}$ is rich enough, typically $\mathcal{E}$ will be a presheaf topos. Note however that when $\mathcal{E} = \mathsf{psh}_{\mathbf{Set}}(\mathcal{C})$ and $\mathbb{M}$ is a $\mathcal{T}$-algebra, then the telescopic contextualization of $\mathbb{M}$ is the contextual core; even though the contextualization and the contextual core are different constructions, they are not completely unrelated.

6.1.1 Displayed contextualization

We are interested in generalizing the contextualization operation so as to generate displayed $\mathcal{T}$-algebras; the goal is to obtain displayed $\mathcal{T}$-algebras over the initial $\mathcal{T}$-algebra, so as to use its universal property. In particular, we need a displayed variant of the notion of higher-order model. Intuitively, if a higher-order model is a morphism $\mathcal{T} \to \mathbf{Set}$, a displayed higher-order model should be a morphism $\mathcal{T} \to \mathbf{Fam}$. However, we also need to consider the base model, which is determined by a functor $\mathbb{M} : \mathcal{T} \to \mathsf{psh}_{\mathbf{Set}}(\mathcal{M})$. Indeed, a displayed higher-order model is not displayed over a base higher-order model, but over a base (first-order) $\mathcal{T}$-algebra instead.

This leads to the consideration of the pullback

$$\begin{array}{ccccc} & & \mathbf{Fam}[\mathsf{eval}_1] & \longrightarrow & \mathbf{Fam} \\ & & \downarrow & \lrcorner & \downarrow \\ \mathcal{T} & \xrightarrow{\mathbb{M}} & \mathsf{psh}_{\mathbf{Set}}(\mathcal{M}) & \xrightarrow{\mathsf{eval}_1} & \mathbf{Set} \end{array}$$

The category **Fam** has a $(\Sigma,\Pi_{\mathsf{rep}})$-CwF structure, but it is not preserved by the pseudo-morphism eval_1, so the pullback is not automatically a $(\Sigma,\Pi_{\mathsf{rep}})$-CwF. However the fact that eval_1 preserves Π-types laxly is sufficient to equip the pullback with a $(\Sigma,\Pi_{\mathsf{rep}})$-CwF structure.

Construction 6.1.7. Let $\mathcal{C}$ be a $(\Sigma,\Pi_{\mathsf{rep}})$-CwF and $F : \mathcal{C} \to \mathbf{Set}$ be a pseudo-morphism of Σ-CwFs. Then we define a displayed $(\Sigma,\Pi_{\mathsf{rep}})$-CwF $\mathbf{Fam}[F]$ over $\mathcal{C}$.

- The underlying displayed category of $\mathbf{Fam}[F]$ is the restriction of **Fam** over F.
- A displayed type over $(\Gamma \vdash A\ \mathsf{type}) \in \mathcal{C}$ and $\Gamma' : F(\Gamma) \to \mathsf{Set}$ is a family

$$A' : \forall(\gamma : F(\Gamma))(\gamma' : \Gamma'(\gamma)) \to F(A)(\gamma) \to \mathsf{Set}.$$

  The displayed representable types are defined in the same way.

- A displayed term a' of type A' over $(\Gamma \vdash a : A) \in \mathcal{C}$ is a family

$$a' : \forall(\gamma : F(\Gamma))(\gamma' : \Gamma'(\gamma)) \to A'(\gamma, \gamma', F(a)(\gamma)).$$

- The displayed empty context is the family

$$\diamond' = \lambda\_ \mapsto \{\star\}.$$

  The extension of a displayed context Γ' by a displayed type A' is the family

$$(\Gamma'.A') = \lambda(\gamma, a) \mapsto (\gamma' : \Gamma'(\gamma)) \times A'(\gamma', a).$$

  Here the quantification over (γ, a) is justified by the fact that F preserves context extensions up to isomorphism.

- The displayed **1**-type is the family

$$\mathbf{1}' = \lambda\_\_ \mapsto \{\star\}.$$

  Given displayed types A' over a displayed context Γ' and B' over the displayed context $\Gamma'.A'$, the displayed Σ-type over $\Sigma(A, B)$ is the family

$$\Sigma'(A', B') = \lambda\gamma\gamma'(a, b) \mapsto (a' : A'(\gamma', a)) \times (b' : B'((\gamma', a'), b)).$$

  Here the quantification over (a, b) is justified by the fact that F preserves Σ-types up to isomorphism.

- Given a displayed representable type A' over a displayed context Γ' and a displayed type B' over the displayed context $\Gamma'.A'$, the displayed Π-type over $\Pi(A, B)$ is the family

$$\Pi'(A', B') = \lambda\gamma\gamma' f \mapsto (\forall(a : F(A)(\gamma))(a' : A'(\gamma', a)) \to B'((\gamma', a'), F(\mathsf{app})(\gamma, f, a))).$$

  Here $F(\mathsf{app}) : \forall\gamma \to F(\Pi(A, B))(\gamma) \to (a : F(A)(\gamma)) \to F(B)(\gamma, a)$ witnesses the lax preservation of Π-types by the pseudo-morphism F. ⌟

We will be focusing on the instance $\mathbf{Fam}[\mathsf{eval}_1]$ of this construction for the pseudo-morphism

$$\mathsf{eval}_1 : \mathsf{psh}_{\mathbf{Set}}(\mathcal{M}) \to \mathbf{Set}.$$

Definition 6.1.8. Let $\mathbb{M} : \mathcal{T} \to \mathsf{psh}_{\mathbf{Set}}(\mathcal{M})$ be a base $\mathcal{T}$-algebra.

A **displayed higher-order model** of $\mathcal{T}$ is a dependent $(\Sigma, \Pi_{\mathsf{rep}})$-CwF morphism

$$\mathcal{T} \to \mathbf{Fam}[\mathsf{eval}_1][\mathbb{M}].$$ ⌟

Let $\mathcal{M}^\bullet : \mathcal{T} \to \mathbf{Fam}[\mathsf{eval}_{1_\mathcal{M}}][\mathbb{M}]$ be a displayed higher-order model.

- For every object $X \in \mathcal{T}$, we have a family

$$\mathcal{M}^\bullet_X : \mathbb{M}_X(1_\mathcal{M}) \to \mathsf{Set}.$$

- For every type $(X \vdash A\ \mathsf{type}) \in \mathcal{T}$, we have a family

$$\mathcal{M}^\bullet_A : (x : \mathbb{M}_X(1_\mathcal{M})) \to \mathcal{M}^\bullet_X(x) \to \mathbb{M}_A(1_\mathcal{M}, x) \to \mathsf{Set}.$$

- In the case of Π-types, we have

$$\mathcal{M}^\bullet_{\Pi(A,B)}(x,x',f) = (\forall(a : \mathbb{M}_A(1_\mathcal{M}, x))\ (a' : \mathcal{M}^\bullet_A(x,x',a)) \to \mathcal{M}^\bullet_B((x,a),(x',a'),f(a)).$$

- For every term $(X \vdash a : A) \in \mathcal{T}$, we have a map

$$\mathcal{M}^\bullet_a : (x : \mathbb{M}_X(1_\mathcal{M}))(x' : \mathcal{M}^\bullet_X(x)) \to \mathcal{M}^\bullet_A(x,x',\mathbb{M}_a(1_\mathcal{M},x)).$$

Example 6.1.9. We can compute the notion of displayed higher-order model over the SOGAT $\mathcal{T}_\Sigma$ of a representable family with Σ-types. Let $\mathcal{M}$ be a $\mathcal{T}_\Sigma$-algebra. A displayed higher-order model over $\mathcal{M}$ consists of the following components:

$$\begin{aligned}
&\mathsf{ty}^\bullet : \mathcal{M}.\mathsf{Ty}(1_\mathcal{M}) \to \mathsf{Set}, && \text{(Displayed types)}\\
&\mathsf{tm}^\bullet : \forall A \to \mathsf{ty}^\bullet(A) \to \mathcal{M}.\mathsf{Tm}(1_\mathcal{M}, A) \to \mathsf{Set}, && \text{(Displayed terms)}\\
&\Sigma^\bullet : \forall A\ B \to (A^\bullet : \mathsf{ty}^\bullet(A)) \to (\forall a\ (a^\bullet : \mathsf{tm}^\bullet(A^\bullet, a)) \to \mathsf{ty}^\bullet(B[a])) \to \mathsf{ty}^\bullet(\Sigma(A,B)),\\
&\mathsf{pair}^\bullet : \forall A\ B\ A^\bullet\ B^\bullet\ a\ b\ (a^\bullet : \mathsf{tm}^\bullet(A^\bullet,a))\ (b^\bullet : \mathsf{tm}^\bullet(B^\bullet(a,a^\bullet),b))\\
&\quad \to \mathsf{tm}^\bullet(\Sigma^\bullet(A^\bullet,B^\bullet),\mathsf{pair}(a,b)),\\
&\mathsf{fst}^\bullet : \forall A\ B\ A^\bullet\ B^\bullet\ p\ (p^\bullet : \mathsf{tm}^\bullet(\Sigma^\bullet(A^\bullet,B^\bullet),p)) \to \mathsf{tm}^\bullet(A^\bullet,\mathsf{fst}(p)),\\
&\mathsf{snd}^\bullet : \forall A\ B\ A^\bullet\ B^\bullet\ p\ (p^\bullet : \mathsf{tm}^\bullet(\Sigma^\bullet(A^\bullet,B^\bullet),p)) \to \mathsf{tm}^\bullet(B^\bullet(\mathsf{fst}(p),\mathsf{fst}^\bullet(p)),\mathsf{snd}(p)),\\
&- : \mathsf{fst}^\bullet(\mathsf{pair}^\bullet(a^\bullet,b^\bullet)) = a^\bullet,\\
&- : \mathsf{snd}^\bullet(\mathsf{pair}^\bullet(a^\bullet,b^\bullet)) = b^\bullet,\\
&- : \mathsf{pair}^\bullet(\mathsf{fst}^\bullet(p^\bullet),\mathsf{snd}^\bullet(p^\bullet)) = p^\bullet.
\end{aligned}$$

Computing for example the specification of $\Sigma^\bullet$ involves interpreting

$$(A : \mathsf{ty}, B : \mathsf{tm}(A) \to \mathsf{ty} \vdash \Sigma(A,B) : \mathsf{ty}) \in \mathcal{T}_\Sigma$$

in $\mathbf{Fam}[\mathsf{eval}_{1_\mathcal{M}}][\mathbb{M}]$. In particular, the interpretation of the Π-type $\mathsf{tm}(A) \to \mathsf{ty}$ introduce quantification on $(a : \mathcal{M}.\mathsf{Tm}(1_\mathcal{M}, A))$ and $(a^\bullet : \mathsf{tm}^\bullet(A^\bullet, a))$. ⌟

Definition 6.1.10. Given a displayed higher-order model $\mathbb{M}^\bullet$, the **telescopic displayed contextualization** $\mathsf{Cxl}_{\mathsf{Tele}}(\mathbb{M}^\bullet)$ is the displayed contextual $\mathcal{T}$-algebra over the contextual core $\mathsf{cxl}(\mathcal{M})$ determined by the composite map

$$\mathcal{T}^{\mathsf{cxl}} \to \mathcal{T} \xrightarrow{\mathbb{M}^\bullet} \mathbf{Fam}[\mathsf{eval}_1] \to \mathbf{Fam}.$$ ⌟

The definition of the **Scone**-contextualization, which is the displayed variant of the **Set**-contextualization, is unfortunately more complicated than its non-displayed counterpart. In applications, it is possible to only use the telescopic displayed contextualization. For computations with concrete displayed higher-order models, it is however easier to compute the **Scone**-contextualization.

The following construction will play the same role as the Yoneda embedding $よ : \mathbf{Set} \to \mathbf{Psh}(\mathbf{Set})$ in the definition of the **Set**-contextualization.

Construction 6.1.11. We construct a $(\Sigma, \Pi_{\mathsf{rep}})$-CwF pseudo-morphism

$$\mathbb{Y} : \mathbf{Fam}[\mathsf{eval}_1] \to \mathsf{psh}_{\mathbf{Fam}}(\mathbf{Scone}_\mathcal{M})$$

that lies over $\mathsf{id} : \mathsf{psh}_{\mathbf{Set}}(\mathcal{M}) \to \mathsf{psh}_{\mathbf{Set}}(\mathcal{M})$. ⌟

Proof. When $\mathcal{M} = 1_{\mathbf{Cat}}$, the displayed category $\mathbf{Fam}[\mathsf{eval}_1]$ becomes **Set**, the displayed category $\mathsf{psh}_{\mathbf{Fam}}(\mathbf{Scone}_{\mathcal{M}})$ becomes $\mathbf{Psh}(\mathbf{Set})$, and the pseudo-morphism $\mathbb{Y}$ should reduce to the Yoneda embedding.

An object $X' : X(1_{\mathcal{M}}) \to \mathrm{Set}$ is sent to the following dependent presheaf (object of $\mathsf{psh}_{\mathbf{Fam}}(\mathbf{Scone}_{\mathcal{M}})$ over X)

$$\mathbb{Y}_{X'}(\Gamma, \Gamma', x : X(\Gamma)) = \forall(\gamma : \mathcal{M}(1, \Gamma))\ (\gamma' : \Gamma'(\gamma)) \to X'(x[\gamma]).$$

The action on a morphism $(f : \Delta \to \Gamma, f' : \forall \delta \to \Delta'(\delta) \to \Gamma'(f \circ \delta))$ is precomposition with f and f'.

A morphism $(\alpha : X \Rightarrow Y, \alpha' : X'(x) \to Y'(\alpha_{1_{\mathcal{M}}}(x)))$ is sent to the dependent natural transformation

$$\begin{aligned}
&\mathbb{Y}_{\alpha'}(\Gamma, \Gamma', x : X(\Gamma), x' : \mathbb{Y}_{X'}(\Gamma, \Gamma', x)) : \mathbb{Y}_{Y'}(\Gamma, \Gamma', \alpha_{1_{\mathcal{M}}}(x)),\\
&\mathbb{Y}_{\alpha'}(\Gamma, \Gamma', x : X(\Gamma), x' : \mathbb{Y}_{X'}(\Gamma, \Gamma', x))\\
&\quad = \lambda \gamma\ \gamma' \mapsto \alpha'(x(\gamma, \gamma')),
\end{aligned}$$

i.e. post composition with α'. Naturality follows from the commutation of precomposition with postcomposition.

A type $A' : (x : X(1_{\mathcal{M}}))(x' : X'(x)) \to A(1_{\mathcal{M}}, x) \to \mathrm{Set}$ is sent to the following dependent presheaf (type of $\mathsf{psh}_{\mathbf{Fam}}(\mathbf{Scone}_{\mathcal{M}})$ over $\mathbb{Y}_{X'}$ and A)

$$\begin{aligned}
&\mathbb{Y}_{A'}(\Gamma, \Gamma', x : X(\Gamma), x' : \mathbb{Y}_{X'}(\Gamma, \Gamma', x), a : A(\Gamma, x))\\
&\quad = \forall(\gamma : \mathcal{M}(1, \Gamma))\ (\gamma' : \Gamma'(\gamma)) \to A'(x[\gamma], x'(\gamma, \gamma'), a[\gamma]).
\end{aligned}$$

We check the weak preservation of extended contexts: we have natural isomorphisms

$$\begin{aligned}
&\mathbb{Y}_{X'.A'}(\Gamma, \Gamma', x : X(\Gamma), a : A(\Gamma, x))\\
&\quad = \forall(\gamma : \mathcal{M}(1, \Gamma))\ (\gamma' : \Gamma'(\gamma)) \to (x' : X'(x[\gamma])) \times (a' : A'(x', a[\gamma]))\\
&\quad \cong (x' : \forall \gamma\ \gamma' \to X'(x[\gamma])) \times (a' : \forall \gamma\ \gamma' \to A'(x'(\gamma, \gamma', a[\gamma])))\\
&\quad \cong (x' : \mathbb{Y}_{X'}(\Gamma, \Gamma', x)) \times (a' : \mathbb{Y}_{A'}(\Gamma, \Gamma', x, x', a)).
\end{aligned}$$

Finally we can check the weak preservation of Π-types: we have

$$\begin{aligned}
&\mathbb{Y}_{\Pi(A',B')}(\Gamma, \Gamma', x : X(\Gamma), x' : \mathbb{Y}_{X'}(\Gamma, \Gamma', x), f : (\Pi(A, B))(\Gamma, x))\\
&\quad = \forall \gamma\ \gamma' \to \forall(a : \mathcal{C}.\mathsf{Tm}(1, A[\gamma]))\ (a' : A'(\gamma', a)) \to B'((\gamma', a'), \mathsf{app}(f[\gamma], a)),\\
&\quad \cong \forall(\gamma, a)\ (\gamma', a') \to B'((\gamma', a'), \mathsf{app}(f[\gamma], a)).
\end{aligned}$$

To conclude, observe that Π-types in $\mathsf{psh}_{\mathbf{Fam}}(\mathbf{Scone}_{\mathcal{M}})$ are computed as described in Construction 5.2.2.

$$\begin{aligned}
&(\Pi(\mathbb{Y}_{A'}, \mathbb{Y}_{B'}))(\Gamma, \Gamma', x, x', f),\\
&\quad = \mathbb{Y}_{B'}((\Gamma.A), (\Gamma'.\mathbb{Y}_{A'}), x[\boldsymbol{p}_A], (\lambda(\gamma', a') \mapsto x'(\gamma')), \mathsf{app}(f, \boldsymbol{q}_A)),\\
&\quad \cong \forall(\gamma, a)\ (\gamma', a') \to B'((\gamma', a'), \mathsf{app}(f, \boldsymbol{q}_A)[\langle \gamma, a \rangle]),\\
&\quad \cong \forall(\gamma, a)\ (\gamma', a') \to B'((\gamma', a'), \mathsf{app}(f[\gamma], a)).
\end{aligned}$$

□

Definition 6.1.12. Given a displayed higher-order model $\mathbb{M}^\bullet$, the **Scone-contextualization** $\mathsf{cxl}(\mathbb{M}^\bullet)$ is the displayed $\mathcal{T}$-algebra over $\mathcal{M}$ determined by the composite map

$$\mathcal{T} \xrightarrow{\mathbb{M}^\bullet} \mathbf{Fam}[\mathsf{eval}_{1_{\mathcal{M}}}] \xrightarrow{\mathbb{Y}} \mathsf{psh}_{\mathbf{Fam}}(\mathbf{Scone}_{\mathcal{M}}).$$

⌟

Lemma 6.1.13. *The contextual core of the* **Scone***-contextualization is isomorphic to the telescopic displayed contextualization.*

Proof. The contextual core of the **Scone**-contextualization is the contextual model classified by the following composite map

$$\mathcal{T}^{\mathsf{cxl}} \to \mathcal{T} \xrightarrow{\mathbb{M}^\bullet} \mathbf{Fam}[\mathsf{eval}_{1_\mathcal{M}}] \xrightarrow{\mathbb{Y}} \mathsf{psh}_{\mathbf{Fam}}(\mathbf{Scone}_\mathcal{M}) \xrightarrow{\mathsf{eval}_1} \mathbf{Fam}.$$

But an easy computation shows that the composition

$$\mathbf{Fam}[\mathsf{eval}_{1_\mathcal{M}}] \xrightarrow{\mathbb{Y}} \mathsf{psh}_{\mathbf{Fam}}(\mathbf{Scone}_\mathcal{M}) \xrightarrow{\mathsf{eval}_1} \mathbf{Fam}$$

is naturally isomorphic to the projection $\mathbf{Fam}[\mathsf{eval}_{1_\mathcal{M}}] \to \mathbf{Fam}$.

So that map is also the one classifying the telescopic displayed contextualization. □

Example 6.1.14. We compute the components of the **Scone**-contextualization $\mathsf{Cxl}_{\mathbf{Scone}}(\mathbb{M}^\bullet)$ of a displayed higher-order model $\mathbb{M}^\bullet$ of $\mathcal{T}_\Sigma$, with components as in example 6.1.9.

It is a displayed $\mathcal{T}_\Sigma$-algebra, with $\mathbf{Scone}_\mathcal{M}$ as the base displayed category.

- A displayed type over $(\Gamma : \mathcal{M}, \Gamma' : \mathcal{M}(1,\Gamma) \to \mathsf{Set})$ and $A : \mathcal{M}.\mathsf{Ty}(\Gamma)$ is

  $$A' : \forall(\gamma : \mathcal{M}(1,\Gamma))\ (\gamma' : \Gamma'(\gamma)) \to \mathsf{Ty}^\bullet(A[\gamma]).$$

  The extension of Γ' by the type A' is

  $$(\Gamma'.A') = \lambda\langle\gamma, a\rangle \mapsto (\gamma' : \Gamma'(\gamma)) \times (a' : \mathsf{Tm}^\bullet(A'(\gamma,\gamma'), a[\gamma])).$$

  The substitution operation is precomposition with

  $$f' : (\delta' : \Delta'(\delta)) \to \Gamma'(f \circ \delta).$$

- A displayed term over $(\Gamma : \mathcal{M}, \Gamma' : \mathcal{M}(1,\Gamma) \to \mathsf{Set})$ and $a : \mathcal{M}.\mathsf{Tm}(\Gamma, A)$, of type A', is

  $$a' : \forall(\gamma : \mathcal{M}(1,\Gamma))\ (\gamma' : \Gamma'(\gamma)) \to \mathsf{Tm}^\bullet(A'(\gamma,\gamma'), a[\gamma]).$$

- The operations of the Σ-type structure are as follows:

  $$\begin{aligned}
  &\Sigma'(A', B') = \lambda\gamma\ \gamma' \mapsto \Sigma^\bullet(A'(\gamma,\gamma'), \lambda a\ a' \mapsto B'(\langle\gamma, a\rangle, (\gamma', a'))),\\
  &\mathsf{pair}'(a', b') = \lambda\gamma\ \gamma' \mapsto \mathsf{pair}^\bullet(a'(\gamma,\gamma'), b'(\gamma,\gamma')),\\
  &\mathsf{fst}'(p') = \lambda\gamma\ \gamma' \mapsto \mathsf{fst}^\bullet(p'(\gamma,\gamma')),\\
  &\mathsf{snd}'(p') = \lambda\gamma\ \gamma' \mapsto \mathsf{snd}^\bullet(p'(\gamma,\gamma')).
  \end{aligned}$$

  The most interesting case is the binder Σ: because B' is in an extended context, it depends on both $\gamma' : \Gamma'(\gamma)$ and $a' : \mathsf{Tm}^\bullet(A'(\gamma,\gamma'), a[\gamma])$ for some a. But the second argument of $\Sigma^\bullet$ quantifies exactly on the additional data a and a'.

- The equations from the SOGAT (such as $\mathsf{fst}(\mathsf{pair}(x,y)) = x$) all follow pointwise from the corresponding equations in the higher-order model.

- The naturality conditions are all essentially associativity of function composition, e.g.

$$\begin{aligned}
&\mathsf{fst}'(p')[f'] \\
&\quad = \lambda\delta\ \delta' \mapsto \mathsf{fst}^\bullet(p'(f \circ \delta, f'(\delta'))), \\
&\quad = \mathsf{fst}'(p'[f']). \\
&\Sigma'(A', B')[f'] \\
&\quad = \lambda\delta\ \delta' \mapsto \Sigma^\bullet(A'(f \circ \delta, f'(\delta')), \lambda a\ a' \mapsto B'(\langle f \circ \delta, a\rangle, (f'(\gamma'), a'))), \\
&\quad = \Sigma'(A'[f'], B'[\lambda(\gamma', a') \mapsto (f(\gamma'), a')]).
\end{aligned}$$

⌟

Remark 6.1.15. A displayed higher-order model over the terminal model is the same as a non-displayed higher-order model. In that case, the displayed and non-displayed contextualizations are equivalent. ⌟

6.2 Application: canonicity

As a first example of the use of displayed higher-order models and displayed contextualizations, we use these tools to prove canonicity for MLTT.

Definition 6.2.1. We say that a $\mathcal{T}_{\mathsf{MLTT}}$-algebra $\mathcal{C}$ satisfies **canonicity** if for every closed boolean term $(\vdash b : \mathsf{Bool}) \in \mathcal{C}$, we have either $b = \mathsf{true}$ or $b = \mathsf{false}$. ⌟

Remark 6.2.2. Let $\mathcal{T}_{\mathsf{Bool}}$ be the GAT of bipointed sets, so that the initial $\mathcal{T}_{\mathsf{Bool}}$-algebra is the set $\{\mathsf{true}, \mathsf{false}\}$ of booleans. There is a GAT morphism $B : \mathcal{T}_{\mathsf{Bool}} \to \mathcal{T}^{\mathsf{fo}}_{\mathsf{MLTT}}$ that selects the sort $\mathsf{tm}(\diamond, \mathsf{Bool})$ of closed boolean terms and the closed boolean terms true and false. Then canonicity for $\mathcal{C} : \mathbf{Alg}_{\mathcal{T}_{\mathsf{MLTT}}}$ says that the $\mathcal{T}_{\mathsf{Bool}}$-algebra morphism $\mathbf{0}_{\mathsf{Bool}} \to B^*(\mathcal{C})$ is a trivial fibration of $\mathcal{T}_{\mathsf{Bool}}$-algebras.

This means that canonicity can be seen as an embedding result: the theory of booleans can faithfully be embedded into MLTT. The same question can be asked for other GAT morphisms $F : \mathcal{T}_1 \to \mathcal{T}_2$. When is a $\mathcal{T}_1$-algebra morphism $f : \mathcal{M} \to F^*(\mathcal{N})$ a trivial fibration? In particular, the morphisms $\mathbf{0}_{\mathcal{T}_1} \to F^*(\mathbf{0}_{\mathcal{T}_2})$ and more generally the units $\eta_{\mathcal{M}} : \mathcal{M} \to F^*(F_!(\mathcal{M}))$ are interesting. ⌟

Remark 6.2.3. There are multiple slightly different statements of canonicity, e.g. we could also include that $\mathsf{true} \neq \mathsf{false}$.

Canonicity is often used to justify that a type theory has computational content, although it has meaning even in a classical metatheory: it says that every closed boolean is standard in the syntax of type theory. ⌟

6.2.1 The canonicity higher-order model

We construct a displayed higher-order model Can of $\mathcal{T}_{\mathsf{MLTT}}$ over the initial model $\mathbf{0}_{\mathsf{MLTT}}$.

- A displayed n-small type over a closed type $A : \mathbf{0}_{\mathsf{MLTT}}.\mathsf{Ty}_n(1)$ is a **n-small canonicity family**, that is a family

$$A^\bullet : \mathbf{0}_{\mathsf{MLTT}}.\mathsf{Tm}(1, A) \to \mathsf{Set}_n.$$

  The family $A^\bullet$ is also called logical predicate (or computability/reducibility predicate, or computability family to emphasize the fact that it is a family of sets rather than a family of propositions).

An element of $A^\bullet(a)$ should be seen as a witness for the computability of the closed term a. In particular, we will need to know that an element of $\mathsf{Bool}^\bullet(b)$ ensures that the closed boolean term b is canonical. Because we are defining a displayed higher-order model, the logical predicate $A^\bullet$ will be computed by induction on the structure of the type A.

- A displayed term of a closed term $a : \mathbf{0}_{\mathsf{MLTT}}.\mathsf{Tm}(1, A)$ is an element

$$a^\bullet : A^\bullet(a).$$

Lifting The definition of the displayed lifting operation is essentially forced by the required isomorphism $\mathsf{Tm}^\bullet(\mathsf{Lift}^\bullet(A^\bullet), a) \cong \mathsf{Tm}^\bullet(A^\bullet, \mathsf{lower}(a))$:

$$\mathsf{Lift}^\bullet(A^\bullet) = \lambda a \mapsto A^\bullet(\mathsf{lower}(a)).$$

Universes The definition of the displayed universe $\mathcal{U}_n^\bullet$ is essentially forced by the required isomorphism $\mathsf{Tm}^\bullet(\mathcal{U}_n^\bullet, A) \cong \mathsf{Ty}^\bullet(\mathsf{El}(A))$: an element of $\mathcal{U}_n^\bullet(A)$ is a n-small canonicity family over the type $\mathsf{El}(A)$.

1-type The definition of the displayed **1**-type is essentially forced by the required isomorphism $\mathsf{Tm}^\bullet(\mathbf{1}^\bullet, x) \cong \{\star\}$:

$$\mathbf{1}^\bullet = \lambda x \mapsto \{\star\}.$$

Σ-types The definition of the displayed Σ-types is essentially forced by the required isomorphism $\mathsf{Tm}^\bullet(\Sigma^\bullet(A^\bullet, B^\bullet), p) \cong (a^\bullet : \mathsf{Tm}^\bullet(A^\bullet, \mathsf{fst}(p))) \times \mathsf{Tm}^\bullet(B^\bullet(a^\bullet), \mathsf{snd}(b))$:

$$\Sigma^\bullet(A^\bullet, B^\bullet) = \lambda p \mapsto (a^\bullet : A^\bullet(\mathsf{fst}(p))) \times B^\bullet(\mathsf{snd}(p)).$$

Π-types The definition of the displayed Π-types is essentially forced by the required isomorphism $\mathsf{Tm}^\bullet(\Pi^\bullet(A^\bullet, B^\bullet), f) \cong (\forall a(a^\bullet : \mathsf{Tm}^\bullet(A^\bullet, a)) \to \mathsf{Tm}^\bullet(B^\bullet(a^\bullet), \mathsf{app}(f, a)))$:

$$\Pi^\bullet(A^\bullet, B^\bullet) = \lambda f \mapsto (\forall a(a^\bullet : A^\bullet(a)) \to B^\bullet(a^\bullet)(\mathsf{app}(f, a))).$$

Empty type The family $\mathsf{Empty}^\bullet : \mathbf{0}_{\mathsf{MLTT}}.\mathsf{Tm}(1, \mathsf{Empty}) \to \mathsf{Set}$ is the empty family:

$$\mathsf{Empty}^\bullet(x) = \{\}.$$

Boolean type The family $\mathsf{Bool}^\bullet : \mathbf{0}_{\mathsf{MLTT}}.\mathsf{Tm}(1, \mathsf{Bool}) \to \mathsf{Set}$ is inductively generated by constructors:

$$\mathsf{true}^\bullet : \mathsf{Bool}^\bullet(\mathsf{true}),$$
$$\mathsf{false}^\bullet : \mathsf{Bool}^\bullet(\mathsf{false}).$$

We have an isomorphism $\mathsf{Bool}^\bullet(b) \cong (b = \mathsf{true}) + (b = \mathsf{false})$.
The displayed eliminator for booleans is defined by induction over that family:

$$\mathsf{elimBool}^\bullet(P^\bullet, t^\bullet, f^\bullet, \mathsf{true}^\bullet) = t^\bullet,$$
$$\mathsf{elimBool}^\bullet(P^\bullet, t^\bullet, f^\bullet, \mathsf{false}^\bullet) = f^\bullet.$$

Identity types For any displayed type $A^\bullet$ and displayed term $x^\bullet$, The family

$$\mathsf{Id}^\bullet(a^\bullet, x^\bullet, -) : \forall y\ (y^\bullet : A^\bullet(y)) \to \mathbf{0}_{\mathsf{MLTT}}.\mathsf{Tm}(1, \mathsf{Id}(A, x, y)) \to \mathsf{Set}$$

is inductively generated by a constructor

$$\mathsf{refl}^\bullet : \mathsf{Id}^\bullet(A^\bullet, x^\bullet, x^\bullet, \mathsf{refl}).$$

As with booleans, the displayed eliminator for identity types is defined by induction over that family.

W-types For any displayed type $A^\bullet$ and displayed dependent type $B^\bullet$, the family $W^\bullet(A^\bullet, B^\bullet) : \mathbf{0}_{\mathsf{MLTT}}.\mathsf{Tm}(1, W(A, B)) \to \mathsf{Set}$ is inductively generated by a single constructor

$$\begin{aligned}\mathsf{sup}^\bullet : \forall a\ f\ (a^\bullet : A^\bullet(a))(f^\bullet : \forall b\ b^\bullet \to W^\bullet(A^\bullet, B^\bullet, \mathsf{app}(f, b)))\\ \to W^\bullet(A^\bullet, B^\bullet, \mathsf{sup}(a, f)).\end{aligned}$$

As with booleans, the displayed eliminator for W-types is defined by induction over that family.

6.2.2 The canonicity result

We can construct the displayed contextualization $\mathsf{Cxl}(\mathsf{Can})$ (either the telescopic contextualization or the **Scone**-contextualization). It is displayed over $\mathbf{0}_{\mathsf{MLTT}}$. By initiality of $\mathbf{0}_{\mathsf{MLTT}}$, we have a section $[\![-]\!] : \mathbf{0}_{\mathsf{MLTT}} \to \mathsf{Cxl}(\mathsf{Can})$.

Theorem 6.2.4. *The initial algebra $\mathbf{0}_{\mathsf{MLTT}}$ satisfies canonicity.*

Proof. Let $(\vdash b : \mathsf{Bool}) \in \mathbf{0}_{\mathsf{MLTT}}$ be a closed boolean term. Applying the section $[\![-]\!]$, we have $[\![b]\!] : \mathsf{Can}.\mathsf{Tm}^\bullet([\![\mathsf{Bool}]\!], b)$. By definition of Can, this means that $[\![b]\!] : \mathsf{Bool}^\bullet(b)$, i.e. that b is either true or false. □

Using the fact that $[\![-]\!]$ preserves the operations of a $\mathcal{T}_{\mathsf{MLTT}}$-algebra, we can compute its action on closed terms. For any $\Gamma \in \mathbf{0}_{\mathsf{MLTT}}$, we can see $[\![\Gamma]\!] : \mathbf{0}_{\mathsf{MLTT}}(1, \Gamma) \to \mathsf{Set}$ as a family of **evaluation environments**. an element $\gamma^\bullet : [\![\Gamma]\!](\gamma)$ contains a mapping from every variable $\underline{a}$ in Γ to a semantic value in $\mathsf{Can}.\mathsf{Tm}^\bullet([\![A]\!](\gamma^\bullet), a)$, where γ sends the variable $\underline{a}$ to $a : \mathbf{0}_{\mathsf{MLTT}}.\mathsf{Tm}(1, A)$. Then for any open term a and evaluation environment $\gamma^\bullet$, $[\![a]\!](\gamma^\bullet)$ is a semantic value over a.

The following example demonstrates how to compute the action of $[\![-]\!]$ on a small term.

$$\begin{aligned}&[\![\mathsf{app}(\mathsf{lam}(\lambda b \mapsto \mathsf{elimBool}(\mathsf{false}, \mathsf{true}, b)), \mathsf{true})]\!]\\ &\quad = [\![\mathsf{lam}(\lambda b \mapsto \mathsf{elimBool}(\mathsf{false}, \mathsf{true}, b))]\!]([\![\mathsf{true}]\!]) && \text{(Preservation of app)}\\ &\quad = [\![\mathsf{elimBool}(\mathsf{false}, \mathsf{true}, \underline{b})]\!][\underline{b} \mapsto [\![\mathsf{true}]\!]] && \text{(Preservation of lam)}\\ &\quad = \mathsf{elimBool}^\bullet([\![\mathsf{false}]\!], [\![\mathsf{true}]\!], [\![\underline{b}]\!][\underline{b} \mapsto [\![\mathsf{true}]\!]]) && \text{(Preservation of elimBool)}\\ &\quad = \mathsf{elimBool}^\bullet([\![\mathsf{false}]\!], [\![\mathsf{true}]\!], [\![\mathsf{true}]\!]) && \text{(Preservation of variables)}\\ &\quad = \mathsf{elimBool}^\bullet(\mathsf{false}^\bullet, \mathsf{true}^\bullet, \mathsf{true}^\bullet) && \text{(Preservation of true and false)}\\ &\quad = \mathsf{false}^\bullet.\end{aligned}$$

6.3 Relative induction principles

Now that we know how to prove properties of closed terms by induction, we turn our attention to relative induction principles, which will allow for the use of the displayed contextualization even when proving properties of open terms.

Fix a SOGAT $\mathcal{T}$.

Definition 6.3.1. For any $\mathcal{T}$-algebra $\mathcal{C}$, we define a functor $\mathcal{C}[-] : \mathcal{C}^{\mathsf{op}} \to \mathbf{Alg}_{\mathcal{T}}$ which sends $\Gamma \in \mathcal{C}$ to the free extension $\mathcal{C}[\underline{\gamma} : 1 \to \Gamma]$ of $\mathcal{C}$ by a new morphism $\underline{\gamma} : 1 \to \Gamma$. We also denote this free extension by $\mathcal{C}[\underline{\gamma} : \Gamma]$, or just $\mathcal{C}[\Gamma]$. For any other $\mathcal{T}$-algebra $\mathcal{D}$, algebra morphism $F : \mathcal{C} \to \mathcal{D}$ and morphism $\gamma : 1_{\mathcal{D}} \to F(\Gamma)$, there exists a unique morphism $G : \mathcal{C}[\underline{\gamma} : \Gamma] \to \mathcal{D}$ such that $G \circ i = F$ and $G(\underline{\gamma}) = \gamma$

The action of this functor on $f : \Delta \to \Gamma$ is the morphism $\mathcal{C}[f] : \mathcal{C}[\underline{\gamma} : \Gamma] \to \mathcal{C}[\underline{\delta} : \Delta]$ that sends $\underline{\gamma} : 1 \to \Gamma$ to $(f \circ \underline{\gamma}) : 1 \to \Delta$. ⌟

For any $\mathcal{T}$-algebra $\mathcal{D}$ and functor $F : \mathcal{C} \to \mathcal{D}$, we write $\mathcal{D}[F(-)]$ for the composition of $\mathcal{D}[-] : \mathcal{D}^{\mathsf{op}} \to \mathbf{Alg}_{\mathcal{T}}$ with $F : \mathcal{C}^{\mathsf{op}} \to \mathcal{D}^{\mathsf{op}}$.

Proposition 6.3.2. *The construction of $\mathcal{C}[-] : \mathcal{C}^{\mathsf{op}} \to \mathbf{Alg}_{\mathcal{T}}$ has an action on $\mathcal{T}$-algebra morphisms: for any $\mathcal{T}$-algebra morphism $F : \mathcal{C} \to \mathcal{D}$, there is a natural transformation*

$$F[-] : \mathcal{C}[-] \Rightarrow \mathcal{D}[F(-)],$$

functorially in F.

Proof. Take a morphism $F : \mathcal{C} \to \mathcal{D}$. Given an object $\Gamma \in \mathcal{C}$,

$$F[\Gamma] : \mathcal{C}[\underline{\gamma} : \Gamma] \to \mathcal{D}[\underline{\gamma'} : F(\Gamma)]$$

is defined as the $\mathcal{T}$-algebra morphism that sends $\underline{\gamma}$ to $\underline{\gamma'}$. The naturality square

$$\begin{array}{ccc} \mathcal{C}[\underline{\gamma} : \Gamma] & \xrightarrow{F[\Gamma]} & \mathcal{D}[\underline{\gamma'} : F(\Gamma)] \\ \downarrow{\scriptstyle \mathcal{C}[f]} & & \downarrow{\scriptstyle \mathcal{D}[F(f)]} \\ \mathcal{C}[\underline{\delta} : \Delta] & \xrightarrow{F[\Delta]} & \mathcal{D}[\underline{\delta'} : F(\Delta)] \end{array}$$

commutes by the universal property of $\mathcal{C}[\underline{\gamma} : \Gamma]$, as both paths send $\underline{\gamma}$ to $F(f) \circ \delta'$.

Checking functoriality is straightforward. □

Lemma 6.3.3. *If $\mathcal{C}$ is contextual, then for any representable sort $(\partial A \vdash A\ \mathsf{type}_{\mathsf{rep}}) \in \mathcal{T}$ and closed element $(\vdash \sigma : \partial A) \in \mathcal{C}$, then $(\mathcal{C} /\!\!/ p) : \mathcal{C} \to (\mathcal{C} /\!\!/ 1.(a : A(\sigma)))$ has the universal property of $\mathcal{C}[a : A(\sigma)]$, where p is the projection $1.A(\sigma) \to 1$.*

Proof. Note that because $\mathcal{C}$ is contextual, we have $\mathcal{C} \cong (\mathcal{C} /\!\!/ 1)$; we implicitly identify these two models. Similarly, $((\mathcal{C} /\!\!/ 1.A(\sigma)) /\!\!/ 1.A(\sigma)) \cong (\mathcal{C} /\!\!/ 1.A(\sigma).A(\sigma))$.

We name some morphisms between contexts:

$$1.(a_1 : A(\sigma)).(a_2 : A(\sigma)) \underset{d}{\overset{p_1}{\underset{p_2}{\leftarrow}}} 1.(a : A(\sigma)) \xrightarrow{p} 1.$$

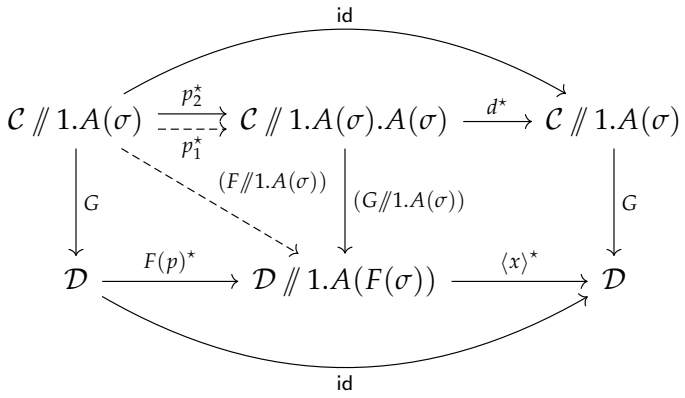
The diagonal morphism d is a section of both p_1 and p_2.

Take any contextual model $\mathcal{D}$, with a morphism $F : \mathcal{C} \to \mathcal{D}$ and an element $(\vdash x : A(F(\sigma))) \in \mathcal{D}$. We have a morphism $\langle x \rangle : 1 \to 1.A(F(\sigma))$ in $\mathcal{D}$. It is a section of $F(p)$.

We have to prove that there exists a unique morphism $G : (\mathcal{C} /\!\!/ 1.(a : A(\sigma))) \to \mathcal{D}$ such that $G(a) = x$ and $G \circ p^\star = F$. Assume given such a morphism.

Note that $d = \langle a \rangle$ in $1.A(\sigma).A(\sigma) \to 1.A(\sigma)$.

Consider the following diagram:



Because $G(p_2) = F(p)$ (this follows from $p_2 = p^\star(p)$ and $G \circ p^\star = F$) and $G(a) = \langle x \rangle$, the two solid squares commute. Because d and $\langle x \rangle$ are sections of p_2 and $F(p)$, the solid part of the diagram commutes.

The triangle formed by the two dashed arrows commutes, because it is the application of the functor $(- /\!\!/ 1.A(\sigma))$ to the equality $G \circ p^\star = F$.

However $d^\star \circ p_1^\star = d^\star \circ p_2^\star = \mathsf{id}$. Therefore the pentagon obtained by composing the dashed triangle with the right square commutes.

Therefore $G = \langle x \rangle^\star \circ (F /\!\!/ 1.A(\sigma))$. This proves unicity, and the candidate $\langle x \rangle^\star \circ (F /\!\!/ 1.A(\sigma))$ always satisfies the necessary properties, witnessing the existence. □

Remark 6.3.4. There is another way to arrive at the functor

$$\mathcal{C} \mapsto (\Gamma \mapsto (\mathcal{C} /\!\!/ \Gamma))$$

that sends a $\mathcal{T}$-algebra to its contextual slice functor.

Consider the morphism

$$\mathcal{T}^{\mathsf{cxl}} \to \mathcal{T}$$

of Σ-CwFs from definition 5.3.11. Its domain $\mathcal{T}^{\mathsf{cxl}}$ is a GAT, which can be seen as a SOGAT without generating representable sorts.

Applying the functor that sends a SOGAT to its GAT of algebras, we obtain a morphism of GATs

$$(\mathcal{T}^{\mathsf{cxl}})^{\mathsf{fo}} \to \mathcal{T}^{\mathsf{fo}},$$

which induces a functor $\mathbf{Alg}_{\mathcal{T}^{\mathsf{fo}}} \to \mathbf{Alg}_{(\mathcal{T}^{\mathsf{cxl}})^{\mathsf{fo}}}$.

An object of $\mathbf{Alg}_{(\mathcal{T}^{\mathsf{cxl}})^{\mathsf{fo}}}$ is equivalently a category $\mathcal{C}$ with a terminal object together with a functor $\mathcal{C}^{\mathsf{op}} \to \mathbf{Alg}_{\mathcal{T}^{\mathsf{cxl}}}$. The functor $\mathbf{Alg}_{\mathcal{T}^{\mathsf{fo}}} \to \mathbf{Alg}_{(\mathcal{T}^{\mathsf{cxl}})^{\mathsf{fo}}}$ sends a $\mathcal{T}$-algebra to its contextual slice functor. It is not clear whether this provides an useful insight. ⌟

Proposition 6.3.5. *Let $\mathcal{D}$ be a contextual $\mathcal{T}$-algebra, $K : \mathcal{T}_{\mathsf{Fam}_{\mathsf{rep}}} \to \mathcal{T}$ be a GAT morphism selecting a representable sort $(A : \mathsf{Ty} \vdash \mathsf{Tm}(A)\ \mathsf{type}_{\mathsf{rep}}) \in \mathcal{T}$ and and $F : \mathcal{C} \to K^*(\mathcal{D})$ be a CwF morphism.*

In the internal language of $\mathbf{Psh}(\mathcal{C})$*, for every* $A : \mathcal{C}.\mathsf{Ty}(1)$*, the map*

$$\mathcal{D}_F[\underline{a} : F(A)] \to (\prod_{a:\mathcal{C}.\mathsf{Tm}(A)} \mathcal{D}_F)$$

which sends $\underline{a}$ *to* $(\lambda a \mapsto F(a))$ *is an isomorphism of* $\mathcal{T}$*-algebras.*

Proof. By lemma 6.3.3, we can consider $(\mathcal{D}_F /\!\!/ (1.F(A)))$ instead of $\mathcal{D}_F[\underline{a} : F(A)]$.

The isomorphism can be checked levelwise. For any $\Gamma \in \mathcal{C}$ and $A : \mathcal{C}.\mathsf{Ty}(\Gamma)$, we have to compare the evaluation at (Γ, A) of the functors corresponding to

$$(\mathcal{D}_F /\!\!/ 1.F(A))$$

and

$$\prod_{a:\mathcal{C}.\mathsf{Tm}(A)} \mathcal{D}_F.$$

Write $\mathbb{D}[F(-)]$ for the functor corresponding to $\mathcal{D}_F$.

The operation $(- /\!\!/ 1.-)$ can be induced by a morphism of GATs

$$\mathcal{T}^{\mathsf{cxl}} \to \mathcal{T}^{\mathsf{cxl}}[\underline{A} : \mathsf{Ty}(1)].$$

Therefore it is computed objectwise, and the evaluation of the left hand side at Γ is

$$(\mathbb{D}[F(\Gamma)] /\!\!/ 1.F(A))$$

which is isomorphic to $(\mathbb{D}[F(\Gamma).F(A)])$.

Products with a locally representable domain can be computed by evaluating the codomain at an extended context. This is also true for products of algebras of arbitrary GATs. Thus the evaluation of the right hand side at Γ is

$$\mathbb{D}[F(\Gamma.A)].$$

Because F is a morphism of CwFs, we have $F(\Gamma).F(A) \cong F(\Gamma.A)$, inducing an isomorphism between the left and right hand sides.

If we track the underlying map of the isomorphism, we see that it coincides with the components of the natural transformations corresponding to the internal map. □

Write $\mathcal{S}$ for the initial $\mathcal{T}$-algebra. Some results could also be stated for an arbitrary initial algebra $\mathcal{C}$, but can be recovered by working with the initial algebra of the GAT $\mathcal{T}[\backslash\mathcal{C}]$ of $\mathcal{T}$-algebras under $\mathcal{C}$.

Our goal is to establish an universal property for the functor $\mathcal{S}[F(-)] : \mathcal{C}^{\mathsf{op}} \to \mathbf{Alg}_{\mathcal{T}}$, depending on the functor F. Recall that the functor category $[\mathcal{C}^{\mathsf{op}}, \mathbf{Alg}_{\mathcal{T}}]$ is a CwF, where objects are presheaves of algebras, types are presheaves of displayed algebras and terms are presheaves of sections of displayed algebras. A universal property for an object of this CwF is a way to construct terms of a given type, i.e. presheaves of sections of displayed algebras over a given presheaf of displayed algebras.

Take a dependent functor

$$\mathcal{S}^{\bullet}(-) : \mathcal{C}^{\mathsf{op}} \to \mathbf{DispAlg}_{\mathcal{T}}[S[F(-)]].$$

We want to understand how to construct a dependent functor

$$S(-) : \mathcal{C}^{\mathsf{op}} \to \mathbf{Sect}_{\mathcal{T}}[\mathcal{S}^{\bullet}(-)].$$

We first give an explicit description of the total category of $\mathbf{Sect}_{\mathcal{T}}[\mathcal{S}^{\bullet}(-)]$.

Definition 6.3.6. Let $\mathcal{S}^\bullet(-) : \mathcal{C}^{\mathsf{op}} \to \mathbf{DispAlg}_{\mathcal{T}}[S[F(-)]]$ be a dependent functor. The **category of sections** $\mathbf{Sect}^{\mathsf{op}}_{\mathcal{T}}[\mathcal{S}^\bullet]$ of $\mathcal{S}^\bullet$ is the displayed category over $\mathcal{C}$ obtained as the following pullback in **Cat**:

$$\begin{array}{ccc} \mathbf{Sect}^{\mathsf{op}}_{\mathcal{T}}[\mathcal{S}^\bullet] & \longrightarrow & \mathbf{Sect}^{\mathsf{op}}_{\mathcal{T}} \\ \downarrow{\scriptstyle \pi} & \lrcorner & \downarrow \\ \mathcal{C} & \xrightarrow{\mathcal{S}^\bullet(-)} & \mathbf{DispAlg}^{\mathsf{op}}_{\mathcal{T}} \end{array}$$

In other words,

- A displayed object over $\Gamma \in \mathcal{C}$ is a section S_Γ of $\mathcal{S}^\bullet(\Gamma)$ over $\mathcal{S}[F(\Gamma)]$.

  By the universal property of $\mathcal{S}[F(\Gamma)]$ (which is $\mathcal{S}[\underline{\gamma} : F(\Gamma)]$, such a section is uniquely determined by $S_\Gamma(\underline{\gamma}) : \mathcal{S}^\bullet(\Gamma).\mathsf{Hom}(1, F(\Gamma))[\underline{\gamma}]$.

- A displayed morphism over $(f : \Delta \to \Gamma) \in \mathcal{C}$ is a witness of the equality $S_\Delta \circ \mathcal{S}[F(f)] = \mathcal{S}^\bullet(f) \circ S_\Gamma$.

$$\begin{array}{ccc} \mathcal{S}^\bullet(\Gamma) & \xrightarrow{\mathcal{S}^\bullet(f)} & \mathcal{S}^\bullet(\Delta) \\ {\scriptstyle S_\Gamma}\uparrow \dashv \downarrow & & \downarrow \dashv \uparrow{\scriptstyle S_\Delta} \\ \mathcal{S}[F(\Gamma)] & \xrightarrow{\mathcal{S}[F(f)]} & \mathcal{S}[F(\Delta)] \end{array}$$

  To check this equality, it suffices to check that

$$(\mathcal{S}^\bullet(f)(S_\Gamma(\underline{\gamma})) = S_\Delta(F(f) \circ \underline{\delta})) \in \mathcal{S}^\bullet(\Delta).\mathsf{Hom}(1, F(\Gamma))[F(f) \circ \underline{\delta}]$$

- Identities and compositions are determined by horizontal compositions of the above squares. ⌟

The data of a dependent functor $\mathcal{C}^{\mathsf{op}} \to \mathbf{Sect}_{\mathcal{T}}[\mathcal{S}^\bullet(-)]$ is equivalent to the data of a section of $\pi : \mathbf{Sect}^{\mathsf{op}}_{\mathcal{T}}[\mathcal{S}^\bullet] \to \mathcal{C}$.

We are typically interested in situations where $\mathcal{C}$ is defined similarly to the category of renamings of a CwF, meaning that $\mathcal{C}$ is initial among categories equipped with a terminal object preserved by F, some context extensions (locally representable presheaves) that are also preserved by F, and possibly some additional operations preserved by F (in the case of renamings, we do not include any such operation), up to some equations.

We therefore have to understand how to equip $\mathbf{Sect}^{\mathsf{op}}_{\mathcal{T}}[\mathcal{S}^\bullet]$ with the same data, as the universal property of $\mathcal{C}$ can then provide a section of $\pi : \mathbf{Sect}^{\mathsf{op}}_{\mathcal{T}}[\mathcal{S}^\bullet] \to \mathcal{C}$.

We show in proposition 6.3.7 and proposition 6.3.8 how to equip $\mathbf{Sect}^{\mathsf{op}}_{\mathcal{T}}[\mathcal{S}^\bullet]$ with a terminal object and with representing objects for local representable presheaves.

Proposition 6.3.7. *If the category $\mathcal{C}$ has a terminal object $1_{\mathcal{C}}$ that is preserved by F, then $\mathbf{Sect}^{\mathsf{op}}_{\mathcal{T}}[\mathcal{S}^\bullet]$ has a terminal object that is strictly preserved by $\pi : \mathbf{Sect}^{\mathsf{op}}_{\mathcal{T}}[\mathcal{S}^\bullet] \to \mathcal{C}$.*

Proof. Note that $\mathcal{S}[F(1_{\mathcal{C}})] \cong \mathcal{S}[1_{\mathcal{S}}] \cong \mathcal{S}$ is initial.

The terminal object is $(1_{\mathcal{C}}, S_{1_{\mathcal{C}}})$ where $S_{1_{\mathcal{C}}}$ is the unique section of $\mathcal{S}^\bullet(1_{\mathcal{C}})$ over $\mathcal{S}[F(1_{\mathcal{C}})]$ obtained from the initiality of $\mathcal{S}[F(1_{\mathcal{C}})]$.

To check that this object is terminal, it suffices to check that the following diagram commutes.

$$\begin{array}{ccc} \mathcal{S}^\bullet(1_{\mathcal{C}}) & \xrightarrow{\mathcal{S}^\bullet(f)} & \mathcal{S}^\bullet(\Gamma) \\ {\scriptstyle S_{1_{\mathcal{C}}}}\uparrow \dashv \downarrow & & \downarrow \dashv \uparrow{\scriptstyle S_\Gamma} \\ \mathcal{S}[F(1_{\mathcal{C}})] & \xrightarrow{\mathcal{S}[F(f)]} & \mathcal{S}[F(\Gamma)] \end{array}$$

This also follows from the initiality of $\mathcal{S}[F(1_{\mathcal{C}})]$. □

Proposition 6.3.8. *Let $(X \vdash Y\ \mathsf{type}_{\mathsf{rep}}) \in \mathcal{T}$ be a representable sort of $\mathcal{T}$. This induces a SOGAT morphism $K : \mathcal{T}_{\mathsf{Fam}_{\mathsf{rep}}} \to \mathcal{T}$, which induces in turn a functor $K^* : \mathbf{Alg}_{\mathcal{T}} \to \mathbf{CwF}$.*

Assume that $\mathcal{C}$ has a CwF structure and that the functor F extends to a CwF morphism $F : \mathcal{C} \to K^*(\mathcal{S})$. We write $X_{\mathcal{C}}, Y_{\mathcal{C}}$ for the types and terms of the CwF $\mathcal{C}$, $X_{\mathcal{S}}$ and $Y_{\mathcal{S}}$ for the types and terms of $K^*(\mathcal{S})$ and F_X, F_Y for the actions of F on types and terms. We write $X_{\mathcal{S}^\bullet}$ and $Y_{\mathcal{S}^\bullet}$ for the presheaf and dependent presheaf determined by the composition

$$\mathcal{C}^{\mathsf{op}} \xrightarrow{\mathcal{S}^\bullet(-)} \mathbf{Alg}_{\mathcal{T}} \xrightarrow{K^*} \mathbf{CwF} \xrightarrow{\mathsf{eval}_1} \mathbf{Fam},$$

where eval_1 is evaluation of the presheaves of types and terms at the terminal object.

Assume that for every $(\Gamma, S_\Gamma) \in \mathbf{Sect}_{\mathcal{T}}^{\mathsf{op}}[\mathcal{S}^\bullet]$ and $x : X_{\mathcal{C}}(\Gamma)$, we have an operation

$$F_Y^\bullet(\Gamma, S_\Gamma, x) : Y_{\mathcal{S}^\bullet}(\Gamma.Y_{\mathcal{C}}[x], S_\Gamma(F_X(x))[\boldsymbol{p}_{Y_{\mathcal{C}}[x]}], F_Y(\boldsymbol{q}_{Y_{\mathcal{C}}[x]}))),$$

which is moreover natural in (Γ, S_Γ): for every morphism $f : (\Delta, S_\Delta) \to (\Gamma, S_\Gamma)$, we have

$$F_Y^\bullet(\Gamma, S_\Gamma, x)[f^+] = F_Y^\bullet(\Delta, S_\Delta, x[f]).$$

Then the dependent presheaf (over $\pi^*(X_{\mathcal{C}}) \in \mathbf{Psh}(\mathbf{Sect}_{\mathcal{T}}^{\mathsf{op}}[\mathcal{S}^\bullet])$)

$$Y'(\Gamma, S_\Gamma, x) \triangleq \{y : Y_{\mathcal{C}}(\Gamma, x) \mid S_\Gamma(y) = F_Y^\bullet(\Gamma, S_\Gamma, x)[y]\}.$$

is locally representable (thus equipping $\mathbf{Sect}^{\mathsf{op}}[\mathcal{S}^\bullet]$ with the structure of a CwF, and the projection $\pi : \mathbf{Sect}^{\mathsf{op}}[\mathcal{S}^\bullet] \to \mathcal{C}$ strictly preserves context extensions (thus determining a CwF morphism).

Proof. We first check some equalities that are required for the statement to make sense. In the line

$$F_Y^\bullet(\Gamma, S_\Gamma, x) : Y_{\mathcal{S}^\bullet}(\Gamma.Y_{\mathcal{C}}[x], S_\Gamma(F_X(x))[\boldsymbol{p}_{Y_{\mathcal{C}}[x]}]), F_Y(\boldsymbol{q}_{Y_{\mathcal{C}}[x]})),$$

we have $S_\Gamma(F_X(x)) : X_{\mathcal{S}^\bullet}(\Gamma, F_X(x))$, and thus $S_\Gamma(F_X(x))[\boldsymbol{p}_{Y_{\mathcal{C}}[x]}] : X_{\mathcal{S}^\bullet}(\Gamma.Y_{\mathcal{C}}[x], F_X(x[\boldsymbol{p}_{Y_{\mathcal{C}}[x]}]))$. Then $F_Y(\boldsymbol{q}_{Y_{\mathcal{C}}[x]}) : Y_{\mathcal{S}}(F_X(x[\boldsymbol{p}_{Y_{\mathcal{C}}[x]}]))$.

In the line

$$F_Y^\bullet(\Gamma, S_\Gamma, x)[f^+] = F_Y^\bullet(\Delta, S_\Delta, x[f]),$$

we have $f^+ : \Delta.Y_{\mathcal{C}}[x[f]] \to \Gamma.X_{\mathcal{C}}[x]$.

In the line

$$Y'(\Gamma, S_\Gamma, x) \triangleq \{y : Y_{\mathcal{C}}(\Gamma, x) \mid S_\Gamma(y) = F_Y^\bullet(\Gamma, S_\Gamma, x)[y]\},$$

we have $S_\Gamma(y) : Y_{\mathcal{S}^\bullet}(\Gamma, S_\Gamma(F_X(x)), F_Y(y))$, and

$$F_Y^\bullet(\Gamma, S_\Gamma, x) : Y_{\mathcal{S}^\bullet}(\Gamma.Y_{\mathcal{C}}[x], S_\Gamma(F_X(x))[\boldsymbol{p}_{Y_{\mathcal{C}}[x]}], F_Y(\boldsymbol{q}_{Y_{\mathcal{C}}[x]})).$$

Thus $F_Y^\bullet(\Gamma, S_\Gamma, x)[y] : Y_{\mathcal{S}^\bullet}(\Gamma, S_\Gamma(F_X(x))[\boldsymbol{p}_{Y_{\mathcal{C}}[x]}][y], F_Y(\boldsymbol{q}_{Y_{\mathcal{C}}[x]})[y])$, i.e.

$$F_Y^\bullet(\Gamma, S_\Gamma, x)[y] : Y_{\mathcal{S}^\bullet}(\Gamma, S_\Gamma(F_X(x)), F_Y(y)).$$

Now take an object $(\Gamma, S_\Gamma) \in \mathbf{Sect}_{\mathcal{T}}^{\mathsf{op}}[\mathcal{S}^\bullet]$ and an element $x : \pi^*(X_{\mathcal{C}})(\Gamma, S_\Gamma)$, i.e. $x : X_{\mathcal{C}}(\Gamma)$. We have an extended context $\Gamma.Y_{\mathcal{C}}[x]$ in $\mathcal{C}$ representing the extension of Γ by $Y_{\mathcal{C}}$ at x. We want to construct an object in $\mathbf{Sect}_{\mathcal{T}}^{\mathsf{op}}[\mathcal{S}^\bullet]$ that represents the extension of (Γ, S_Γ) by Y' at x.

It should be displayed over $\Gamma.Y_{\mathcal{C}}[x]$; its second component should be a section $S_{\Gamma.Y_{\mathcal{C}}[x]}$ of $\mathcal{S}^\bullet(\Gamma.Y_{\mathcal{C}}[x])$ over $\mathcal{S}[F(\Gamma.Y_{\mathcal{C}}[x])]$.

Because F is a CwF morphism, we have $F(\Gamma.Y_{\mathcal{C}}[x]) = F(\Gamma).Y_{\mathcal{S}}[F_X(x)]$.

$$\begin{array}{ccc} \mathcal{S}^\bullet(\Gamma) & \xrightarrow{\mathcal{S}^\bullet(\boldsymbol{p}_{Y_{\mathcal{C}}[x]})} & \mathcal{S}^\bullet(\Gamma.Y_{\mathcal{C}}[x]) \\ {\scriptstyle S_\Gamma}\uparrow\downarrow & & \downarrow\uparrow{\scriptstyle S_{\Gamma.Y_{\mathcal{C}}[x]}} \\ \mathcal{S}[\boldsymbol{\gamma} : F(\Gamma)] & \xrightarrow{\mathcal{S}[\boldsymbol{p}_{Y_{\mathcal{C}}[x]}]} & \mathcal{S}[(\boldsymbol{\gamma}, \boldsymbol{y}) : F(\Gamma).Y_{\mathcal{S}}[F_X(x)]] \end{array}$$

We define $S_{\Gamma.Y_{\mathcal{C}}[x]}$ as the extension of $\mathcal{S}^\bullet(\boldsymbol{p}_{Y_{\mathcal{C}}[x]}) \circ S_\Gamma$ that sends $\boldsymbol{y}$ to $F^\bullet_Y(\Gamma, S_\Gamma, x)$.

This determines an object $(\Gamma.Y_{\mathcal{C}}[x], S_{\Gamma.Y_{\mathcal{C}}[x]})$ of $\mathbf{Sect}^{\mathsf{op}}_{\mathcal{T}}[\mathcal{S}^\bullet]$. The map $\boldsymbol{p}_{Y_{\mathcal{C}}[x]}$ extends to a morphism $(\Gamma.Y_{\mathcal{C}}[x], S_{\Gamma.Y_{\mathcal{C}}[x]}) \to (\Gamma.Y_{\mathcal{C}}[x], S_{\Gamma.Y_{\mathcal{C}}[x]})$, corresponding to the commutation of the above square.

We have an element $\boldsymbol{q}_{Y_{\mathcal{C}}[x]} : Y'(\Gamma.Y_{\mathcal{C}}[x], S_{\Gamma.Y_{\mathcal{C}}[x]}, x[\boldsymbol{p}_{Y_{\mathcal{C}}[x]}])$. For this we need to check

$$S_{\Gamma.Y_{\mathcal{C}}[x]}(\boldsymbol{q}_{Y_{\mathcal{C}}[x]}) = F^\bullet_Y(\Gamma.Y_{\mathcal{C}}[x], S_{\Gamma.Y_{\mathcal{C}}[x]}, x[\boldsymbol{p}_{Y_{\mathcal{C}}[x]}])[\boldsymbol{q}_{Y_{\mathcal{C}}[x]}].$$

Indeed, $S_{\Gamma.Y_{\mathcal{C}}[x]}(\boldsymbol{q}_{Y_{\mathcal{C}}[x]}) = F^\bullet_Y(\Gamma, S_\Gamma, x)$ by definition, and $F^\bullet_Y(\Gamma.Y_{\mathcal{C}}[x], S_{\Gamma.Y_{\mathcal{C}}[x]}, x[\boldsymbol{p}_{Y_{\mathcal{C}}[x]}])[\boldsymbol{q}_{Y_{\mathcal{C}}[x]}] = F^\bullet_Y(\Gamma, S_\Gamma, x)$ by naturality.

Now take any (Δ, S_Δ), a morphism $f : (\Delta, S_\Delta) \to (\Gamma, S_\Gamma)$ and an element $y : Y'(\Delta, S_\Delta, x[f])$. There is an extended substitution $\langle f, y\rangle : \Delta \to \Gamma.Y_{\mathcal{C}}[x]$ in $\mathcal{C}$. It extends to a morphism in $\mathbf{Sect}^{\mathsf{op}}_{\mathcal{T}}[\mathcal{S}^\bullet]$, due to the commutation of the following square:

$$\begin{array}{ccc} \mathcal{S}^\bullet(\Gamma.Y_{\mathcal{C}}[x]) & \xrightarrow{\mathcal{S}^\bullet(\langle f,y\rangle)} & \mathcal{S}^\bullet(\Delta) \\ {\scriptstyle S_{\Gamma.Y_{\mathcal{C}}[x]}}\uparrow\downarrow & & \downarrow\uparrow{\scriptstyle S_\Delta} \\ \mathcal{S}[(\boldsymbol{\gamma}, \boldsymbol{y}) : F(\Gamma).Y_{\mathcal{S}}[F_X(x)]] & \xrightarrow{\mathcal{S}[\langle f,y\rangle]} & \mathcal{S}[\delta : F(\Delta)]. \end{array}$$

Indeed, it suffices to check that $\boldsymbol{\gamma}$ and $\boldsymbol{y}$ are sent to the same element of $\mathcal{S}^\bullet(\Delta)$ by the two paths. For $\boldsymbol{\gamma}$, this follows from $f : (\Delta, S_\Delta) \to (\Gamma, S_\Gamma)$ being a morphism in $\mathbf{Sect}^{\mathsf{op}}_{\mathcal{T}}[\mathcal{S}^\bullet]$. In the case of $\boldsymbol{y}$, both path send it to $F^\bullet_Y(\Delta, S_\Delta, x[f])[y]$, using the definition of $S_{\Gamma.Y_{\mathcal{C}}[x]}$ and the naturality of $F^\bullet_Y$ for the top path, and the fact that $y : Y'(\Delta, S_\Delta, x[f])$ for the bottom path.

Because the displayed morphisms of $\mathbf{Sect}^{\mathsf{op}}_{\mathcal{T}}[\mathcal{S}^\bullet]$ are propositional, the uniqueness of the extended substitution holds for free.

This completes the definition of the representing objects for Y', which is therefore locally representable. By construction of the representing objects, they are strictly preserved by the projection $\pi : \mathbf{Sect}^{\mathsf{op}}[\mathcal{S}^\bullet] \to \mathcal{C}$. □

6.3.1 Relative induction principle over renamings

We now give several variants of relative induction principles over renamings, i.e. relative to the functor $F : \mathbf{Ren}(\mathcal{S}) \to \mathcal{S}$, or more generally to a functor obtained as a composition $\mathbf{Ren}(\mathcal{C}) \xrightarrow{F} \mathcal{C} \xrightarrow{G} \mathcal{S}$.

Definition 6.3.9. Let $(X \vdash Y\ \mathsf{type}_{\mathsf{rep}}) \in \mathcal{T}$ be a representable sort of $\mathcal{T}$. This induces a SOGAT morphism $K : \mathcal{T}_{\mathsf{Fam}_{\mathsf{rep}}} \to \mathcal{T}$, which induces in turn a functor $K^* : \mathbf{Alg}_{\mathcal{T}} \to \mathbf{CwF}$.

Let $\mathcal{C}$ be a CwF, $\mathcal{S}$ be the initial $\mathcal{T}$-algebra, $F : \mathcal{R} \to \mathcal{C}$ be the CwF of renamings of $\mathcal{C}$ and $G : \mathcal{C} \to K^*(\mathcal{S})$ be a CwF morphism[1].

[1] In most applications, $C = K^*(\mathcal{S})$ and G is the identity. The additional flexibility can be useful when $G : \mathbf{0}_{\mathcal{T}_1} \to \mathbf{0}_{\mathcal{T}_2}$ is a morphism between the initial algebras of two theories. In that situation we get a relative induction principle for $\mathbf{0}_{\mathcal{T}_2}$, but internally to the category of renamings of $\mathbf{0}_{\mathcal{T}_1}$.

A **displayed $\mathcal{T}$-algebra with partial variables** is a displayed $\mathcal{T}$-algebra $\mathcal{S}^\bullet$ over $\mathcal{S}_F$ together with a family of partial functions

$$\begin{aligned}\mathcal{S}^\bullet.\mathsf{var} : (A : \mathcal{C}_F.\mathsf{Ty}(1))(A^\bullet : \mathcal{S}^\bullet.\mathsf{Ty}(1, G(A)))\\ \to (a : \mathsf{Var}(A)) \rightharpoonup \mathcal{S}^\bullet.\mathsf{Tm}(1, A^\bullet, \mathsf{var}(a)).\end{aligned}$$

Note that the notion of partial function is generalized algebraic; there is a GAT of displayed $\mathcal{T}$-algebras with partial variables.

A **displayed $\mathcal{T}$-algebra with variables** is a displayed $\mathcal{T}$-algebra $\mathcal{S}^\bullet$ over $\mathcal{S}_F$ together with an operation

$$\begin{aligned}\mathcal{S}^\bullet.\mathsf{var} : (A : \mathcal{C}_F.\mathsf{Ty}(1))(A^\bullet : \mathcal{S}^\bullet.\mathsf{Ty}(1, G(A)))\\ \to (a : \mathsf{Var}(A)) \to \mathcal{S}^\bullet.\mathsf{Tm}(1, A^\bullet, \mathsf{var}(a)).\end{aligned}$$ ⌟

Theorem 6.3.10 (Relative induction principle over a category of renamings)**.** *In the setting of definition 6.3.9, the $\mathcal{T}$-algebra $\mathcal{S}_{FG}$, trivially displayed over itself, is initial among $\mathcal{T}$-algebras with variables.*

In other words, for every displayed $\mathcal{T}$-algebra $\mathcal{S}^\bullet$ over $\mathcal{S}_{FG}$ and operation

$$\mathsf{var}^\bullet : (A : \mathcal{C}_F.\mathsf{Ty}(1))(A^\bullet : \mathcal{S}^\bullet.\mathsf{Ty}(1, G(a)))(a : \mathsf{Var}(A)) \to \mathcal{S}^\bullet.\mathsf{Tm}(1, A^\bullet, G(\mathsf{var}(A, a))),$$

there is a section $[\![-]\!]$ of $\mathcal{S}^\bullet$ over $\mathcal{S}_{FG}$ such that

$$[\![G(\mathsf{var}(A, a))]\!] = \mathsf{var}^\bullet([\![G(A)]\!], a).$$

Proof. First note that the universal property we want to prove is the universal property of the initial algebra of some global internal GAT. It suffices to prove the universal property when $\mathcal{S}^\bullet$ is a *global* displayed algebra of this global internal GAT.

Take a global displayed $\mathcal{T}$-algebra $\mathcal{S}^\bullet$ over S_{FG} and a global operation

$$\mathsf{var}^\bullet : (A : \mathcal{C}.\mathsf{Ty})(A^\bullet : \mathcal{S}^\bullet.\mathsf{Ty}(1, G(a)))(a : \mathsf{Var}(A)) \to \mathcal{S}^\bullet.\mathsf{Tm}(1, A^\bullet, \mathsf{var}(A, a)).$$

Externally, this corresponds to the data of a dependent functor

$$\mathcal{S}^\bullet(-) : \mathcal{R}^{\mathsf{op}} \to \mathbf{DispAlg}_{\mathcal{T}}[\mathcal{S}[F(-)]]$$

and an external natural transformation corresponding to $\mathsf{var}^\bullet$. This natural transformation sends $\Gamma \in \mathcal{R}$, $A : \mathcal{C}.\mathsf{Ty}(F(\Gamma))$, $A^\bullet : \mathcal{S}^\bullet(\Gamma).\mathsf{Ty}(1, G(A))$ and $a : \mathsf{Var}(\Gamma, A)$ to

$$\mathsf{var}^\bullet(\Gamma, A^\bullet, a) : \mathcal{S}^\bullet(\Gamma).\mathsf{Tm}(1, A^\bullet, G(F(a))).$$

We consider the displayed category of sections $\mathbf{Sect}^{\mathsf{op}}_{\mathcal{T}}[\mathcal{S}^\bullet]$.

- By proposition 6.3.7, this category has a terminal object.
- By proposition 6.3.8, $\mathbf{Sect}^{\mathsf{op}}_{\mathcal{T}}[\mathcal{S}^\bullet]$ has a CwF structure, where:
  - A type over (Γ, S_Γ) is a type of $\mathcal{R}$ over Γ;
  - A term of type A over (Γ, S_Γ) is a term a of type A in $\mathcal{R}$ (i.e. a variable of type A) such that
$$S_\Gamma(\mathsf{var}(a)) = \mathsf{var}^\bullet(\Gamma, S_\Gamma(A), a).$$

Thus, by the universal property of the CwF of renamings $\mathcal{R}$, we obtain a section of $\pi : \mathbf{Sect}_{\mathcal{T}}^{\mathsf{op}}[\mathcal{S}^\bullet] \to \mathcal{R}$, corresponding to a dependent functor

$$S(-) : \mathcal{R}^{\mathsf{op}} \to \mathbf{Sect}_{\mathcal{T}}[\mathcal{S}^\bullet(-)],$$

i.e. a section of $\mathcal{S}^\bullet(-)$, as needed. □

We can generalize this relative induction principle to arbitrary $\mathcal{T}$-algebras. For a $\mathcal{T}$-algebra $\mathcal{S}$, we write $\Delta\mathcal{S}$ for the internal $\mathcal{T}$-algebra corresponding to a constant presheaf

$$\mathcal{C}^{\mathsf{op}} \ni - \mapsto \mathcal{S} \in \mathbf{Alg}_{\mathcal{T}}.$$

For any $F : \mathcal{C} \to \mathcal{S}$, there is a morphism $\Delta\mathcal{S} \to \mathcal{S}_F$ corresponding to the natural transformation with components

$$\mathcal{S} \to \mathcal{S}[F(\Gamma)].$$

Theorem 6.3.11. *Let $(X \vdash Y\ \mathsf{type}_{\mathsf{rep}}) \in \mathcal{T}$ be a representable sort of $\mathcal{T}$. This induces a SOGAT morphism $K : \mathcal{T}_{\mathsf{Fam}_{\mathsf{rep}}} \to \mathcal{T}$, which induces in turn a functor $K^* : \mathbf{Alg}_{\mathcal{T}} \to \mathbf{CwF}$. Let $\mathcal{C}$ be a CwF, $\mathcal{S}$ be any contextual $\mathcal{T}$-algebra, $F : \mathcal{R} \to \mathcal{C}$ be the CwF of renamings of $\mathcal{C}$ and $G : \mathcal{C} \to K^*(\mathcal{S})$ be a CwF morphism.*

Then, internally to $\mathbf{Psh}(\mathbf{Ren}(\mathcal{C}))$, the $\mathcal{T}$-algebra $\mathcal{S}_{FG}$, trivially displayed over itself, is initial among $\mathcal{T}$-algebras with variables under $\Delta\mathcal{S}$.

Proof. First remark that $(\mathcal{S}, \mathsf{id}_{\mathcal{S}})$ is the initial algebra of the GAT $\mathcal{T}^{\mathsf{cxl}}[\backslash\mathcal{S}]$. We can construct a SOGAT $\mathcal{T}[\backslash\mathcal{S}]$ such that $(\mathcal{T}[\backslash\mathcal{S}])^{\mathsf{cxl}} \cong \mathcal{T}^{\mathsf{cxl}}[\backslash\mathcal{S}]$; indeed $\mathcal{T}^{\mathsf{cxl}}[\backslash\mathcal{S}]$ was constructed by adding terms and equations to $\mathcal{T}^{\mathsf{cxl}}$, and we can just add the same terms and equations to $\mathcal{T}$, at the correct sorts.

By applying theorem 6.3.10 to $\mathcal{T}[\backslash\mathcal{S}]$, we get a relative induction principle for $\mathcal{S}_{FG}$ (as an internal $\mathcal{T}[\backslash\mathcal{S}]$-algebra).

One can show that a $\mathcal{T}[\backslash\mathcal{S}]$-algebra is a $\mathcal{T}$-algebra together with a morphism from $\mathcal{T}$. Thus the relative induction principle of $\mathcal{S}_{FG}$ as an internal $\mathcal{T}[\backslash\mathcal{S}]$-algebra is the induction principle from the statement. □

In some applications, instantiating $\mathsf{var}^\bullet$ can be more difficult than expected: we need to define it for arbitrary values of $A^\bullet$, while we would expect to only define it for $A^\bullet = [\![A]\!]$. The problem is that the relative induction principle exhibits $\mathcal{S}_F$ as the initial algebra of an internal *recursive* GAT. This GAT needs to be recursive because one cannot freely add variables in one step, since the type of some variable may involve other variables.

We can have more control over the recursion by making it explicit using a sequential colimit.

Theorem 6.3.12. *We work in the setting of definition 6.3.9, in the internal language of $\mathbf{Psh}(\mathbf{Ren}(\mathcal{C}))$.*

Observe that there is a endofunctor I over displayed $\mathcal{T}$-algebras with partial variables such that $\mathcal{T}$-algebras with variables are exactly I-algebras. This endofunctor takes a displayed $\mathcal{T}$-algebra with partial variables $\mathcal{S}^\bullet$ and freely extends $\mathcal{S}^\bullet$ by defining the variables whose type exists in $\mathcal{S}^\bullet$.

Note that the initial $\mathcal{T}$-algebra $\mathbf{0}_{\mathcal{T}}$ is also the initial displayed $\mathcal{T}$-algebra with partial variables, in which none of the variables are defined.

Then $\mathcal{S}_{FG}$ is the initial I-algebra, and by Adámek's fixpoint theorem, it can be written as the sequential colimit

$$\mathcal{S}_{FG} \cong \operatorname*{colim}_{n<\omega} I^n(\mathbf{0}_{\mathcal{T}}).$$

Proof. This follows directly from theorem 6.3.10. □

Analogously, in the setting of theorem 6.3.11, we have an isomorphism

$$\mathcal{S}_{FG} \cong \operatorname*{colim}_{n<\omega} I^n(\Delta\mathcal{S}).$$

6.4 Application: normalization for MLTT and decidability of equality

Our first application of a relative induction principle is a *normalization proof* for MLTT. For our purposes, normalization means that every term (and type) of the syntax of MLTT admits a normal form, which is moreover unique. In principle, multiple different notions of normal forms could be chosen for different purposes. For example, we could say that every terms has a unique normal form (itself), yielding a trivial (and useless) definition of normal form. Instead, the notion of normal form is chosen so as to distinguish unequal terms and permit a proof of *decidability of equality*. Normal forms will be defined as an inductive family over terms. Normal forms are stable under renamings, but not under arbitrary substitutions. Indeed, the normal form of a variable is always just a variable, but a term with an arbitrary normal form can be substituted for a variable. This motivates the use of the relative induction principle (theorem 6.3.10) for the functor $F : \mathbf{Ren}(\mathcal{S}) \to \mathcal{S}$ which sends renamings to substitutions.

The normalization proof follows the structure of other algebraic and reduction-free normalization proofs for dependent type theory (Altenkirch and Kaposi 2017; Kaposi 2017; Coquand 2019; Sterling 2022). It is closely related to normalization-by-evaluation (NbE). This means that normalization is performed in two steps. First, terms are evaluated into a semantic domain (corresponding to a logical relation interpretation). Secondly, elements of this semantic domain are mapped to a normal form for the original term.

The uniqueness of normal forms follows from a property called stability of the normalization function, which is proven by induction on normal forms. The proof of this property relies on the computational behavior of the normalization function.

We will proceed to prove decidability of equality for the syntax. Intuitively, decidability of equality is a direct consequence of normalization: normal forms "obviously" have decidable equality, and normalization implies that to decide equalities between terms, it suffices to decide equalities between their normal forms. Making this argument formal is however not straightforward. One reason is that we define normal forms and prove normalization in the internal language of a presheaf category. But decidability of equality holds externally but not internally, and relating the internal normal forms to an external notion of normal form is not easy: it involves computing the external components of internal indexed inductive types; a task that seems "obvious" but is hard to make formal. We will avoid external normal forms by working internally with the notion of levelwise decidable propositions.

We work in the internal language of $\mathbf{Psh}(\mathbf{Ren}(\mathcal{S}))$. We have access to the relativized syntax $\mathcal{S}_F$, which satisfies the corresponding relative induction principle (theorem 6.3.10).

6.4.1 Normal forms

We first have to define normal forms (and normal types). They are defined at the same time as *neutral forms*. The neutral forms are the terms that behave like variables in

normalization. For MLTT, the neutral forms are spines of eliminators, blocked on a variable.

There is no general procedure that characterizes the neutral and normal forms for a given type theory. As a heuristic, the neutral terms are the terms are indistinguishable from variables in normal forms. Normal forms are not stable under arbitrary substitutions, but they are stable under renamings, which are substitutions built from variables. They should also be stable under neutral substitutions, which are substitutions build from terms which are neutral. Moreover the neutral substitutions should be the maximal class of substitution that stabilizes the normal forms. This cannot be used as a definition of neutral form, since the definition of normal form depends on the notion of neutral form.

The normal types, neutral forms and normal terms are defined as inductive families

$$\begin{aligned}
&\mathsf{NfTy} : \mathcal{S}_F.\mathsf{Ty}_n(1) \to \mathsf{Set},\\
&\mathsf{Ne}_A : \mathcal{S}_F.\mathsf{Tm}(1, A) \to \mathsf{Set},\\
&\mathsf{Nf}_A : \mathcal{S}_F.\mathsf{Ty}(1, A) \to \mathsf{Set}.
\end{aligned}$$

Any variable is a neutral term:

$$\mathsf{var}^{\mathsf{ne}} : (a^{\mathsf{var}} : \mathsf{Var}(A)) \to \mathsf{Ne}_A(\mathsf{var}(a^{\mathsf{var}})).$$

We then add some constructors to these families for the type and term formers of MLTT. The general heuristic is the following:

- Constructors have a corresponding normal form.
- Eliminators have a corresponding neutral form.
- Neutral forms for terms with a positive type (type structures without an η-rule) can be turned into normal forms.

Lifting

$$\begin{aligned}
&\mathsf{lower}^{\mathsf{ne}} : \mathsf{NfTy}_n(A) \to \mathsf{Ne}_{\mathsf{Lift}_n A}(a) \to \mathsf{Ne}_A(\mathsf{lower}(a)),\\
&\mathsf{Lift}_n^{\mathsf{nfty}} : \mathsf{NfTy}_n(A) \to \mathsf{NfTy}_{n+1}(\mathsf{Lift}_n(A)),\\
&\mathsf{lift}^{\mathsf{nf}} : \mathsf{NfTy}_n(A) \to \mathsf{Nf}_A(a) \to \mathsf{Nf}_{\mathsf{Lift}_n(A)}(\mathsf{lift}(a)).
\end{aligned}$$

Universes

$$\begin{aligned}
&\mathcal{U}_n^{\mathsf{nfty}} : \mathsf{NfTy}_{n+1}(\mathcal{U}_n),\\
&\mathsf{El}^{\mathsf{nfty}} : \mathsf{Ne}_{\mathcal{U}_n}(A) \to \mathsf{NfTy}_n(\mathsf{El}(A)),\\
&\mathsf{ne}_{\mathsf{El}}^{\mathsf{nf}} : \mathsf{Ne}_{\mathcal{U}_n}(A) \to \mathsf{Ne}_{\mathsf{El}(A)}(a) \to \mathsf{Nf}_{\mathsf{El}(A)}(a),\\
&\mathsf{nfty}_{\mathcal{U}_n}^{\mathsf{nf}} : \mathsf{NfTy}_n(\mathsf{El}(A)) \to \mathsf{Nf}_{\mathcal{U}_n}(A).
\end{aligned}$$

1-type

$$\begin{aligned}
&\mathbf{1}^{\mathsf{nfty}} : \mathsf{NfTy}(\mathbf{1}),\\
&\mathsf{tt}^{\mathsf{nf}} : \mathsf{Nf}_{\mathbf{1}}(\mathsf{tt}).
\end{aligned}$$

Empty type

$$\begin{aligned}
&\mathsf{Empty}^{\mathsf{nfty}} : \mathsf{NfTy}(\mathsf{Empty}),\\
&\mathsf{absurd}^{\mathsf{ne}} : \mathsf{Ne}_{\mathsf{Empty}}(x) \to \mathsf{Ne}(\mathsf{absurd}(x)),\\
&\mathsf{ne}^{\mathsf{nf}}_{\mathsf{Empty}} : \mathsf{Ne}_{\mathsf{Empty}}(a) \to \mathsf{Nf}_{\mathsf{Empty}}(a).
\end{aligned}$$

Boolean type

$$\begin{aligned}
&\mathsf{Bool}^{\mathsf{nfty}} : \mathsf{NfTy}(\mathsf{Bool}),\\
&\mathsf{elimBool} : (\forall b^{\mathsf{var}} \to \mathsf{NfTy}(P[\mathsf{var}(b)])) \to \mathsf{Nf}(t) \to \mathsf{Nf}(f) \to\\
&\quad \mathsf{Ne}_{\mathsf{Bool}}(b) \to \mathsf{Ne}_{P[b]}(\mathsf{elimBool}(P,t,f,b)),\\
&\mathsf{true}^{\mathsf{nf}} : \mathsf{Nf}(\mathsf{true}),\\
&\mathsf{false}^{\mathsf{nf}} : \mathsf{Nf}(\mathsf{false}),\\
&\mathsf{ne}^{\mathsf{nf}}_{\mathsf{Bool}} : \mathsf{Ne}_{\mathsf{Bool}}(x) \to \mathsf{Ne}_{\mathsf{Bool}}(x).
\end{aligned}$$

Σ-types

$$\begin{aligned}
&\Sigma^{\mathsf{nfty}} : \mathsf{NfTy}(A) \to (\forall a^{\mathsf{var}} \to \mathsf{NfTy}(B[\mathsf{var}(a^{\mathsf{var}})])) \to \mathsf{NfTy}(\Sigma(A,B)),\\
&\mathsf{fst}^{\mathsf{ne}} : \mathsf{Ne}_{\Sigma(A,B)}(p) \to \mathsf{Ne}_A(\mathsf{fst}(p)),\\
&\mathsf{snd}^{\mathsf{ne}} : \mathsf{Ne}_{\Sigma(A,B)}(p) \to \mathsf{Ne}_{B[\mathsf{fst}(p)]}(\mathsf{snd}(p)),\\
&\mathsf{pair}^{\mathsf{nf}} : \mathsf{Ne}_{\Sigma(A,B)}(p) \to \mathsf{Ne}_{B[\mathsf{fst}(p)]}(\mathsf{snd}(p)).
\end{aligned}$$

Π-types

$$\begin{aligned}
&\Pi^{\mathsf{nfty}} : \mathsf{NfTy}(A) \to (\forall a^{\mathsf{var}} \to \mathsf{NfTy}(B[\mathsf{var}(a^{\mathsf{var}})])) \to \mathsf{NfTy}(\Pi(A,B)),\\
&\mathsf{app}^{\mathsf{ne}} : \mathsf{Ne}_{\Pi(A,B)}(f) \to \mathsf{Nf}_A(x) \to \mathsf{Ne}_{B[x]}(\mathsf{app}(f,x)),\\
&\mathsf{lam}^{\mathsf{nf}} : (\forall a^{\mathsf{var}} \to \mathsf{Nf}_{B[\mathsf{var}(a^{\mathsf{var}})]}(b[\mathsf{var}(a^{\mathsf{var}})])) \to \mathsf{Nf}_{\Pi(A,B)}(\mathsf{lam}(b)).
\end{aligned}$$

Id-types

$$\begin{aligned}
&\mathsf{Id}^{\mathsf{nfty}} : \mathsf{NfTy}(A) \to \mathsf{Nf}_A(x) \to \mathsf{Nf}_A(y) \to \mathsf{NfTy}(\mathsf{Id}(A,x,y)),\\
&\mathsf{J}^{\mathsf{ne}} : \mathsf{NfTy}(A) \to \mathsf{Nf}_A(x) \to\\
&\quad (\forall y^{\mathsf{var}}\, p^{\mathsf{var}} \to \mathsf{NfTy}(P[\mathsf{var}(y^{\mathsf{var}}), \mathsf{var}(p^{\mathsf{var}})])) \to \mathsf{Nf}_{P[x,\mathsf{refl}(A,x)]}(d) \to\\
&\quad \mathsf{Nf}(y) \to \mathsf{Ne}_{\mathsf{Id}(A,x,y)}(p) \to \mathsf{Ne}_{P[y,p]}(\mathsf{J}(A,x,P,d,y,p)),\\
&\mathsf{refl}^{\mathsf{nf}} : \mathsf{NfTy}(A) \to \mathsf{Nf}_A(x) \to \mathsf{Nf}_{\mathsf{Id}(A,x,x)}(\mathsf{refl}(A,x)),\\
&\mathsf{ne}^{\mathsf{nf}}_{\mathsf{Id}} : \mathsf{NfTy}(A) \to \mathsf{Nf}_A(x) \to \mathsf{Nf}_A(y) \to \mathsf{Ne}_{\mathsf{Id}(A,x,y)}(p) \to \mathsf{Nf}_{\mathsf{Id}(A,x,y)}(p).
\end{aligned}$$

W-types

$$
\begin{aligned}
&W^{\mathsf{nfty}} : \mathsf{NfTy}(A) \to (\forall a^{\mathsf{var}} \to \mathsf{NfTy}(B[\mathsf{var}(a^{\mathsf{var}})])) \to \mathsf{NfTy}(W(A,B)),\\
&\mathsf{elimW}^{\mathsf{ne}} : \mathsf{NfTy}(A) \to (\forall a^{\mathsf{var}} \to \mathsf{NfTy}(B[\mathsf{var}(a^{\mathsf{var}})]) \to\\
&\quad (\forall x^{\mathsf{var}} \to \mathsf{NfTy}(P[\mathsf{var}(x^{\mathsf{var}})])) \to\\
&\quad (\forall a^{\mathsf{var}} f^{\mathsf{var}} \to s[\mathsf{var}(a^{\mathsf{var}}), \mathsf{var}(f^{\mathsf{var}})]) \to\\
&\quad \mathsf{Ne}_{W(A,B)}(x) \to \mathsf{Ne}_{P[x]}(\mathsf{elimW}(A,B,P,s,x))),\\
&\mathsf{sup}^{\mathsf{nf}} : \mathsf{Nf}_A(a) \to \mathsf{Nf}_{B[a]\to W(A,B)}(f) \to \mathsf{Nf}_{W(A,B)}(\mathsf{sup}(a,f)),\\
&\mathsf{ne}^{\mathsf{nf}}_W : \mathsf{NfTy}(A) \to (\forall a^{\mathsf{var}} \to \mathsf{NfTy}(B[\mathsf{var}(a^{\mathsf{var}})])) \to \mathsf{Ne}_{W(A,B)}(x) \to \mathsf{Nf}_{W(A,B)}(x).
\end{aligned}
$$

6.4.2 The normalization displayed higher-order model

We construct a displayed higher-order model Norm of $\mathcal{T}_{\mathsf{MLTT}}$ over the relativized syntax $\mathcal{S}_F$.

- A displayed type $A^\bullet$ over a closed n-small type $A : \mathcal{S}_F.\mathsf{Ty}_n(1)$ is a **n-small normalization family**, that is a tuple $(A^\bullet.\mathsf{pred}, A^\bullet.\mathsf{nfty}, A^\bullet.\mathsf{reflect}, A^\bullet.\mathsf{reify})$ consisting of the following components:
  - $A^\bullet.\mathsf{pred} : \mathcal{S}_F.\mathsf{Tm}(1, A) \to \mathsf{Set}_n$ is a unary logical predicate over closed terms of type A.
  - $A^\bullet.\mathsf{nfty} : \mathsf{NfTy}_n(A)$ is a normal form for the type A.
  - $A^\bullet.\mathsf{reflect} : (a : \mathcal{S}_F.\mathsf{Tm}(1, A)) \to \mathsf{Ne}_A(a) \to A^\bullet.\mathsf{pred}(a)$ is a function showing that neutral forms satisfy the predicate $A^\bullet.\mathsf{pred}$.
  - $A^\bullet.\mathsf{reify} : (a : \mathcal{S}_F.\mathsf{Tm}(1, A)) \to A^\bullet.\mathsf{pred}(a) \to \mathsf{Nf}_A(a)$ is a function showing that terms satisfying the predicate $A^\bullet.\mathsf{pred}$ admit a normal form.
- A displayed term $a^\bullet$ of type $A^\bullet$ over a closed term $a : \mathcal{S}_F.\mathsf{Tm}(1, A)$ is an element

$$a^\bullet : A^\bullet.\mathsf{pred}(a).$$

Lifting

The normalization family $\mathsf{Lift}^\bullet_n(A^\bullet)$ is defined using the corresponding components of $A^\bullet$.

$$
\begin{aligned}
&\mathsf{Lift}^\bullet_n(A^\bullet).\mathsf{nfty} = \mathsf{Lift}^{\mathsf{nf}}(A^\bullet.\mathsf{nfty}),\\
&\mathsf{Lift}^\bullet_n(A^\bullet).\mathsf{pred}(x) = A^\bullet.\mathsf{pred}(\mathsf{lower}(x)),\\
&\mathsf{Lift}^\bullet_n(A^\bullet).\mathsf{reflect}(x^{\mathsf{ne}}) = A^\bullet.\mathsf{reflect}(\mathsf{lower}^{\mathsf{ne}}(x^{\mathsf{ne}})),\\
&\mathsf{Lift}^\bullet_n(A^\bullet).\mathsf{reify}(x^\bullet) = \mathsf{lift}^{\mathsf{nf}}(A^\bullet.\mathsf{reify}(x^\bullet)).
\end{aligned}
$$

Note that the definition of the logical predicate uses implicit lifting from Set_n to Set_{n+1} in the metatheory.

Universes

The normal form of the universe $\mathcal{U}_n$ is $\mathcal{U}_n^{\mathsf{nfty}}$.

$$\mathcal{U}_n^\bullet.\mathsf{nfty} = \mathcal{U}_n^{\mathsf{nfty}}.$$

For any $A : \mathcal{S}_F.\mathsf{Tm}(1, \mathcal{U}_n)$, the set $\mathcal{U}_n^\bullet.\mathsf{pred}(A)$ is defined as the set of n-small normalization structures over the type $\mathsf{El}(A)$. This set is $(n+1)$-small.

For any neutral term $A^{\mathsf{ne}} : \mathsf{Ne}_{\mathcal{U}_n}(A)$, $\mathcal{U}_n^\bullet.\mathsf{reflect}(A^{\mathsf{ne}})$ should be a n-small normalization structure over $\mathsf{El}(A)$:

$$\begin{aligned}
&\mathcal{U}_n^\bullet.\mathsf{reflect}(A^{\mathsf{ne}}).\mathsf{nfty} = \mathsf{El}^{\mathsf{nfty}}(A^{\mathsf{ne}}),\\
&\mathcal{U}_n^\bullet.\mathsf{reflect}(A^{\mathsf{ne}}).\mathsf{pred} = \lambda a \mapsto \mathsf{Ne}_{\mathsf{El}(A)}(a),\\
&\mathcal{U}_n^\bullet.\mathsf{reflect}(A^{\mathsf{ne}}).\mathsf{reflect} = \lambda a^{\mathsf{ne}} \mapsto a^{\mathsf{ne}},\\
&\mathcal{U}_n^\bullet.\mathsf{reflect}(A^{\mathsf{ne}}).\mathsf{reify} = \lambda a^{\mathsf{ne}} \mapsto \mathsf{ne}^{\mathsf{nf}}_{\mathsf{El}}(A^{\mathsf{ne}}, a^{\mathsf{ne}}).
\end{aligned}$$

The function $\mathcal{U}_n^\bullet.\mathsf{reify}$ then projects the normal type out of the n-small normalization structure.

$$\mathcal{U}_n^\bullet.\mathsf{reify} = \lambda A^\bullet \mapsto \mathsf{nfty}^{\mathsf{nf}}_{\mathcal{U}_n}(A^\bullet.\mathsf{nfty}).$$

1-type

The definition of the displayed **1**-type is straightforward.

$$\begin{aligned}
&\mathbf{1}^\bullet.\mathsf{nfty} = \mathbf{1}^{\mathsf{nfty}},\\
&\mathbf{1}^\bullet.\mathsf{pred} = \lambda - \mapsto \{\star\},\\
&\mathbf{1}^\bullet.\mathsf{reflect} = \lambda - \mapsto \star,\\
&\mathbf{1}^\bullet.\mathsf{reify} = \lambda - \mapsto \mathsf{tt}^{\mathsf{nf}}.
\end{aligned}$$

Boolean types

The normal form of Bool is $\mathsf{Bool}^{\mathsf{nfty}}$.

$$\mathsf{Bool}^\bullet.\mathsf{nfty} = \mathsf{Bool}^{\mathsf{nfty}}.$$

The logical predicate $\mathsf{Bool}^\bullet.\mathsf{pred}$ is inductively generated by three constructors.

$$\begin{aligned}
&\mathsf{Bool}^\bullet.\mathsf{pred} : \mathcal{S}_F.\mathsf{Tm}(1, \mathsf{Bool}) \to \mathsf{Set},\\
&\mathsf{reflect}_{\mathsf{Bool}} : \mathsf{Ne}_{\mathsf{Bool}}(b) \to \mathsf{Bool}^\bullet.\mathsf{pred}(b),\\
&\mathsf{true}^\bullet : \mathsf{Bool}^\bullet.\mathsf{pred}(\mathsf{true}),\\
&\mathsf{false}^\bullet : \mathsf{Bool}^\bullet.\mathsf{pred}(\mathsf{false}).
\end{aligned}$$

Remark that (the total space of) $\mathsf{Bool}^\bullet.\mathsf{pred}$ is the free bipointed set over the set of neutral forms of type Bool.

The reflection function is the constructor $\mathsf{reflect}_{\mathsf{Bool}}$.

The reification function for booleans is defined by induction over $\mathsf{Bool}^\bullet.\mathsf{pred}$:

$$\begin{aligned}
&\mathsf{Bool}^\bullet.\mathsf{reify}(\mathsf{reflect}_{\mathsf{Bool}}(b^{\mathsf{ne}})) = \mathsf{ne}^{\mathsf{nf}}_{\mathsf{Bool}}(b^{\mathsf{ne}}),\\
&\mathsf{Bool}^\bullet.\mathsf{reify}(\mathsf{true}^\bullet) = \mathsf{true}^{\mathsf{nf}},\\
&\mathsf{Bool}^\bullet.\mathsf{reify}(\mathsf{false}^\bullet) = \mathsf{false}^{\mathsf{nf}}.
\end{aligned}$$

The displayed eliminator $\mathsf{elimBool}^\bullet$ is defined by induction on $\mathsf{Bool}^\bullet.\mathsf{pred}$:

$$\begin{aligned}
&\mathsf{elimBool}^\bullet(P^\bullet, t^\bullet, f^\bullet, \mathsf{reflect}_{\mathsf{Bool}}(b^{\mathsf{ne}})) = \mathsf{elimBool}^{\mathsf{ne}}(\ldots, b^{\mathsf{ne}}),\\
&\mathsf{elimBool}^\bullet(P^\bullet, t^\bullet, f^\bullet, \mathsf{true}^\bullet) = t^\bullet,\\
&\mathsf{elimBool}^\bullet(P^\bullet, t^\bullet, f^\bullet, \mathsf{false}^\bullet) = f^\bullet.
\end{aligned}$$

Empty types

The normal form of Empty is $\mathsf{Empty}^{\mathsf{nfty}}$.

$$\mathsf{Empty}^\bullet.\mathsf{nfty} = \mathsf{Empty}^{\mathsf{nfty}}.$$

The logical predicate $\mathsf{Empty}^\bullet.\mathsf{pred}$ is inductively generated by a single constructor.

$$\begin{aligned}
&\mathsf{Empty}^\bullet.\mathsf{pred} : \mathcal{S}_F.\mathsf{Tm}(1, \mathsf{Empty}) \to \mathsf{Set},\\
&\mathsf{reflect}_{\mathsf{Empty}} : \mathsf{Ne}_{\mathsf{Empty}}(x) \to \mathsf{Empty}^\bullet.\mathsf{pred}(x).
\end{aligned}$$

The reification function for booleans is defined by induction over $\mathsf{Empty}^\bullet.\mathsf{pred}$:

$$\mathsf{Empty}^\bullet.\mathsf{reify}(\mathsf{reflect}_{\mathsf{Empty}}(x^{\mathsf{ne}})) = \mathsf{ne}^{\mathsf{nf}}_{\mathsf{Empty}}(x^{\mathsf{ne}}).$$

The displayed eliminator $\mathsf{absurd}^\bullet$ is defined by induction over $\mathsf{Empty}^\bullet.\mathsf{pred}$:

$$\mathsf{absurd}^\bullet(P^\bullet, \mathsf{reflect}_{\mathsf{Empty}}(x^{\mathsf{ne}})) = \mathsf{absurd}^{\mathsf{ne}}(\ldots, b^{\mathsf{ne}}).$$

Σ-types

The definition of the logical predicate for Σ-type is forced by the required isomorphism

$$\mathsf{Tm}^\bullet(\Sigma^\bullet(A^\bullet, B^\bullet), p) \cong (a^\bullet : \mathsf{Tm}^\bullet(A^\bullet, \mathsf{fst}(p))) \times \mathsf{Tm}^\bullet(B^\bullet(a^\bullet), \mathsf{snd}(p)).$$

$$\begin{aligned}
&\Sigma^\bullet(A^\bullet, B^\bullet).\mathsf{nfty} = \Sigma^{\mathsf{nfty}}(A^\bullet.\mathsf{nfty}, \lambda a^{\mathsf{var}} \mapsto B^\bullet(\mathsf{var}^{\mathsf{ne}}(a^{\mathsf{var}})).\mathsf{nfty}),\\
&\Sigma^\bullet(A^\bullet, B^\bullet).\mathsf{pred} = \lambda p \mapsto (a^\bullet : A^\bullet.\mathsf{pred}(\mathsf{fst}(p))) \times (b^\bullet : B^\bullet(a^\bullet).\mathsf{pred}(\mathsf{snd}(p))).
\end{aligned}$$

Then the reflection operation is defined using the reflection operations of $A^\bullet$ and $B^\bullet$, and the reification operation is defined using the reification operations of $A^\bullet$ and $B^\bullet$.

$$\begin{aligned}
&\Sigma^\bullet(A^\bullet, B^\bullet).\mathsf{reflect} = \lambda p^{\mathsf{ne}} \mapsto (a^\bullet, B^\bullet(a^\bullet).\mathsf{reflect}(\mathsf{snd}^{\mathsf{ne}}(p^{\mathsf{ne}}))),\\
&\quad \text{where } a^\bullet = A^\bullet.\mathsf{reflect}(\mathsf{fst}^{\mathsf{ne}}(p^{\mathsf{ne}})),\\
&\Sigma^\bullet(A^\bullet, B^\bullet).\mathsf{reify} = \lambda(a^\bullet, b^\bullet) \mapsto \mathsf{pair}^{\mathsf{nf}}(A^\bullet.\mathsf{reify}(a^\bullet), B^\bullet(a^\bullet).\mathsf{reify}(b^\bullet)).
\end{aligned}$$

Π-types

The definition of the logical predicate for Σ-type is forced by the required isomorphism

$$\mathsf{Tm}^\bullet(\Pi^\bullet(A^\bullet, B^\bullet), f) \cong (\forall a(a^\bullet : \mathsf{Tm}^\bullet(A^\bullet, a)) \to \mathsf{Tm}^\bullet(B^\bullet(a^\bullet), \mathsf{app}(f, a))).$$

$$\begin{aligned}
&\Pi^\bullet(A^\bullet, B^\bullet).\mathsf{nfty} = \Pi^{\mathsf{nfty}}(A^\bullet.\mathsf{nfty}, \lambda a^{\mathsf{var}} \mapsto B^\bullet(\mathsf{var}^{\mathsf{ne}}(a^{\mathsf{var}})).\mathsf{nfty}),\\
&\Pi^\bullet(A^\bullet, B^\bullet).\mathsf{pred} = \lambda f \mapsto \forall a\ (a^\bullet : A^\bullet.\mathsf{pred}(a)) \to (b^\bullet : B^\bullet(a^\bullet).\mathsf{pred}(\mathsf{app}(f, a))),
\end{aligned}$$

The reflection operation is defined using the reification operation of $A^\bullet$ and the reflection operations of $B^\bullet$, whereas the reification operation is defined using the reflection operation of $A^\bullet$ and the reification operation of $B^\bullet$.

$$\begin{aligned}
&\Pi^\bullet(A^\bullet, B^\bullet).\mathsf{reflect} = \lambda f^{\mathsf{ne}} \mapsto \lambda a^\bullet \mapsto B^\bullet(a^\bullet).\mathsf{reflect}(\mathsf{app}^{\mathsf{ne}}(f^{\mathsf{ne}}, a^{\mathsf{nf}})),\\
&\quad \text{where } a^{\mathsf{nf}} = A^\bullet.\mathsf{reify}(a^\bullet),\\
&\Pi^\bullet(A^\bullet, B^\bullet).\mathsf{reify} = \lambda f^\bullet \mapsto \mathsf{lam}^{\mathsf{nf}}(\lambda a^{\mathsf{var}} \mapsto B^\bullet(a^\bullet).\mathsf{reify}(f^\bullet(a^\bullet))),\\
&\quad \text{where } a^\bullet = A^\bullet.\mathsf{reflect}(\mathsf{var}^{\mathsf{ne}}(a^{\mathsf{var}})).
\end{aligned}$$

Id-types

$$\mathsf{Id}^\bullet(A^\bullet, x^\bullet, y^\bullet).\mathsf{nfty} = \mathsf{Id}^{\mathsf{nfty}}(A^\bullet.\mathsf{nfty}, A^\bullet.\mathsf{reify}(x^\bullet), A^\bullet.\mathsf{reify}(y^\bullet)).$$

The logical predicate $\mathsf{Id}^\bullet(A^\bullet, x^\bullet, y^\bullet).\mathsf{pred}$ is defined as an indexed inductive family with two constructors.

$$\begin{aligned}
&\mathsf{Id}^\bullet(A^\bullet, x^\bullet, -).\mathsf{pred}(-)\\
&\quad : (y : \mathcal{S}_F.\mathsf{Tm}(1, A))(y^\bullet : A^\bullet.\mathsf{pred}(y)) \to \mathcal{S}_F.\mathsf{Tm}(1, \mathsf{Id}(A, x, y)) \to \mathsf{Set},\\
&\mathsf{reflect}_{\mathsf{Id}} : \forall y^\bullet \to \mathsf{Ne}_{\mathsf{Id}(A,x,y)}(p) \to \mathsf{Id}^\bullet(A^\bullet, x^\bullet, y^\bullet).\mathsf{pred}(p),\\
&\mathsf{refl}^\bullet : \mathsf{Id}^\bullet(A^\bullet, x^\bullet, x^\bullet).\mathsf{pred}(\mathsf{refl}(A, x)).
\end{aligned}$$

The reflection function is the constructor $\mathsf{reflect}_{\mathsf{Id}}$.
The reification function is defined by induction over that family:

$$\begin{aligned}
&\mathsf{Id}^\bullet(A^\bullet, x^\bullet, y^\bullet).\mathsf{reify}(\mathsf{reflect}_{\mathsf{Id}}(p^{\mathsf{ne}})) = \mathsf{ne}^{\mathsf{nf}}_{\mathsf{Id}}(p^{\mathsf{ne}}),\\
&\mathsf{Id}^\bullet(A^\bullet, x^\bullet, x^\bullet).\mathsf{reify}(\mathsf{refl}^\bullet) = \mathsf{refl}^{\mathsf{nf}}(\dots).
\end{aligned}$$

The identity type eliminator is also defined by induction over it:

$$\begin{aligned}
&\mathsf{J}^\bullet(\dots, \mathsf{reflect}_{\mathsf{Id}}(p^{\mathsf{ne}})) = \mathsf{J}^{\mathsf{ne}}(\dots, p^{\mathsf{ne}}),\\
&\mathsf{J}^\bullet(\dots, d^\bullet, \mathsf{refl}^\bullet) = d^\bullet.
\end{aligned}$$

W-types
The normal form of a W-type is defined in the same way as for Σ- and Π- types.

$$W^\bullet(A^\bullet, B^\bullet).\mathsf{nfty} = W^{\mathsf{nfty}}(A^\bullet.\mathsf{nfty}, \lambda a^{\mathsf{var}} \mapsto B^\bullet(\mathsf{var}^{\mathsf{ne}}(a^{\mathsf{var}})).\mathsf{nfty}).$$

The logical predicate $W^\bullet(A^\bullet, B^\bullet).\mathsf{pred}$ is defined as an indexed inductive family with two constructors.

$$\begin{aligned}
&W^\bullet(A^\bullet, B^\bullet).\mathsf{pred}(-)\\
&\quad : (x : \mathcal{S}_F.\mathsf{Tm}(1, W(A, B))) \to \mathsf{Set},\\
&\mathsf{reflect}_W : \mathsf{Ne}_{W(A,B)}(x) \to W^\bullet(A^\bullet, B^\bullet).\mathsf{pred}(x),\\
&\mathsf{sup}^\bullet : (a^\bullet : A^\bullet.\mathsf{pred}(a))(f^\bullet : \forall b\ (b^\bullet : B^\bullet(a^\bullet).\mathsf{pred}(b)) \to W^\bullet(A^\bullet, B^\bullet).\mathsf{pred}(\mathsf{app}(f, b)))\\
&\quad \to W^\bullet(A^\bullet, B^\bullet).\mathsf{pred}(\mathsf{sup}(a, f))
\end{aligned}$$

The reflection function is the constructor $\mathsf{reflect}_W$.

The reification function is defined by induction:

$$W^\bullet(A^\bullet, B^\bullet).\mathsf{reify}(\mathsf{reflect}_W(x^{\mathsf{ne}})) = \mathsf{ne}^{\mathsf{nf}}_W(x^{\mathsf{ne}}),$$
$$W^\bullet(A^\bullet, B^\bullet).\mathsf{reify}(\mathsf{sup}(a^\bullet, f^\bullet))$$
$$= \mathsf{sup}^{\mathsf{nf}}(A^\bullet.\mathsf{reify}(a^\bullet), \lambda b^{\mathsf{var}} \mapsto W^\bullet(A^\bullet, B^\bullet).\mathsf{reify}(f^\bullet(B^\bullet(a^\bullet).\mathsf{reflect}(\mathsf{var}^{\mathsf{ne}}(b^{\mathsf{var}})))))$$

The eliminator is defined using the universal property of $W^\bullet(A^\bullet, B^\bullet).\mathsf{pred}(-)$.

$$\begin{aligned}
&\mathsf{elimW}^\bullet(P^\bullet, s^\bullet, \mathsf{reflect}_W(x^{\mathsf{ne}}))\\
&\quad \triangleq P^\bullet(\mathsf{reflect}_W(x^{\mathsf{ne}})).\mathsf{reflect}(\mathsf{elimW}^{\mathsf{ne}}(P^{\mathsf{nfty}}, s^{\mathsf{nf}}, x^{\mathsf{ne}})),\\
&\quad \text{where } P^{\mathsf{nfty}} \triangleq \lambda x^{\mathsf{var}} \mapsto P^\bullet(\mathsf{reflect}_W(\mathsf{var}^{\mathsf{ne}}(x^{\mathsf{var}}))).\mathsf{nfty},\\
&\qquad\quad s^{\mathsf{nf}} \triangleq \lambda a^{\mathsf{var}}\, f^{\mathsf{var}} \mapsto P^\bullet(\mathsf{sup}^\bullet(a^\bullet, f^\bullet)).\mathsf{reify}(s^\bullet(a^\bullet, f^\bullet)),\\
&\qquad\quad a^\bullet \triangleq A^\bullet.\mathsf{reflect}(\mathsf{var}^{\mathsf{ne}}(a^{\mathsf{var}})),\\
&\qquad\quad f^\bullet \triangleq \lambda b\, b^\bullet \mapsto \mathsf{reflect}_W(\mathsf{app}^{\mathsf{ne}}(\mathsf{var}^{\mathsf{ne}}(f^{\mathsf{var}}), B^\bullet(a^\bullet).\mathsf{reify}(b^\bullet)))\\
&\mathsf{elimW}^\bullet(P^\bullet, s^\bullet, \mathsf{sup}^\bullet(a^\bullet, f^\bullet))\\
&\quad \triangleq s^\bullet(a^\bullet, \lambda b^\bullet \mapsto \mathsf{elimW}^\bullet(P^\bullet, s^\bullet, f^\bullet(b^\bullet))).
\end{aligned}$$

6.4.3 Normalization function

Note that we can define a displayed operation over variable terms:

$$\mathsf{var}^\bullet : (A^\bullet : \mathsf{Ty}^\bullet(A))(a^{\mathsf{var}} : \mathsf{Var}(A)) \to \mathsf{Tm}^\bullet(A^\bullet, \mathsf{var}(a^{\mathsf{var}})),$$
$$\mathsf{var}^\bullet(A^\bullet, a^{\mathsf{var}}) \triangleq A^\bullet.\mathsf{reflect}(\mathsf{var}^{\mathsf{ne}}(a^{\mathsf{var}})).$$

The universal property of $\mathcal{S}_F$ thus implies that we have a section $[\![-]\!]$ of the displayed contextualization $\mathsf{Cxl}(\mathsf{Norm})$, such that $[\![\mathsf{var}_A(a^{\mathsf{var}})]\!] = \mathsf{var}^\bullet([\![A]\!], a^{\mathsf{var}})$.

We define normalization functions:

$$\mathsf{norm}_{\mathsf{ty}} : (A : \mathsf{Ty}) \to \mathsf{NfTy}(A),$$
$$\mathsf{norm}_{\mathsf{tm}} : (A : \mathsf{Ty})(a : \mathsf{Tm}(A)) \to \mathsf{Nf}_A(a),$$
$$\mathsf{norm}_{\mathsf{ty}}(A) = [\![A]\!].\mathsf{nfty},$$
$$\mathsf{norm}_{\mathsf{tm}}(A, a) = [\![A]\!].\mathsf{reify}([\![a]\!]).$$

6.4.4 Uniqueness of normal forms

Now that we have normalization functions, establishing the existence of normal forms, we can prove uniqueness by induction on normal forms. Each case relies on some of the computation rules of the normalization function.

Lemma 6.4.1. *The following three facts hold:*

- *For every* $A^{\mathsf{nfty}} : \mathsf{NfTy}(A)$, *we have* $\mathsf{norm}_{\mathsf{ty}}(A) = A^{\mathsf{nfty}}$.
- *For every* $a^{\mathsf{nf}} : \mathsf{Nf}_A(a)$, *we have* $\mathsf{norm}_{\mathsf{tm}}(a) = a^{\mathsf{nf}}$.
- *For every* $a^{\mathsf{ne}} : \mathsf{Ne}_A(a)$, *we have* $[\![A]\!].\mathsf{reflect}(a^{\mathsf{ne}}) = [\![a]\!]$.

Proof. By mutual induction on NfTy, Ne and Nf. We only list some of the cases.

Constructor $\mathsf{var}^{\mathsf{ne}}$

$$\begin{aligned}
&[\![A]\!].\mathsf{reflect}(\mathsf{var}^{\mathsf{ne}}(a^{\mathsf{var}})) \\
&\quad = \mathsf{var}^{\bullet}([\![A]\!], a^{\mathsf{var}}) && \text{(Definition of } \mathsf{var}^{\bullet}\text{)} \\
&\quad = [\![\mathsf{var}_A(a)]\!] && \text{(Computation rule of } [\![-]\!]\text{)}
\end{aligned}$$

Constructor $\mathsf{app}^{\mathsf{ne}}$

$$\begin{aligned}
&[\![B(a)]\!].\mathsf{reflect}(\mathsf{app}^{\mathsf{ne}}(f^{\mathsf{ne}}, a^{\mathsf{nf}})) \\
&\quad = [\![B]\!]([\![a]\!]).\mathsf{reflect}(\mathsf{app}^{\mathsf{ne}}(f^{\mathsf{ne}}, a^{\mathsf{nf}})) \\
&\qquad\qquad \text{(Computation rule of } [\![-]\!] \text{ on substituted types)} \\
&\quad = [\![B]\!]([\![a]\!]).\mathsf{reflect}(\mathsf{app}^{\mathsf{ne}}(f^{\mathsf{ne}}, [\![A]\!].\mathsf{reify}([\![a]\!]))) && \text{(Induction hypothesis for } a^{\mathsf{nf}}\text{)} \\
&\quad = [\![\Pi(A,B)]\!].\mathsf{reflect}(f^{\mathsf{ne}}, a^{\bullet}) && \text{(Computation rule for } [\![-]\!] \text{ on } \Pi\text{)} \\
&\quad = [\![f]\!]([\![a]\!]) && \text{(Induction hypothesis for } f^{\mathsf{ne}}\text{)} \\
&\quad = [\![\mathsf{app}(f,a)]\!] && \text{(Computation rule of } [\![-]\!] \text{ on app)}
\end{aligned}$$

Constructor Π^{nfty}

$$\begin{aligned}
&[\![\Pi(A,B)]\!].\mathsf{nfty} \\
&\quad = \Pi^{\mathsf{nfty}}([\![A]\!].\mathsf{nfty}, \lambda a^{\mathsf{var}} \mapsto [\![B]\!]([\![A]\!].\mathsf{reflect}(\mathsf{var}^{\mathsf{ne}}(a^{\mathsf{var}}))).\mathsf{nfty}) \\
&\qquad\qquad \text{(Computation rule of } [\![-]\!] \text{ on } \Pi\text{)} \\
&\quad = \Pi^{\mathsf{nfty}}([\![A]\!].\mathsf{nfty}, \lambda a^{\mathsf{var}} \mapsto [\![B]\!]([\![\mathsf{var}(a^{\mathsf{var}})]\!]).\mathsf{nfty}) \\
&\qquad\qquad \text{(Computation rule of } [\![-]\!] \text{ on variables)} \\
&\quad = \Pi^{\mathsf{nfty}}([\![A]\!].\mathsf{nfty}, \lambda a^{\mathsf{var}} \mapsto [\![B(\mathsf{var}(a^{\mathsf{var}}))]\!].\mathsf{nfty}) \\
&\qquad\qquad \text{(Computation rule of } [\![-]\!] \text{ on substituted types)} \\
&\quad = \Pi^{\mathsf{nfty}}(A^{\mathsf{nfty}}, B^{\mathsf{nfty}}) && \text{(Induction hypotheses for } A^{\mathsf{nfty}} \text{ and } B^{\mathsf{nfty}}\text{)}
\end{aligned}$$

Constructor $\mathsf{lam}^{\mathsf{nf}}$

$$\begin{aligned}
&[\![\Pi(A,B)]\!].\mathsf{reify}([\![\mathsf{lam}(b)]\!]) \\
&\quad = \mathsf{lam}^{\mathsf{nf}}(\lambda a^{\mathsf{var}} \mapsto [\![B]\!]([\![A]\!].\mathsf{reflect}(\mathsf{var}^{\mathsf{ne}}(a^{\mathsf{var}}))).\mathsf{reify}([\![b]\!]([\![A]\!].\mathsf{reflect}(\mathsf{var}^{\mathsf{ne}}(a^{\mathsf{var}}))))) \\
&\qquad\qquad \text{(Computation rule of } [\![-]\!] \text{ on } \Pi \text{ and lam)} \\
&\quad = \mathsf{lam}^{\mathsf{nf}}(\lambda a^{\mathsf{var}} \mapsto [\![B]\!]([\![\mathsf{var}(a^{\mathsf{var}})]\!]).\mathsf{reify}([\![b]\!]([\![\mathsf{var}(a^{\mathsf{var}})]\!]))) \\
&\qquad\qquad \text{(Computation rule of } [\![-]\!] \text{ on variables)} \\
&\quad = \mathsf{lam}^{\mathsf{nf}}(\lambda a^{\mathsf{var}} \mapsto [\![B(\mathsf{var}(a^{\mathsf{var}}))]\!].\mathsf{reify}([\![b(\mathsf{var}(a^{\mathsf{var}}))]\!])) \\
&\qquad\qquad \text{(Computation rule of } [\![-]\!] \text{ on substituted types and terms)} \\
&\quad = \mathsf{lam}^{\mathsf{nf}}(b^{\mathsf{nf}}) && \text{(Induction hypothesis for } b^{\mathsf{nf}}\text{)} \quad \square
\end{aligned}$$

Theorem 6.4.2 (Normalization)**.** *Any type* $A : \mathcal{S}_F.\mathsf{Ty}(1)$ *or term* $a : \mathcal{S}_F.\mathsf{Tm}(1, A)$ *has a unique normal form (element of* $\mathsf{NfTy}(A)$ *or* $\mathsf{Nf}_A(a)$*).*

Proof. Existence is provided by the normalization function $\mathsf{norm}_{\mathsf{ty}}$ and $\mathsf{norm}_{\mathsf{tm}}$, and uniqueness follows from lemma 6.4.1. $\square$

6.4.5 Decidability of equality

Normalization should imply decidability of equality: normal forms are defined as indexed inductive families, and we know how to characterize equalities of such families. Decidability of equality is however not stable under renaming. For example, different variables x, y can become equal after applying the renaming $[x \mapsto y]$. However variables have a levelwise decidable equality, and we can prove internally to $\mathbf{Psh}(\mathcal{C})$ that types and terms also have a levelwise decidable equality. Recall that we have defined levelwise decidable propositions in section 2.3.5. We have proven that variables have levelwise decidable equality in proposition 3.1.17.

Lemma 6.4.3. *The sets*

$$(A : \mathcal{S}_F.\mathsf{Ty}_n(1)) \times \mathsf{NfTy}_n(A),$$
$$(A : \mathcal{S}_F.\mathsf{Ty}_n(1)) \times (a : \mathcal{S}_F.\mathsf{Tm}(1, A)) \times \mathsf{Ne}_A(a),$$
$$(A : \mathcal{S}_F.\mathsf{Ty}_n(1)) \times (a : \mathcal{S}_F.\mathsf{Tm}(1, A)) \times \mathsf{Nf}_A(a)$$

have levelwise decidable equality.

Proof. Write $(\doteq)$ for equality in the total spaces of the families NfTy, Ne and Nf, e.g. $(A_1^{\mathsf{nfty}} \doteq A_2^{\mathsf{nfty}})$ stands for $(A_1, A_1^{\mathsf{nfty}}) = (A_2, A_2^{\mathsf{nfty}})$ in $(A : \mathcal{S}_F.\mathsf{Ty}_n(1)) \times \mathsf{NfTy}_n(A)$.

There are two approaches:

- By mutual double induction on pairs of normal types, neutral types or normal forms, we can recursively characterize equality in the total spaces.

  For example,

  $$(\Pi^{\mathsf{nfty}}(A_1^{\mathsf{nfty}}, B_1^{\mathsf{nfty}}) \doteq \Pi^{\mathsf{nfty}}(A_2^{\mathsf{nfty}}, B_2^{\mathsf{nfty}}))$$
  $$= (A_1^{\mathsf{nfty}} \doteq A_2^{\mathsf{nfty}}) \wedge (\forall a^{\mathsf{var}} \to B_1^{\mathsf{nfty}}(a^{\mathsf{var}}) \doteq B_2^{\mathsf{nfty}}(a^{\mathsf{var}})).$$

  Note that $\wedge$ above is actually dependent: we need to know that $A_1 = A_2$ so as to make a^{var} a variable of that type. Each case can be written using dependent sums, quantification over variables and equalities between smaller neutral or normal forms. Using the closure properties of levelwise decidable propositions, we can prove that these propositions are levelwise decidable.

- An alternative is to present NfTy, Ne and Nf using an indexed W-type, i.e. as the least fixed point of an indexed container.

  Then one can make use of general theorems to characterize equalities in the total spaces of indexed W-types. We prove a suitable theorem in theorem A.2.12. □

Corollary 6.4.4. *The sets $\mathcal{S}_F.\mathsf{Ty}_n(1)$ and $\mathcal{S}_F.\mathsf{Tm}(1, A)$ have levelwise decidable equality.*

Proof. The sets

$$(A : \mathcal{S}_F.\mathsf{Ty}_n(1)) \times \mathsf{NfTy}_n(A),$$
$$(A : \mathcal{S}_F.\mathsf{Ty}_n(1)) \times (a : \mathcal{S}_F.\mathsf{Tm}(1, A)) \times \mathsf{Nf}_A(a)$$

have levelwise decidable equality.

By normalization, $\mathsf{NfTy}_n(A)$ and $\mathsf{Nf}_A(a)$ are contractible. Thus the sets

$$\mathcal{S}_F.\mathsf{Ty}_n(1),$$
$$(A : \mathcal{S}_F.\mathsf{Ty}_n(1)) \times \mathcal{S}_F.\mathsf{Tm}(1, A)$$

have levelwise decidable equality, as needed. □

Finally, we can obtain decidability of equality externally for the syntax $\mathcal{S}$.

Theorem 6.4.5. *The sets of types and terms of the initial algebra of MLTT have decidable equality.*

Proof. We prove decidability of equality for types, the case of terms is analogous. Take a context $\Gamma \in \mathcal{S}$. Since $F : \mathbf{Ren}(\mathcal{S}) \to \mathcal{S}$ is surjective on objects, we have a renaming context $\Gamma_0 \in \mathbf{Ren}(\mathcal{S})$ such that $F(\Gamma_0) = \Gamma$.

By lemma 6.3.3, the set $\mathcal{S}.\mathsf{Ty}(\Gamma)$ is isomorphic to the set $\mathcal{S}[\Gamma].\mathsf{Ty}(1)$, which is equal to $\mathcal{S}[F(\Gamma_0)].\mathsf{Ty}(1)$. But, by corollary 6.4.4, the presheaf $\mathcal{S}_F.\mathsf{Ty}(1)$ over $\mathbf{Ren}(\mathcal{S})$ has levelwise decidable equality. This implies that its evaluation at Γ_0 has decidable equality, i.e. $\mathcal{S}[F(\Gamma_0)].\mathsf{Ty}(1)$ has decidable equality. □

6.5 Application: normalization for extensions of MLTT

6.5.1 Extension: Strict algebras

We show normalization for an extension of MLTT with a type of lists satisfying the monoid laws strictly. The methods we use should also allow for extensions of MLTT with types of strict algebras for other generalized algebraic theories. Decidability of equality should however be provable exactly when the GAT has decidable equality, meaning that the components of its classifying Σ-CwF have decidable equality (which is the case for the algebraic theory of monoids). There is a caveat: when considering algebraic theories with non-linear equations, such as groups (with the non-linear equation $(x - x) = 0$), normal forms are no longer stable under all renamings, and normalization needs to be interleaved with the decidability of equality. The case of non-linear equations will be discussed in more details in section 6.5.3.

The signature for this extension consists of the following operations and equations:

$$
\begin{aligned}
&\mathsf{List} : \mathsf{Ty}_n \to \mathsf{Ty}_n,\\
&\iota : A \to \mathsf{List}(A),\\
&\mathsf{nil} : \mathsf{List}(A),\\
&(- +\!\!+ -) : \mathsf{List}(A) \to \mathsf{List}(A) \to \mathsf{List}(A),\\
&\mathsf{nil} +\!\!+ x = x,\\
&x +\!\!+ \mathsf{nil} = x,\\
&(x +\!\!+ y) +\!\!+ z = x +\!\!+ (y +\!\!+ z),\\
&\mathsf{elimList} : (P : \mathsf{List}(A) \to \mathsf{Ty}_m)(n : P(\mathsf{nil}))(c : \forall x\ l \to P(l) \to P(\iota(x) +\!\!+ l)) \to\\
&\quad \forall l \to P(l),\\
&\mathsf{elimList}(P, n, c, \mathsf{nil}) = n,\\
&\mathsf{elimList}(P, n, c, \iota(x) +\!\!+ y) = c(x, \mathsf{elimList}(P, n, c, y)).
\end{aligned}
$$

We have to identify what are the normal forms for this type of lists, and how they interact with neutral forms. The observation is that, analogously to how normal forms for elements of a boolean type are elements of the free bipointed set over the set of neutral boolean terms, the normal forms for these lists should be elements of the free monoid over the set of neutral list terms.

For any type $A : \mathcal{S}_F.\mathsf{Ty}(1)$, the set $\mathcal{S}_F.\mathsf{Tm}(1, \mathsf{List}_A)$ is equipped with a monoid structure, with the operations nil and $(- +\!\!+ -)$. We define a family

$$\mathsf{NfList}_A : \mathcal{S}_F.\mathsf{Tm}(1, \mathsf{List}_A) \to \mathsf{Set}$$

of **normal lists** as the underlying family of the free displayed monoid over $\mathcal{S}_F.\mathsf{Tm}(1, \mathsf{List}(A))$ equipped with maps

$$\mathsf{ne}^{\mathsf{nflist}} : \mathsf{Ne}_{\mathsf{List}(A)}(l) \to \mathsf{NfList}_A(l),$$
$$\iota^{\mathsf{nflist}} : \mathsf{Nf}_A(x) \to \mathsf{NfList}_A(\iota(x)).$$

This means that an element of $\mathsf{NfList}_A(l)$ is a list whose elements are either neutral elements of List_A or normal elements of A, and such that the term l is the concatenation of their underlying terms.

We then define neutral and normal forms by extending the definition from section 6.4.1 with three new constructors:

$$\mathsf{List}^{\mathsf{nfty}} : \mathsf{NfTy}(A) \to \mathsf{NfTy}(\mathsf{List}(A)),$$
$$\mathsf{elimList}^{\mathsf{ne}} : \mathsf{NfTy}(A) \to \mathsf{Nf}(b) \to \mathsf{Nf}(c)$$
$$\to \mathsf{Ne}_{\mathsf{List}(A)}(l) \to \mathsf{NfList}_A(w) \to \mathsf{Ne}(\mathsf{elimList}(P, b, c, l +\!\!+ w)),$$
$$\mathsf{nflist}^{\mathsf{nf}} : \mathsf{NfList}_A(l) \to \mathsf{Nf}_{\mathsf{List}(A)}(l)$$

For any displayed type $A^\bullet$ in the normalization higher-order model, we define the family $\mathsf{List}^\bullet(A^\bullet).\mathsf{pred} : \mathcal{S}_F.\mathsf{Tm}(1, \mathsf{List}(A)) \to \mathsf{Set}$ as the underlying family of the free displayed monoid over $\mathcal{S}_F.\mathsf{Tm}(1, \mathsf{List}(A))$ equipped with maps

$$\mathsf{reflect}_{\mathsf{List}} : \mathsf{Ne}_{\mathsf{List}(A)}(l) \to \mathsf{List}^\bullet(A^\bullet).\mathsf{pred}(l),$$
$$\iota^\bullet : A^\bullet.\mathsf{pred}(x) \to \mathsf{List}^\bullet(A^\bullet).\mathsf{pred}(\iota(x)).$$

This means that an element of $\mathsf{List}^\bullet(A^\bullet).\mathsf{pred}(l)$ is a list whose elements are either neutral elements of List_A or terms satisfying the predicate $A^\bullet.\mathsf{pred}$, and such that the term l is the concatenation of their underlying terms.

The map $\mathsf{List}^\bullet(A^\bullet).\mathsf{reify} : \mathsf{List}^\bullet(A^\bullet).\mathsf{pred}(l) \to \mathsf{Nf}_{\mathsf{List}(A)}(l)$ is defined as a composition

$$\mathsf{List}^\bullet(A^\bullet).\mathsf{pred}(l) \xrightarrow{\mathsf{reifyList}} \mathsf{NfList}_A(l) \xrightarrow{\mathsf{nflist}^{\mathsf{nf}}} \mathsf{Nf}_{\mathsf{List}(A)}(l),$$

where the first function is defined using the universal property of $\mathsf{List}^\bullet(A^\bullet).\mathsf{pred}(l)$ and $A^\bullet.\mathsf{reify}$, as the unique displayed monoid morphism such that

$$\mathsf{reifyList}(\mathsf{reflect}_{\mathsf{List}}(l^{\mathsf{ne}})) = \mathsf{ne}^{\mathsf{nflist}}(l^{\mathsf{ne}}),$$
$$\mathsf{reifyList}(\iota^\bullet(x^\bullet)) = \iota^{\mathsf{nflist}}(A^\bullet.\mathsf{reify}(x^\bullet)).$$

The normal type $\mathsf{List}^\bullet(A^\bullet).\mathsf{nfty}$ is $\mathsf{List}^{\mathsf{nfty}}(A^\bullet.\mathsf{nfty})$.

The displayed operations $\mathsf{nil}^\bullet$ and $+\!\!+^\bullet$ are the operations of the displayed monoid $\mathsf{List}^\bullet(A^\bullet).\mathsf{pred}$. The displayed strict associativity and identity laws follow from the fact that the displayed monoid $\mathsf{List}^\bullet(A^\bullet).\mathsf{pred}$ satisfies these laws.

It remains to define the displayed eliminator. We know that $\mathsf{List}^\bullet(A^\bullet).\mathsf{pred}(-)$ is a free displayed monoid equipped with maps $\mathsf{reflect}_{\mathsf{List}}$ and $\iota^\bullet$. Because free monoids are just lists, we know that $\mathsf{List}^\bullet(A^\bullet).\mathsf{pred}(-)$ is also free among displayed families over $\mathcal{S}_F.\mathsf{Tm}(1, \mathsf{List}(A))$ equipped with operations

$$\mathsf{nil}^\bullet : \mathsf{List}^\bullet(A^\bullet).\mathsf{pred}(\mathsf{nil}),$$
$$(\mathsf{reflect}_{\mathsf{List}}(-) +\!\!+ -)$$
$$: (l^{\mathsf{ne}} : \mathsf{Ne}_{\mathsf{List}(A)}(l)) \to \mathsf{List}^\bullet(A^\bullet).\mathsf{pred}(w) \to \mathsf{List}^\bullet(A^\bullet).\mathsf{pred}(l +\!\!+ w),$$
$$(\iota^\bullet(-) +\!\!+ -)$$
$$: (x^\bullet : A^\bullet.\mathsf{pred}(x)) \to \mathsf{List}^\bullet(A^\bullet).\mathsf{pred}(w) \to \mathsf{List}^\bullet(A^\bullet).\mathsf{pred}(\iota(x) +\!\!+ w).$$

We can therefore define:

$$\begin{aligned}
&\mathsf{elimList}^\bullet(P^\bullet, n^\bullet, c^\bullet, \mathsf{nil}^\bullet) = n^\bullet,\\
&\mathsf{elimList}^\bullet(P^\bullet, n^\bullet, c^\bullet, \mathsf{reflect}_{\mathsf{List}}(l^{\mathsf{ne}}) +\!\!+ w^\bullet)\\
&\quad = P^\bullet(\dots).\mathsf{reflect}(\mathsf{elimList}^{\mathsf{ne}}(\dots, l^{\mathsf{ne}}, \mathsf{reifyList}(w^\bullet)))\\
&\mathsf{elimList}^\bullet(P^\bullet, n^\bullet, c^\bullet, \iota^\bullet(x^\bullet) +\!\!+ w^\bullet)\\
&\quad = c^\bullet(x^\bullet, \mathsf{elimList}^\bullet(P^\bullet, n^\bullet, c^\bullet, w^\bullet)).
\end{aligned}$$

Remark 6.5.1. The eliminator elimList is not an eliminator for free monoids, but rather for right-nested lists.

We could include distinct eliminators for both right-nested and left-nested lists, and the normalization proof would still go through. ⌟

Remark 6.5.2. The proof of decidability of equality is trickier than in the case of MLTT, because normal forms are now defined as a quotient inductive-inductive type. In this case, we can choose to define NfList using right-nested lists so as to avoid quotienting; decidability of equality can then be proven in the same way as for MLTT.

For more complicated algebraic theories, such as rings, quotienting cannot be avoided in this way. One has to make use of the fact that the free algebras functors are finitary and commute with sequential colimits. The normal forms can then be expressed as a sequential colimit of normal forms with bounded height. See theorem A.2.13 for the details. ⌟

6.5.2 Extension: Strict functoriality

We extend the theory further by the following operations and equations:

$$\begin{aligned}
&\mathsf{map} : (A \to B) \to \mathsf{List}(A) \to \mathsf{List}(B),\\
&\mathsf{map}(f, \iota(x)) = \iota(f(x)),\\
&\mathsf{map}(f, \mathsf{nil}) = \mathsf{nil},\\
&\mathsf{map}(f, x +\!\!+ y) = \mathsf{map}(f, x) +\!\!+ \mathsf{map}(f, y),\\
&\mathsf{map}(\mathsf{id}, l) = l,\\
&\mathsf{map}(f \circ g, l) = \mathsf{map}(f, \mathsf{map}(g, l)),
\end{aligned}$$

where $(A \to B)$ is a binder rather than a function type.

Strict functoriality was also covered by the methods of Allais, McBride, and Boutillier (2013), and was later studied from another point of view by Laurent, Lennon-Bertrand, and Maillard (2024). The approach presented here follows the ideas of the latter paper. I have chosen to include strict functoriality to show that it is mostly orthogonal to strict algebras.

When proving normalization for strict monoids, we had to change the notion of normal list. In order to prove normalization for this extension, we have to change the notion of neutral list: for any neutral list l, the list $\mathsf{map}(f, l)$ is also considered neutral. In order to deal with the functoriality laws, we need a representation of neutral lists that includes this equation.

The family $\mathsf{CNeList}_A : \mathcal{S}_F.\mathsf{Tm}(1, \mathsf{List}(A)) \to \mathsf{Set}$ of **compacted neutral list** is inductively generated (mutually with neutral and normal forms) by the following constructor:

$$\begin{aligned}
&\mathsf{map}^{\mathsf{cne}} : \mathsf{NfTy}(B) \to \mathsf{NfTy}(A) \to (\forall b^{\mathsf{var}} \to \mathsf{Nf}_A(f[\mathsf{var}(b^{\mathsf{var}})])) \to\\
&\quad \mathsf{Ne}_{\mathsf{List}(B)}(l) \to \mathsf{CNeList}_A(\mathsf{map}(f, l)).
\end{aligned}$$

Compacted neutral lists are used instead of neutral lists in the constructors for normal forms that previously referred to neutral lists:

$$\begin{aligned}&\mathsf{cne}^{\mathsf{nflist}} : \mathsf{CNeList}_A(l) \to \mathsf{NfList}_A(l),\\&\mathsf{elimList}^{\mathsf{ne}} : \mathsf{NfTy}(A) \to \mathsf{Nf}(b) \to \mathsf{Nf}(c)\\&\quad \to \mathsf{CNeList}_A(l) \to \mathsf{NfList}_A(w) \to \mathsf{Ne}(\mathsf{elimList}(P, b, c, l +\!\!+ w)).\end{aligned}$$

The constructor $\mathsf{reflect}_{\mathsf{List}}$ is changed as follows:

$$\begin{aligned}&\mathsf{reflect}_{\mathsf{List}} : \mathsf{NfTy}(B) \to (f^\bullet : \forall b^{\mathsf{var}} \to A^\bullet(\mathsf{app}(f, \mathsf{var}(b^{\mathsf{var}})))) \to\\&\quad \mathsf{Ne}_{\mathsf{List}(B)}(l) \to \mathsf{List}^\bullet(A^\bullet).\mathsf{pred}(\mathsf{map}(f, l)).\end{aligned}$$

Its arguments can be seen as a semantic counterpart of compacted neutral list.

The reflection function for lists is then defined as

$$\mathsf{List}^\bullet(A^\bullet).\mathsf{reflect}(l^{\mathsf{ne}}) = \mathsf{reflect}_{\mathsf{List}}(A^\bullet.\mathsf{nfty}, (\lambda a^{\mathsf{var}} \mapsto A^\bullet.\mathsf{reflect}(\mathsf{var}^{\mathsf{ne}}(a^{\mathsf{var}}))), l^{\mathsf{ne}}),$$

where we rely on the equation $\mathsf{map}(\mathsf{id}, l) = l$. The rest of the normalization model is unchanged, and it only remains to define the displayed map operation.

The operation $\mathsf{map}^\bullet(f^\bullet) : \mathsf{List}^\bullet(A^\bullet).\mathsf{pred}(l) \to \mathsf{List}^\bullet(B^\bullet).\mathsf{pred}(\mathsf{map}(f, l))$ is defined using the universal property of the free displayed monoid $\mathsf{List}^\bullet(A^\bullet).\mathsf{pred}(-)$. Thus it suffices to define its action on $\iota^\bullet$ and $\mathsf{reflect}_{\mathsf{List}}$.

$$\begin{aligned}&\mathsf{map}^\bullet(f^\bullet, \iota^\bullet(a^\bullet))\\&\quad = \iota^\bullet(f^\bullet(a^\bullet)),\\&\mathsf{map}^\bullet(f^\bullet, \mathsf{reflect}_{\mathsf{List}}(X^{\mathsf{nfty}}, g^\bullet, l^{\mathsf{ne}}))\\&\quad = \mathsf{reflect}_{\mathsf{List}}(X^{\mathsf{nfty}}, (\lambda x^{\mathsf{var}} \to f^\bullet(g^\bullet(x^{\mathsf{var}}))), l^{\mathsf{ne}}).\end{aligned}$$

In order to check the functoriality equations, it suffices to look at the actions on $\iota^\bullet$ and $\mathsf{reflect}_{\mathsf{List}}$, which indeed satisfy the functoriality equations.

6.5.3 Discussion: normalization in presence of non-linear equations

Most of the ideas we have used to prove normalization for MLTT with strict monoids can also be applied to other algebraic structures. However, there is one crucial part where we have relied on the specificities of monoids: when doing normalization for the eliminator, we have gone through the equivalence between free monoids and right-nested lists.

We indeed run into some trouble when trying to deal with eliminators for algebraic structures with non-linear equations. For example, we may want to extend the type theory with types $\mathbb{Z}[A]$ of polynomial rings, which have the structure of a ring with definitional equations. The algebraic theory of rings includes the non-linear equation

$$(x - x) = 0.$$

By itself, this non-linear equation is not an issue, but it becomes problematic when considered in conjunction with an operation $\mathsf{is\text{-}zero} : \mathbb{Z}[A] \to \mathsf{Bool}$ with the computation rule $\mathsf{is\text{-}zero}(0) = \mathsf{true}$.

The problem with non-linear equations is that normalization now needs to be interleaved with equality checking: in order to normalize $\mathsf{is\text{-}zero}(u - v)$, one needs to decide the equality of u and v.

- Normal forms are no longer stable under renamings. Indeed, for x, y different variables, $\mathsf{is\text{-}zero}(x - y)$ is in normal form, and is moreover neutral. But, after renaming $[x \mapsto y]$, it reduces to $\mathsf{is\text{-}zero}(x - x)$, which is no longer neutral and may unblock some other computation.

- At first glance, it seems possible to proceed with a different notion of normal form. A very similar issue arises in normalization for cubical type theories: the normal forms are not stable under general substitution of interval variables, but the semantics of cubical type theory require access to arbitrary substitutions of interval variables. The solution of Sterling and Angiuli (2021) involves changing the notion of normal form, or more specifically the notion of neutral form, by annotating neutrals with a *frontier of instability*, indicating that a neutral may stop being neutral in a "future world" (in the sense of Kripke semantics, i.e. after application of a substitution of interval variables).

  This approximately means that normal forms become trees that include all possible "future" normal forms, quotiented in such a way that redundancy is avoided in the representation.

  The same approach can be adopted to define normal forms in our case: $\mathsf{is\text{-}zero}(x - y)$ should be a neutral with frontier of instability $(x = y)$, meaning that it stops being neutral when $(x = y)$.

- There are still issues when trying to proceed with the normalization proof. The core of the problem lies in the difference between normal forms and semantic values.

  As mentioned $\mathsf{is\text{-}zero}(x - y)$ should stop being neutral when $(x = y)$. This means that we need to explain how to resume computation if we learn that $(x = y)$. Here the equality $(x = y)$ is really an equality between normal forms, i.e. an equality in the total space of Nf. But in the normalization model, we are really computing with semantic values. Even when $(x = y)$, we may have different semantic values $(x^\bullet \neq y^\bullet)$ in the computation of $\mathsf{is\text{-}zero}^\bullet(x^\bullet - y^\bullet)$. This is even more problematic for a more dependent version of $\mathsf{is\text{-}zero}$, such as

  $$\mathsf{is\text{-}zero} : (x : \mathbb{Z}[A]) \to \mathsf{Maybe}(x = 0).$$

  One possible way to proceed could be to make sure that $(x^\bullet = y^\bullet)$ can be deduced from $(x = y)$, e.g. by restricting the logical predicates to propositions, or less restrictively by ensuring that the reification functions are injective. Unfortunately, this doesn't seem possible: for some types, notably Π-types, there is no choice in the definition of the logical predicates, and there exists base models (consider the terminal model) for which the reification functions are not injective (even though we only care about the initial model in the end, initiality is only used after constructing the normalization model; the construction of the normalization model works for an arbitrary base model).

  Another option is to reuse frontiers of instability and consider "exceptional neutrals"

  $$\mathsf{assert}(\varphi),$$

  which are neutral away from some proposition φ, at an arbitrary type and over an arbitrary term. We can then block the computation of $\mathsf{is\text{-}zero}^\bullet(x^\bullet - y^\bullet)$ until

$x^\bullet = y^\bullet$ holds: its value would essentially be

$$(\mathsf{assert}(x^\bullet = y^\bullet); \mathsf{true}^\bullet),$$

where we know that the proposition $x^\bullet = y^\bullet$ holds in $\mathsf{true}^\bullet$.

Using a subsequent induction on the syntax, we would then prove that the exceptional neutrals never occur in normal forms. That approach may work, but seems to require propositional resizing, as propositions φ of arbitrary size can be used in the exceptional neutrals.

6.6 Application: conservativity of two-level type theory

The two-level type theory of Annenkov, Capriotti, Kraus, and Sattler (2023) is a type theory with two levels, called inner and outer levels. Inner types can be seen as outer types, but not the other way around. The two levels are allowed to have different identity types, with different strictness. Typically, the inner level is HoTT and the outer level models MLTT with UIP,

One of the motivation for two-level type theory is to support metaprogramming or staging when the inner language is not strict enough or not rich enough. In the context of HoTT, this is often used to describe constructions that are possible for any externally fixed natural number n, but not by internal quantification on $n : \mathsf{Nat}$.

We present two-level type theory as a SOGAT $\mathcal{T}_{\mathsf{2ltt}}$, parametrized by SOGATs $\mathcal{T}_{\mathsf{inner}}$ and $\mathcal{T}_{\mathsf{outer}}$ for the inner and outer theories.

Definition 6.6.1. Let $\mathcal{T}_{\mathsf{inner}}$ and $\mathcal{T}_{\mathsf{outer}}$ be two SOGATs, along with a chosen representable sort $(A : \mathsf{ty} \vdash \mathsf{tm}(A)\ \mathsf{type}_{\mathsf{rep}}) \in \mathcal{T}_{\mathsf{outer}}$. We define a new SOGAT $\mathcal{T}_{\mathsf{2ltt}}$, presented by:

- A SOGAT morphism $I : \mathcal{T}_{\mathsf{inner}} \to \mathcal{T}_{\mathsf{2ltt}}$.
- A SOGAT morphism $O : \mathcal{T}_{\mathsf{outer}} \to \mathcal{T}_{\mathsf{2ltt}}$.
- For every generating sort $(\gamma : \Gamma \vdash A(\gamma)\ \mathsf{type}) \in \mathcal{T}_{\mathsf{inner}}$, we have a new type

  $$(\gamma : I(\Gamma) \vdash \ulcorner A \urcorner(\gamma) : \mathsf{ty}) \in \mathcal{T}_{\mathsf{2ltt}},$$

  along with an isomorphism

  $$(\gamma : I(\Gamma) \vdash \mathsf{tm}(\ulcorner A \urcorner(\gamma)) \cong I(A)(\gamma)) \in \mathcal{T}_{\mathsf{2ltt}}.$$ ⌟

Thus two-level type theory embeds both the inner and the outer theories, with additional rules expressing that every inner sort corresponds to terms of some outer type.

One of the main classes of models of two-level type theory are presheaf models:

- Simplicial and cubical presheaf models of HoTT also model two-level type theory. The outer layer is interpreted as the presheaf model of extensional type theory, and the inner layer corresponds to its restriction to fibrant types.
- If $\mathcal{C}$ is any model of HoTT, then $\mathbf{Psh}(\mathcal{C})$ is a model of a two-level type theory, with HoTT as the inner layer and extensional type theory as the outer layer. More generally, if $\mathcal{C}$ is any $\mathcal{T}_{\mathsf{inner}}$-algebra, then $\mathbf{Psh}(\mathcal{C})$ is a $\mathcal{T}_{\mathsf{2ltt}}$-algebra with extensional type theory as the outer layer

The Yoneda embedding $よ : \mathcal{C} \to \mathbf{Psh}(\mathcal{C})$ then extends to a morphism of $\mathcal{T}_{\mathsf{inner}}$-algebras that is moreover a contextual isomorphism (bijective on every sort). This is essentially due to the Yoneda lemma.

It is known that two-level type theory with a suitable outer layer is conservative over its inner layer. We show how to prove this result from an application of a relative induction principle over $(\Sigma, \Pi_{\mathsf{rep}})$-CwFs. We can view this result as a generalization of canonicity, in which the canonical forms for terms of inner types are terms of the original model. In fact we can recover the canonicity of MLTT from the conservativity result for a two-level type theory with only booleans in the inner layer.

Our goal is to prove that the actions of I on terms are bijective.

The proof of this theorem will involve induction on both $\mathcal{T}_{\mathsf{inner}}$ and $\mathcal{T}_{\mathsf{2ltt}}$. Write $R : \mathbf{Ren}(\mathcal{T}_{\mathsf{inner}}) \to \mathcal{T}_{\mathsf{inner}}$ for the CwF of renamings of the $(\Sigma, \Pi_{\mathsf{rep}})$-CwF $\mathcal{T}_{\mathsf{inner}}$, and work internally to $\mathbf{Psh}(\mathbf{Ren}(\mathcal{T}_{\mathsf{inner}}))$. Then by theorem 6.3.11, $(\mathcal{T}_{\mathsf{2ltt}})_{IR}$ and $(\mathcal{T}_{\mathsf{inner}})_R$ satisfy the following induction principles:

- For every displayed $(\Sigma, \Pi_{\mathsf{rep}})$-CwF $\mathcal{T}^\bullet$ over $(\mathcal{T}_{\mathsf{2ltt}})_{IR}$, dependent $(\Sigma, \Pi_{\mathsf{rep}})$-CwF morphism $i^\bullet : \Delta\mathcal{T}_{\mathsf{2ltt}} \to \mathcal{T}^\bullet[i]$ (over $i : \Delta\mathcal{T}_{\mathsf{2ltt}} \to (\mathcal{T}_{\mathsf{2ltt}})_{IR}$) and family

  $$\mathsf{var}^\bullet : (A : \mathcal{T}_R.\mathsf{Ty}(1))(A^\bullet : \mathcal{T}^\bullet.\mathsf{Ty}(1)[I(A)])(a : \mathsf{Var}(A)) \to \mathcal{T}^\bullet.\mathsf{Tm}(1, A^\bullet)[I(\mathsf{var}(a))],$$

  there is a section $[\![-]\!]$ of $\mathcal{T}^\bullet$ such that $[\![i(-)]\!] = i^\bullet(-)$ and $[\![\mathsf{var}_A(a)]\!] = \mathsf{var}^\bullet([\![I(A)]\!], a)$.

- For every displayed $(\Sigma, \Pi_{\mathsf{rep}})$-CwF $\mathcal{T}^\S$ over $(\mathcal{T}_{\mathsf{inner}})_R$, dependent $(\Sigma, \Pi_{\mathsf{rep}})$-CwF morphism $i^\S : \Delta\mathcal{T}_{\mathsf{inner}} \to \mathcal{T}^\S[i]$ (over $i : \Delta\mathcal{T}_{\mathsf{inner}} \to (\mathcal{T}_{\mathsf{inner}})_R$) and family

  $$\mathsf{var}^\S : (A : (\mathcal{T}_{\mathsf{inner}})_R.\mathsf{Ty}(1))(A^\S : \mathcal{T}^\S.\mathsf{Ty}(1)[A])(a : \mathsf{Var}(A)) \to \mathcal{T}^\S.\mathsf{Tm}(1, A^\S)[\mathsf{var}(a)],$$

  there is a section $[\![-]\!]$ of $\mathcal{T}^\S$ such that $[\![i(-)]\!] = i^\S(-)$ and $[\![\mathsf{var}_A(a)]\!] = \mathsf{var}^\S([\![A]\!], a)$.

The displayed higher-order model over $(\mathcal{T}_{\mathsf{2ltt}})_{IR}$

We first construct a displayed $(\Sigma, \Pi_{\mathsf{rep}})$-family $\mathbb{T}^\bullet$ over $(\mathcal{T}_{\mathsf{2ltt}})_{IR}$. The components of $\mathbb{T}^\bullet$ are distinguished by a superscript $-^\bullet$.

- A displayed type over $A : (\mathcal{T}_{\mathsf{2ltt}})_{IR}.\mathsf{Ty}(1)$ is a unary logical predicate

  $$A^\bullet.\mathsf{pred} : (\mathcal{T}_{\mathsf{2ltt}})_{IR}.\mathsf{Tm}(1, A) \to \mathsf{Set}.$$

  The displayed representable types are defined in the same way.

- A displayed term of type $A^\bullet$ over $a : (\mathcal{T}_{\mathsf{2ltt}})_{IR}.\mathsf{Tm}(1, A)$ is an element

  $$a^\bullet : A^\bullet.\mathsf{pred}(a).$$

- The logical predicates for the $\mathbf{1}$-, Σ- and Π- types are defined in the same way as previously encountered; up to isomorphism, there is no choice in their definitions.

We can construct the displayed contextualization $\mathsf{Cxl}(\mathbb{T}^\bullet)$, which is a displayed contextual $(\Sigma, \Pi_{\mathsf{rep}})$-CwF over $(\mathcal{T}_{\mathsf{2ltt}})_{IR}$. However we are not yet able to apply the relative induction principle over $(\mathcal{T}_{\mathsf{2ltt}})_{IR}$ and obtain a section of $\mathsf{Cxl}(\mathbb{T}^\bullet)$; we are missing an interpretation $i^\bullet$ of $\Delta\mathcal{T}_{\mathsf{2ltt}}$ and an interpretation $\mathsf{var}^\bullet$ of variables.

We will need to use the universal property of $(\mathcal{T}_{\mathsf{inner}})_R$ to construct the interpretation of variables. The displayed contextualization $\mathsf{Cxl}(\mathbb{T}^\bullet)$ can be restricted to a displayed $(\Sigma, \Pi_{\mathsf{rep}})$-CwF $\mathsf{Cxl}(\mathbb{T}^\bullet)[I_R]$ over $\mathcal{T}$.

$$\begin{array}{ccc} \mathsf{Cxl}(\mathbb{T}^\bullet)[I_R] & \longrightarrow & \mathsf{Cxl}(\mathbb{T}^\bullet) \\ \downarrow & \lrcorner & \downarrow \\ (\mathcal{T}_{\mathsf{inner}})_R & \xrightarrow{I_R} & (\mathcal{T}_{\mathsf{2ltt}})_{IR} \end{array}$$

The displayed higher-order model over $(\mathcal{T}_{\mathsf{inner}})_R$

We define a displayed $(\Sigma, \Pi_{\mathsf{rep}})$-family $\mathbb{T}^{\S}$ over the total $(\Sigma, \Pi_{\mathsf{rep}})$-CwF of $\mathsf{Cxl}(\mathbb{T}^\bullet)[I_R]$:

- A displayed type over $A : \mathcal{T}_{\mathsf{inner}}.\mathsf{Ty}(1)$ and $A^\bullet : \mathcal{T}_{\mathsf{2ltt}}.\mathsf{Tm}(1, I_R(A)) \to \mathsf{Set}$ consists, for every $a : (\mathcal{T}_{\mathsf{2ltt}})_{IR}.\mathsf{Tm}(1, I_R(A))$, of an isomorphism

  $$A^{\S}.\mathsf{reify} : A^\bullet(a) \cong \{a_0 : (\mathcal{T}_{\mathsf{inner}})_R.\mathsf{Tm}(1, A) \mid I_R(a_0) = a\} : A^{\S}.\mathsf{reflect}.$$

  Note that this family of isomorphisms induces a total isomorphism

  $$(\mathcal{T}_{\mathsf{inner}})_R.\mathsf{Tm}(1, A) \cong (a : (\mathcal{T}_{\mathsf{2ltt}})_{IR}.\mathsf{Tm}(1, I_R(A))) \times A^\bullet(a),$$

  that extends the action

  $$(\mathcal{T}_{\mathsf{inner}})_R.\mathsf{Tm}(1, A) \to (\mathcal{T}_{\mathsf{2ltt}})_{IR}.\mathsf{Tm}(1, I_R(A)).$$

  of I_R on closed terms.

- A displayed term of type $A^{\S}$ over $a : \mathcal{T}_{\mathsf{inner}}.\mathsf{Tm}(1, A)$ and $a^\bullet : A^\bullet(I_R(a))$ is a proof of the equality

  $$a^\bullet = A^{\S}.\mathsf{reflect}(a).$$

- The reify and reflect functions for $\mathbf{1}$-types are trivial.

- In the case of Σ- and Π-types, we have $(A, A^\bullet, A^{\S})$ and $(B, B^\bullet, B^{\S})$, and we want to construct an isomorphisms

  $$\Sigma^{\S}.\mathsf{reify}(p) : \Sigma^\bullet(A^\bullet, B^\bullet)(p) \cong \{p_0 : (\mathcal{T}_{\mathsf{inner}})_R.\mathsf{Tm}(1, \Sigma(A, B)) \mid I_R(p_0) = p\},$$
  $$\Pi^{\S}.\mathsf{reify}(f) : \Pi^\bullet(A^\bullet, B^\bullet)(f) \cong \{f_0 : (\mathcal{T}_{\mathsf{inner}})_R.\mathsf{Tm}(1, \Pi(A, B)) \mid I_R(f_0) = f\},$$

  where

  $$\begin{aligned} B^\bullet &: \forall(a : (\mathcal{T}_{\mathsf{2ltt}})_{IR}.\mathsf{Tm}(1, I_R(A)))\ (a^\bullet : A^\bullet(a)) \\ &\quad \to \mathsf{Ty}^\bullet(B[a]), \\ B^{\S} &: \forall(a : (\mathcal{T}_{\mathsf{inner}})_R.\mathsf{Tm}(1, A))\ (a^\bullet : A^\bullet(I_R(a)))\ (a^{\S} : a^\bullet = A^{\S}.\mathsf{reify}(a)) \\ &\quad \to \mathsf{Ty}^{\S}(B[a], B^\bullet(I_R(a), a^\bullet)). \end{aligned}$$

  Note that we can simplify $B^{\S}$ to

  $$B^{\S} : \forall(a : (\mathcal{T}_{\mathsf{inner}})_R.\mathsf{Tm}(1, A)) \to \mathsf{Ty}^{\S}(B[a], B^\bullet(A^{\S}.\mathsf{reify}(a))).$$

The definition of $\Sigma^{\S}$ is straightforward:

$$\Sigma^{\S}(a,b).\mathsf{reify}(a^{\bullet},b^{\bullet}) \triangleq (A^{\S}.\mathsf{reify}(a^{\bullet}), B^{\S}(a).\mathsf{reify}(b^{\bullet})),$$
$$\Sigma^{\S}(a,b).\mathsf{reflect}(a_0,b_0) \triangleq (A^{\S}.\mathsf{reflect}(a_0), B^{\S}(a).\mathsf{reflect}(b_0)).$$

The definition of $\Pi^{\S}$ is more complicated; it is essentially the following:

$$\begin{aligned}
&\Pi^{\S}(f).\mathsf{reify}(f^{\bullet}) \\
&\quad \triangleq \mathsf{lam}(\lambda a^{\mathsf{var}} \mapsto B^{\S}(\mathsf{var}(a^{\mathsf{var}})).\mathsf{reify}(f^{\bullet}(A^{\S}.\mathsf{reflect}(\mathsf{var}(a^{\mathsf{var}}))))), \\
&\Pi^{\S}(f).\mathsf{reflect}(f_0) \\
&\quad \triangleq (\lambda a\ a^{\bullet} \mapsto B^{\S}(A^{\S}.\mathsf{reify}(a^{\bullet})).\mathsf{reflect}(\mathsf{app}(f_0, A^{\S}.\mathsf{reify}(a^{\bullet})))),
\end{aligned}$$

but we need to justify the quantification over variables in the λ-abstraction $\mathsf{lam}(\lambda a^{\mathsf{var}} \mapsto \dots)$ appearing in the definition of the reification. Here lam is the λ-abstraction of the model $(\mathcal{T}_{\mathsf{inner}})_R$; it should take a term in the context $(1.A) \in (\mathcal{T}_{\mathsf{inner}})_R$. This quantification over variables is justified by the isomorphism

$$((\mathcal{T}_{\mathsf{inner}})_R \mathbin{/\!\!/} (1.A)) \cong \prod_{\mathsf{Var}(A)} (\mathcal{T}_{\mathsf{inner}})_R$$

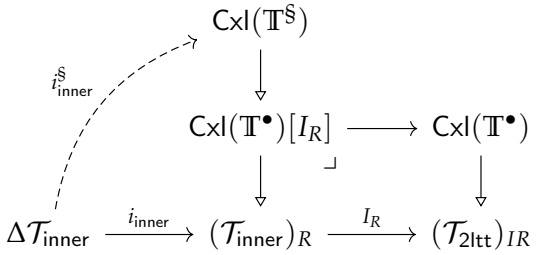
from proposition 6.3.5.

We also need to verify that $\Pi^{\S}(f).\mathsf{reify}$ and $\Pi^{\S}(f).\mathsf{reflect}$ are inverses. We need to prove the β- and $\eta-$ rules transposed through the above isomorphism.

$$\begin{aligned}
&\Pi^{\S}(f).\mathsf{reify}(\Pi^{\S}(f).\mathsf{reflect}(f_0)) \\
&\quad = \mathsf{lam}(\lambda a^{\mathsf{var}} \mapsto B^{\S}(\mathsf{var}(a^{\mathsf{var}})).\mathsf{reify}(B^{\S}(\dots).\mathsf{reflect}(\mathsf{app}(f_0, A^{\S}.\mathsf{reify}(A^{\S}.\mathsf{reflect}(\mathsf{var}(a^{\mathsf{var}}))))))) \\
&\qquad \text{(Unfolding the definitions)} \\
&\quad = \mathsf{lam}(\lambda a^{\mathsf{var}} \mapsto B^{\S}(\mathsf{var}(a^{\mathsf{var}})).\mathsf{reify}(B^{\S}(\mathsf{var}(a^{\mathsf{var}})).\mathsf{reflect}(\mathsf{app}(f_0, \mathsf{var}(a^{\mathsf{var}}))))) \\
&\qquad \text{(Simplifying } A^{\S}\text{)} \\
&\quad = \mathsf{lam}(\lambda a^{\mathsf{var}} \mapsto \mathsf{app}(f_0, \mathsf{var}(a^{\mathsf{var}}))) && \text{(Simplifying } B^{\S}\text{)} \\
&\quad = f, && \text{(By the } \eta\text{-rule)} \\
&\Pi^{\S}(f).\mathsf{reflect}(\Pi^{\S}(f).\mathsf{reify}(f^{\bullet})) \\
&\quad = \lambda a\ a^{\bullet} \mapsto B^{\S}(A^{\S}.\mathsf{reify}(a^{\bullet})).\mathsf{reflect}(\mathsf{app}(f_0, A^{\S}.\mathsf{reify}(a^{\bullet}))) \\
&\quad \text{where } f_0 = \mathsf{lam}(\lambda a^{\mathsf{var}} \mapsto B^{\S}(\mathsf{var}(a^{\mathsf{var}})).\mathsf{reify}(f^{\bullet}(A^{\S}.\mathsf{reflect}(\mathsf{var}(a^{\mathsf{var}}))))) \\
&\qquad \text{(Unfolding the definitions)} \\
&\quad = \lambda a\ a^{\bullet} \mapsto B^{\S}(A^{\S}.\mathsf{reify}(a^{\bullet})).\mathsf{reflect}(B^{\S}(A^{\S}.\mathsf{reify}(a^{\bullet}))).\mathsf{reify}(f^{\bullet}(A^{\S}.\mathsf{reflect}(A^{\S}.\mathsf{reify}(a^{\bullet})))) \\
&\qquad \text{(By the } \beta\text{-rule)} \\
&\quad = \lambda a\ a^{\bullet} \mapsto B^{\S}(A^{\S}.\mathsf{reify}(a^{\bullet})).\mathsf{reflect}(B^{\S}(A^{\S}.\mathsf{reify}(a^{\bullet}))).\mathsf{reify}(f^{\bullet}(a^{\bullet})) \\
&\qquad \text{(Simplifying } A^{\S}\text{)} \\
&\quad = \lambda a\ a^{\bullet} \mapsto f^{\bullet}(a^{\bullet}) && \text{(Simplifying } B^{\S}\text{)} \\
&\quad = f^{\bullet}.
\end{aligned}$$

Applying the relative induction principle over $(\mathcal{T}_{\mathsf{inner}})_R$

This finishes the definition of the displayed $(\Sigma, \Pi_{\mathsf{rep}})$-family $\mathbb{T}^{\S}$. We can now construct its contextualization $\mathsf{Cxl}(\mathbb{T}^{\S})$, which is displayed over $\mathsf{Cxl}(\mathbb{T}^{\bullet})[I_R]$.



Our next goal is to construct a dependent morphism $i^{\S}_{\mathsf{inner}} : \Delta\mathcal{T}_{\mathsf{inner}} \to \mathsf{Cxl}(\mathbb{T}^{\S})[i_{\mathsf{inner}}]$, as shown in the above diagram.

For this purpose, since $\Delta\mathcal{T}_{\mathsf{inner}}$ is type-presented, it suffices to prove that the projection $\mathsf{Cxl}(\mathbb{T}^{\S}) \to (\mathcal{T}_{\mathsf{inner}})_R$ is surjective on types and bijective on terms.

A type of $\mathsf{Cxl}(\mathbb{T}^{\S})$ over a context $(\Gamma, \Gamma^{\bullet}, \Gamma^{\S})$ is a triple $(A, A^{\bullet}, A^{\S})$, where

$$
\begin{aligned}
&A : (\mathcal{T}_{\mathsf{inner}})_R.\mathsf{Ty}(\Gamma),\\
&A^{\bullet} : \forall(\gamma : (\mathcal{T}_{\mathsf{2ltt}})_{IR}(1, I_R(\Gamma)))\ (\gamma^{\bullet} : \Gamma^{\bullet}(\gamma)) \to (\mathcal{T}_{\mathsf{2ltt}})_{IR}.\mathsf{Tm}(1, I_R(A)) \to \mathsf{Set},\\
&A^{\S} : \forall\gamma_0\ (\gamma^{\bullet} : \Gamma^{\bullet}(I_R(\gamma_0)))\ (\gamma^{\S} : \Gamma^{\S}(\gamma_0, \gamma^{\bullet})) \to \forall a \to A^{\bullet}(a) \cong \{a_0 \mid I_R(a_0) = a\}.
\end{aligned}
$$

For any base type A, we can define the other components

$$
\begin{aligned}
A^{\bullet}(\gamma, \gamma^{\bullet}, a) &\triangleq \{a_0 \mid I_R(a_0) = a\},\\
A^{\S}(\gamma, \gamma^{\bullet}, \gamma^{\S}, a) &= \mathsf{id},
\end{aligned}
$$

witnessing that the projection $\mathsf{Cxl}(\mathbb{T}^{\S}) \to (\mathcal{T}_{\mathsf{inner}})_R$ is surjective on types. A term of $\mathsf{Cxl}(\mathbb{T}^{\S})$ of type $(A, A^{\bullet}, A^{\S})$ is a triple $(a, a^{\bullet}, a^{\S})$, where

$$
\begin{aligned}
&a : (\mathcal{T}_{\mathsf{inner}})_R.\mathsf{Tm}(\Gamma, A),\\
&a^{\bullet} : \forall(\gamma : (\mathcal{T}_{\mathsf{2ltt}})_{IR}(1, I_R(\Gamma)))\ (\gamma^{\bullet} : \Gamma^{\bullet}(\gamma)) \to A^{\bullet}(\gamma, \gamma^{\bullet}, I_R(a)[\gamma]),\\
&a^{\S} : \forall\gamma_0\ (\gamma^{\bullet} : \Gamma^{\bullet}(I_R(\gamma_0)))\ (\gamma^{\S} : \Gamma^{\S}(\gamma_0, \gamma^{\bullet})) \to a^{\bullet}(I_R(\gamma_0), \gamma^{\bullet}) = A^{\S}(\gamma_0, \gamma^{\bullet}, \gamma^{\S}).\mathsf{reify}(a[\gamma_0]).
\end{aligned}
$$

Because $(\mathcal{T}_{\mathsf{inner}})_R$ is contextual, and using the telescopic contextualization for $\mathsf{Cxl}(\mathbb{T}^{\S})$, we can see the context $(\Gamma, \Gamma^{\bullet}, \Gamma^{\S})$ as a closed type. As a consequence $\Gamma^{\S}$ is a family of equivalences

$$\{\gamma_0 \mid I_R(\gamma_0) = \gamma\} \cong \Gamma^{\bullet}(\gamma),$$

and $\Gamma^{\S}(\gamma_0, \gamma^{\bullet})$ above means that γ_0 is related to $\gamma^{\bullet}$.

We then observe that $(\gamma^{\bullet} : \Gamma^{\bullet}(I_R(\gamma_0))) \times (\gamma^{\S} : \Gamma^{\S}(\gamma_0, \gamma^{\bullet}))$ is contractible, and that $(\gamma : (\mathcal{T}_{\mathsf{2ltt}})_{IR}(1, I_R(\Gamma))) \times (\gamma^{\bullet} : \Gamma^{\bullet}(\gamma))$ is equivalent to $(\gamma_0 : (\mathcal{T}_{\mathsf{inner}})_R(1, \Gamma))$. Thus both $a^{\bullet}$ and $a^{\S}$ above can be brought to depend on the same parameters $(\gamma_0, \gamma^{\bullet}, \gamma^{\S})$. For any such $(\gamma_0, \gamma^{\bullet}, \gamma^{\S})$, the set

$$(a^{\bullet} : A^{\bullet}(I_R(\gamma_0), \gamma^{\bullet}, I_R(a[\gamma_0]))) \times (a^{\bullet} = A^{\S}(\gamma_0, \gamma^{\bullet}, \gamma^{\S}).\mathsf{reify}(a[\gamma_0]))$$

is contractible. The components $a^{\bullet}$ and $a^{\S}$ are therefore uniquely determined, i.e. the projection $\mathsf{Cxl}(\mathbb{T}^{\S}) \to (\mathcal{T}_{\mathsf{inner}})_R$ is bijective on terms.

We thus obtain the desired section $i^{\S}_{\mathsf{inner}} : \Delta\mathcal{T}_{\mathsf{inner}} \to \mathsf{Cxl}(\mathbb{T}^{\S})[i_{\mathsf{inner}}]$ from $\Delta\mathcal{T}_{\mathsf{inner}}$ being type-presented.

We can then define an operation $\mathsf{var}^{\S}$.

$$\mathsf{var}^{\S} : \forall A\ ((A^{\bullet}, A^{\S}) : \mathsf{Cxl}(\mathbb{T}^{\S}).\mathsf{Ty}(1)[A])\ (a : \mathsf{Var}(A)) \to A^{\bullet}(I_R(\mathsf{var}(a))),$$
$$\mathsf{var}^{\S}((A^{\bullet}, A^{\S}), a) = A^{\S}.\mathsf{reflect}(\mathsf{var}(a)).$$

By the universal property of $(\mathcal{T}_{\mathsf{inner}})_R$, we obtain a section $[\![-]\!]$ of $\mathsf{Cxl}(\mathbb{T}^{\S})$ such that $[\![i(-)]\!] = i^{\S}(-)$ and $[\![\mathsf{var}_A(a)]\!] = \mathsf{var}^{\S}([\![A]\!], a)$.

Applying the relative induction principle over $(\mathcal{T}_{\mathsf{2ltt}})_{IR}$

We now want to use the universal property of $(\mathcal{T}_{\mathsf{2ltt}})_{IR}$ and obtain a section of $\mathsf{Cxl}(\mathbb{T}^{\bullet})$. Our next goal is to construct the dependent morphism $i^{\bullet} : \Delta\mathcal{T}_{\mathsf{2ltt}} \to \mathsf{Cxl}(\mathbb{T}^{\bullet})[i]$. Using the universal property of $\Delta\mathcal{T}_{\mathsf{2ltt}}$, it suffices to define

$$i^{\bullet}_{\Delta I} : \Delta\mathcal{T}_{\mathsf{inner}} \to \mathsf{Cxl}(\mathbb{T}^{\bullet})[i \circ \Delta I],$$

$$i^{\bullet}_{\Delta O} : \Delta\mathcal{T}_{\mathsf{outer}} \to \mathsf{Cxl}(\mathbb{T}^{\bullet})[i \circ \Delta O],$$

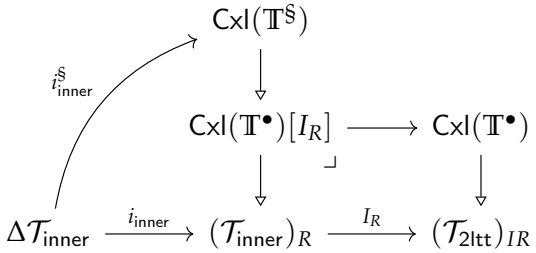
and additional data corresponding to the types

$$(\gamma : \Gamma \vdash \ulcorner A \urcorner(\gamma) : \mathsf{ty}) \in \mathcal{T}_{\mathsf{2ltt}}.$$

and isomorphisms

$$(\gamma : \Gamma \vdash \mathsf{tm}(\ulcorner A \urcorner(\gamma)) \cong A(\gamma)) \in \mathcal{T}_{\mathsf{2ltt}}.$$

The morphism $i^{\bullet}_{\Delta I}$ is defined as the composition of the three top maps in the above diagram:



The morphism $i^{\bullet}_{\Delta O}$ is defined using the universal property of the outer SOGAT $\mathcal{T}_{\mathsf{outer}}$, which we assumed to be $\mathcal{T}_{\mathsf{MLTT}}$. If we unfold the definitions, we observe that we have to define exactly a higher-order displayed model of $\mathcal{T}_{\mathsf{MLTT}}$ over (the underlying model of $\mathcal{T}_{\mathsf{MLTT}}$ of) $(\mathcal{T}_{\mathsf{2ltt}})_{IR}$. We define it exactly like the canonicity model from section 6.2.

Finally, we need to define the interpretation of the types $\ulcorner - \urcorner$ and of the associated isomorphisms. Take a generating sort $(\gamma : \Gamma \vdash A(\gamma)\ \mathsf{type})$ of $\mathcal{T}_{\mathsf{inner}}$. We need to define a displayed term $\ulcorner A \urcorner^{\bullet}$ of type $i^{\bullet}_{\Delta O}(\mathsf{ty})$ in $\mathsf{Cxl}(\mathbb{T}^{\bullet})$ that lies over $i_{\mathsf{inner}}\ulcorner A \urcorner$, along with a displayed type isomorphism $\mathsf{tm}^{\bullet}(\ulcorner A \urcorner^{\bullet}) \cong i^{\bullet}_{\Delta I}(A)$ over $i^{\bullet}_{\Delta I}(\Gamma)$.

$$\ulcorner A \urcorner^{\bullet}(a) \triangleq \{a_0 : (\mathcal{T}_{\mathsf{inner}})_R.\mathsf{Tm}(1, i_{\mathsf{inner}}(A)) \mid I_R(a_0) = a\}.$$

This data determines $i^{\bullet} : \Delta\mathcal{T}_{\mathsf{2ltt}} \to \mathsf{Cxl}(\mathbb{T}^{\bullet})[i]$.

The final step should be to define an operation $\mathsf{var}^{\bullet}$.

$$\mathsf{var}^{\bullet} : \forall A\ (A^{\bullet} : \mathsf{Cxl}(\mathbb{T}^{\bullet}).\mathsf{Ty}(1)[I_R(A)])\ (a : \mathsf{Var}(A)) \to A^{\bullet}(I_R(\mathsf{var}(a))).$$

We essentially want to pose $\mathsf{var}^\bullet(A, A^\bullet, a) = [\![A]\!].\mathsf{reflect}(\mathsf{var}(a))$. This is however not possible because we do not know whether $A^\bullet = [\![A]\!]$. In principle this shouldn't be an issue, because after obtaining the section $(\!|-|\!)$ of $\mathsf{Cxl}(\mathbb{T}^\bullet)$, we would only care about $\mathsf{var}^\bullet(A, (\!|I_R(A)|\!), a)$ and we would know that $(\!|I_R(A)|\!) = [\![A]\!]$.

To make the proof work, we have to mutually perform the construction of $\mathsf{var}^\bullet$, the induction on $(\mathcal{T}_{\mathsf{2ltt}})_{IR}$ and the proof of $(\!|I_R(-)|\!) = [\![-]\!]$ (which is another induction on $(\mathcal{T}_{\mathsf{inner}})_R$. For this purpose, we use the characterization of $(\mathcal{T}_{\mathsf{2ltt}})_{IR}$ as a colimit from theorem 6.3.12, so as to essentially only define $\mathsf{var}^\bullet$ for types that we control.

The details below are quite complex; the idea is to alternate between defining $\mathsf{var}^\bullet$ for some variables, and proving $(\!|I_R(-)|\!) = [\![-]\!]$ for the types that can be built from the variables that have already been seen. This is repeated until $\mathsf{var}^\bullet$ is defined on all variables.

We have, by theorem 6.3.12, the following characterization

$$(\mathcal{T}_{\mathsf{inner}})_R \cong \operatorname*{colim}_{n<\omega} J^n_{\mathsf{inner}}(\Delta\mathcal{T}_{\mathsf{inner}}),$$

where J_{inner} is the endofunctor of displayed algebras with partial variables from theorem 6.3.12.

We write $J^n_{\mathsf{inner}}(\Delta\mathcal{T}_{\mathsf{inner}}) = (\Delta\mathcal{T}_{\mathsf{inner}})[\mathsf{var}(a) \mid a \in \mathsf{Var}_n(A)]$; where $\mathsf{Var}_n(A)$ are the variable that are defined in $J^n_{\mathsf{inner}}(\Delta\mathcal{T}_{\mathsf{inner}})$.

We have the same characterization for $(\mathcal{T}_{\mathsf{2ltt}})_{IR}$, but we can also choose to add variables to $(\Delta\mathcal{T}_{\mathsf{2ltt}})$ at the same speed as for $(\Delta\mathcal{T}_{\mathsf{inner}})$:

$$(\mathcal{T}_{\mathsf{2ltt}})_{IR} \cong \operatorname*{colim}_{n<\omega}(\Delta\mathcal{T}_{\mathsf{2ltt}})[\mathsf{var}(a) \mid a \in \mathsf{Var}_n(A)].$$

The displayed algebras with partial variables $(\Delta\mathcal{T}_{\mathsf{2ltt}})[\mathsf{var}(a) \mid a \in \mathsf{Var}_n(A)]$ actually need to be defined by induction on n, together with maps

$$\alpha_n : (\Delta\mathcal{T}_{\mathsf{inner}})[\mathsf{var}(a) \mid a \in \mathsf{Var}_n(A)] \to (\Delta\mathcal{T}_{\mathsf{2ltt}})[\mathsf{var}(a) \mid a \in \mathsf{Var}_n(A)].$$

We then obtain a morphism of sequential diagrams, inducing the map $I_R : (\mathcal{T}_{\mathsf{inner}})_R \to (\mathcal{T}_{\mathsf{2ltt}})_{IR}$ between the colimits.

For $n \leq m \leq \omega$, we write $j^m_{\mathsf{inner},n}$ and $j^m_{\mathsf{2ltt},n}$ for the finite or transfinite compositions of maps in the two sequential diagrams.

Now, by induction on n, we construct a section $(\!|-|\!)_n$ of the displayed model $\mathsf{Cxl}(\mathcal{T}^\bullet)[j^\omega_{\mathsf{2ltt},n}]$ over

$$(\Delta\mathcal{T}_{\mathsf{2ltt}})[\mathsf{var}(a) \mid a \in \mathsf{Var}_n(A)]$$

such that

$$(\!|\alpha_n(-)|\!)_n = [\![j^\omega_{\mathsf{inner},n}(-)]\!],$$

and for every $a \in \mathsf{Var}_n(A)$,

$$(\!|\mathsf{var}(a)|\!)_n = [\![j^\omega_{\mathsf{inner},n}(A)]\!].\mathsf{reflect}(\mathsf{var}(a))$$

(which is well-typed thanks to the previous equality).

In the base case, we use the section $i^\bullet : \Delta\mathcal{T}_{\mathsf{2ltt}} \to \mathsf{Cxl}(\mathbb{T}^\bullet)[i]$. In the recursive case, we construct $(\!|-|\!)_{n+1}$ by extending $(\!|-|\!)_n$. We have to define the action of $(\!|-|\!)_{n+1}$ on the variables of $\mathsf{Var}_{n+1}(A)$. The variables that are added at stage $n+1$ have types that exist at stage n, so we can define $(\!|\mathsf{var}(a)|\!)_{n+1} \triangleq [\![j^\omega_{\mathsf{inner},n}(A)]\!].\mathsf{reflect}(\mathsf{var}(a))$; this relies on the induction hypothesis $(\!|\alpha_n(-)|\!)_n = [\![j^\omega_{\mathsf{inner},n}(-)]\!]$. By the second induction hypothesis, this coincides with the actions of $(\!|-|\!)_n$ on variables that were already defined at stage n.

We also have to prove $(\!|\alpha_{n+1}(-)|\!)_{n+1} = [\![j^{\omega}_{\mathsf{inner},n+1}(-)]\!]$; this follows from the universal property of $(\Delta\mathcal{T}_{\mathsf{inner}})[\mathsf{var}(a) \mid a \in \mathsf{Var}_{n+1}(A)]$ as an extension of $(\Delta\mathcal{T}_{\mathsf{inner}})[\mathsf{var}(a) \mid a \in \mathsf{Var}_n(A)]$.

By lifting the sections to the colimit, we obtain a section $(\!|-|\!)$ of $\mathsf{Cxl}(\mathcal{T}^{\bullet})$. Take a closed type $A : (\mathcal{T}_{\mathsf{inner}})_R.\mathsf{Ty}(1)$ and a term $a : (\mathcal{T}_{2\mathsf{ltt}})_{IR}.\mathsf{Tm}(1, I_R(A))$. We have $(\!|a|\!) : (\!|I_R(A)|\!).\mathsf{pred}(a)$. Moreover, $(\!|I_R(A)|\!) = [\![A]\!]$, implying that

$$[\![A]\!].\mathsf{reify} : (\!|I_R(A)|\!).\mathsf{pred}(a) \cong \{a_0 : (\mathcal{T}_{\mathsf{inner}})_R.\mathsf{Tm}(1, A) \mid I_R(a_0) = a.\} : [\![A]\!].\mathsf{reflect}$$

Thus $a_0 = [\![A]\!].\mathsf{reify}((\!|a|\!))$ is a preimage of a. Finally, given any other a_1 such that $I_R(a_1) = a$, we have $[\![a_1]\!] : (\!|a|\!) = [\![A]\!].\mathsf{reflect}(a_1)$. Thus $a_1 = [\![A]\!].\mathsf{reify}((\!|a|\!)) = a_0$.

Therefore the actions of I_R on closed terms are bijective.

Theorem 6.6.2. *If $\mathcal{T}_{2\mathsf{ltt}}$ is a two level type theory for an inner theory $\mathcal{T}_{\mathsf{inner}}$ and an outer theory $\mathcal{T}_{\mathsf{outer}} = \mathcal{T}_{\mathsf{MLTT}}$, the $(\Sigma, \Pi_{\mathsf{rep}})$-CwF morphism $I : \mathcal{T}_{\mathsf{inner}} \to \mathcal{T}_{2\mathsf{ltt}}$ is bijective on terms.*

Proof. The result we have just proven internally to $\mathbf{Psh}(\mathbf{Ren}(\mathcal{T}_{\mathsf{inner}}))$ implies that for any $\Gamma : \mathbf{Ren}(\mathcal{T}_{\mathsf{inner}})$, the actions of I on terms in context $R(\Gamma)$ are bijective. But $R : \mathbf{Ren}(\mathcal{T}_{\mathsf{inner}}) \to \mathcal{T}_{\mathsf{inner}}$ is essentially surjective on objects, so the actions of I on terms are bijective over arbitrary contexts. □

Appendix A

Equalities in the total spaces of indexed W-types

The goal of this appendix is to give a characterization of equality in the total space

$$\Sigma_{i:I} W_C(i)$$

of the indexed W-type $W_C : I \to \mathsf{Set}$ associated to an indexed container C.

By characterizing equalities in the total space $\Sigma_{i:I} W_C(i)$, we mean proving that for any $x, y : \Sigma_{i:I} W_C(i)$ belongs to some class of propositions, such as decidable propositions, levelwise decidable propositions, or perhaps just homotopy propositions in a homotopical setting, etc. We will require that this class of propositions is closed under some operations; one can think of it as a dominance (Rosolini 1986).

Note that it is alternatively possible to characterize the equality of $W_C(i)$ itself, as another indexed W-type indexed by $(x, y : I) \times (x = y)$, or to prove that the equality of $W_C(i)$ itself belongs to some class of propositions. Such results have been proven by Hugunin (2017), or by de Jong (2021, Theorem 66).

This is however not directly useful for us: in our applications to normalization, the indexed W-type is the predicate of being a normal form, which is known to be contractible as a result of the normalization proof. Thus we already know that its equality is trivial, and we are actually interested in the equality of I.

Generalizing indexed W-types, we may want to characterize the equality in the total space

$$\Sigma_{i:I} F^\omega$$

of the initial algebra F^ω of an endofunctor $F : \mathbf{Set}^I \to \mathbf{Set}^I$.

We first cover the case of a general endofunctor, and then show how it specializes to indexed containers.

A.1 Endofunctors

Definition A.1.1. A **good class of propositions** L is a family

$$L : \mathsf{Prop} \to \mathsf{Prop},$$

i.e. a subset $L \subseteq \mathsf{Prop}$, closed under:

- Finite conjunctions and disjunctions;

- Dependent sums;
- Sequential colimits. ⌟

Examples include all propositions, decidable propositions, and levelwise decidable propositions in the internal language of presheaves (by lemma 2.3.24).

Definition A.1.2. A set A has L**-equality** if for every $x, y : A$, the proposition $(x = y)$ is in L. ⌟

Lemma A.1.3. *Consider a sequence*

$$X_0 \xrightarrow{\iota_0^1} X_1 \to \ldots \to X_n \xrightarrow{\iota_n^{n+1}} X_{n+1} \to \ldots$$

If the maps ι_n^{n+1} are all injective and the sets X_n have L-equality, then the colimit $\operatorname{colim}_{n<\omega} X_n$ has L-equality.

Proof. We write $\iota_n^m : X_n \to X_m$ for the finite compositions of the maps (when $n \leq m$), and $\iota_n^\omega : X_n \to \operatorname{colim}_{n<\omega} X_n$ for the transfinite composition into the sequential colimit.

Take two elements in the colimit. Because we are proving a proposition, we can assume that they can be written $\iota_n^\omega(x)$ and $\iota_m^\omega(y)$. By moving to $\max(n, m)$, we can assume without loss of generality that $n = m$.

Now equality in a sequential colimit is the sequential colimit of equality:

$$(\iota_n^\omega(x) = \iota_n^\omega(y)) \cong \operatorname*{colim}_{k<\omega}(\iota_n^{n+k}(x) = \iota_n^{n+k}(y)).$$

But the maps $(\iota_n^{n+k}(x) = \iota_n^{n+k}(y)) \to (\iota_n^{n+k+1}(x) = \iota_n^{n+k+1}(y))$ are equivalences because the map ι_{n+k}^{n+k+1} is injective. So the second colimit is just the proposition $(x = y)$. This is an equality in X_{n+k}, hence this proposition is in L.

Thus $(\iota_n^\omega(x) = \iota_n^\omega(y))$ is also in L, as needed. □

Definition A.1.4. A functor $F : \mathbf{Set}^I \to \mathbf{Set}^O$ is ω**-finitary** if it preserves sequential colimits. ⌟

Definition A.1.5. A functor $F : \mathbf{Set}^I \to \mathbf{Set}^O$ **preserves** L**-equality** if for every family $X : I \to \mathsf{Set}$, if $\Sigma_{i:I} X(i)$ has L-equality, then $\Sigma_{o:O} F(X)(o)$ has L-equality. ⌟

Theorem A.1.6. *Let L be a good class of propositions and F be an endofunctor $F : \mathbf{Set}^I \to \mathbf{Set}^I$. If F is ω-finitary, preserves L-equality and preserves monomorphisms, then the total space $\Sigma_{i:I} F^\omega(i)$ has L-equality.*

Proof. By Adámek's fixed point theorem, since F is ω-finitary, the initial algebra can be written as a colimit

$$F^\omega(i) \cong \operatorname*{colim}_{n<\omega} F^n(i),$$

where $F^0(i) = \emptyset$ and $F^{n+1}(i) = F(F^n)(i)$.

We also have

$$\Sigma_{i:I} F^\omega(i) \cong \operatorname*{colim}_{n<\omega} \Sigma_{i:I} F^n(i),$$

since $\Sigma_{i:I}$ commutes with colimits.

We can prove by induction on n that $\Sigma_{i:I} F^n(i)$ has L-equality. The base case is trivial, since $\emptyset$ has L-equality. In the recursive case, we use the fact that F preserves L-equality as an I-indexed container.

Also by induction on n, we can prove that $F^n(i) \to F^{n+1}(i)$ is a monomorphism. The base case is trivial, since any map from $\emptyset$ is a monomorphism. In the recursive case, we use the fact that F preserves monomorphisms.

The maps $\Sigma_{i:I} F^n(i) \to \Sigma_{i:I} F^{n+1}(i)$ are then injective. Thus, by lemma A.1.3, $\Sigma_{i:I} F^\omega(i)$ has L-equality, as needed. □

A.2 Indexed containers

We recall the definition of indexed containers by Altenkirch, Ghani, et al. (2015).

Definition A.2.1. An (I, O)**-indexed container** C consists of the following components:

$$
\begin{aligned}
&C.\mathsf{shape} : \mathsf{Set},\\
&C.\mathsf{position} : C.\mathsf{shape} \to \mathsf{Set},\\
&C.\mathsf{query} : (s : C.\mathsf{shape}) \to C.\mathsf{position}(s) \to I,\\
&C.\mathsf{response} : C.\mathsf{shape} \to O.
\end{aligned}
$$

Its **associated functor** is

$$
\begin{aligned}
&[\![-]\!]_C : \mathbf{Set}^I \to \mathbf{Set}^O,\\
&[\![X]\!]_C(o)\\
&\quad \triangleq (s : C.\mathsf{shape} \mid C.\mathsf{response}(s) = o)\\
&\quad \times ((p : C.\mathsf{position}(s)) \to X(C.\mathsf{query}(p))).
\end{aligned}
$$
⌟

Definition A.2.2. An **indexed** W**-type** is the least fixed point of an (I, I)-indexed container C, i.e. the indexed inductive type with the following presentation:

$$
\begin{aligned}
&W_C : I \to \mathsf{Set},\\
&\mathsf{sup} : (s : C.\mathsf{shape}) \to (f : (p : C.\mathsf{position}(s)) \to W_C(C.\mathsf{query}(p)))\\
&\quad \to W_C(C.\mathsf{response}(s)).
\end{aligned}
$$
⌟

By I-indexed container, we mean an $(I, \top)$-indexed container, i.e. an indexed container without the response component. These are used to describe constructors which do not have a specified output index yet.

Containers are usually constructed from some number of combinators.

- There is an I-indexed container $\top$, with

  $$
  \begin{aligned}
  &\top.\mathsf{shape} = \top,\\
  &\top.\mathsf{position}(-) = \bot.
  \end{aligned}
  $$

  This is used for nullary constructors.

- Given an I-indexed container C and a family of I-indexed containers D indexed by $C.\mathsf{shape}$, there is an I-indexed container $\Sigma(C, D)$, with

  $$
  \begin{aligned}
  &\Sigma(C,D).\mathsf{shape} = (s_c : C.\mathsf{shape}) \times (s_d : D(s_c).\mathsf{shape}),\\
  &\Sigma(C,D).\mathsf{position}(s_c, s_d) = C.\mathsf{position}(s_c) + D(s_c).\mathsf{position}(s_d),\\
  &\Sigma(C,D).\mathsf{query}((s_c, s_d), p_c : C.\mathsf{position}(s_c)) = C.\mathsf{query}(p_c),\\
  &\Sigma(C,D).\mathsf{query}((s_c, s_d), p_d : D(s_c).\mathsf{position}(s_d)) = D(s_c).\mathsf{query}(p_d).
  \end{aligned}
  $$

  This is used for constructors with multiple arguments.

- For any set X, there is an I-indexed container $A(X)$, with X shapes and no positions. This is used for non-recursive arguments to constructors.
- For any map $f : X \to I$, there is an I-indexed container $R_X(f)$, with

$$\begin{aligned} &R_X(f).\mathsf{shape} = X, \\ &R_X(f).\mathsf{position}(x) = \top, \\ &R_X(f).\mathsf{query}(x, -) = f(x). \end{aligned}$$

  This is used for recursive arguments to constructors.
- Given a set X and a family of I-indexed container C indexed by X, there is an I-indexed container C^X, with

$$\begin{aligned} &C^X.\mathsf{shape} = ((x : X) \to C(x).\mathsf{shape}), \\ &C^X.\mathsf{position}(s) = (x : X) \times C(x).\mathsf{position}(s(x)), \\ &C^X.\mathsf{query}(s, (x, p)) = C(x).\mathsf{query}(p). \end{aligned}$$

- Given any I-indexed container C it is possible to attach a response function $\mathsf{response} : C.\mathsf{shape} \to O$ to obtain an (I, O)-indexed container $(C \ \&\ \mathsf{response})$. This is used to specify the output index of a constructor.
- Given two (I, O)-indexed containers C and D, there is a container $(C + D)$, with

$$\begin{aligned} &(C + D).\mathsf{shape} = C.\mathsf{shape} + D.\mathsf{shape}, \\ &(C + D).\mathsf{position} = [C.\mathsf{position}; D.\mathsf{position}], \\ &(C + D).\mathsf{query} = [C.\mathsf{query}; D.\mathsf{query}], \\ &(C + D).\mathsf{response} = [C.\mathsf{response}; D.\mathsf{response}]. \end{aligned}$$

  This is used to combine multiple constructors.

Most containers used in practice will be sums of constructors with an attached response, with each constructor given by an iterated dependent sum of non-recursive and recursive arguments.

For example, consider Fin defined as an inductive family indexed by $\mathbb{N}$.

$$\begin{aligned} &\mathsf{Fin} : \mathbb{N} \to \mathsf{Set}, \\ &\mathsf{fz} : (n : \mathbb{N}) \to \mathsf{Fin}(n + 1), \\ &\mathsf{fs} : (n : \mathbb{N}) \to \mathsf{Fin}(n) \to \mathsf{Fin}(n + 1). \end{aligned}$$

We have corresponding $(\mathbb{N}, \mathbb{N})$-containers for every constructor.

$$\begin{aligned} &C_{\mathsf{fz}} = A(\mathbb{N}) \ \&\ (\lambda n \mapsto n + 1), \\ &C_{\mathsf{fs}} = \Sigma(A(\mathbb{N}), \lambda n \mapsto R_\top(- \mapsto n)) \ \&\ (\lambda(n, -) \mapsto n + 1). \end{aligned}$$

The container for Fin is the sum of the containers for each constructor:

$$C_{\mathsf{Fin}} = C_{\mathsf{fz}} + C_{\mathsf{fs}}.$$

Then the indexed W-type $W_{C_{\mathsf{Fin}}} : \mathbb{N} \to \mathsf{Set}$ is isomorphic to Fin.

Definition A.2.3. An (I, O)-indexed container C is ω-finitary if for every $s : C.\mathsf{shape}$, the set $C.\mathsf{position}(s)$ is ω-compact, meaning that the functor $(C.\mathsf{position}(s) \to -)$ preserves sequential colimits. ⌟

Lemma A.2.4. *If an (I, O)-indexed container C is ω-finitary, then the associated functor $[\![-]\!]_C : \mathsf{Set}^I \to \mathsf{Set}^O$ is ω-finitary, i.e. it preserves sequential colimits.*

Proof. Straightforward. □

Definition A.2.5. An (I, O)-indexed container **preserves L-equality** if its associated functor $[\![-]\!] : \mathbf{Set}^I \to \mathbf{Set}^O$ preserves L-equality. ⌟

We now prove that these two semantic conditions (being ω-finitary and preserving L-equalities) are preserved by the container combinators, with side conditions in some cases.

Lemma A.2.6. *If C preserves L-equality and $D(s)$ preserves L-equality for every s, then $\Sigma(C, D)$ preserves L-equality.*

If C is ω-finitary and $D(s)$ is ω-finitary for every s, then $\Sigma(C, D)$ is ω-finitary.

Proof. We can construct an isomorphism

$$[\![X]\!]_{\Sigma(C,D)} \cong ((s_c, -) : [\![X]\!]_C) \times [\![X]\!]_{D(s_c)}.$$

This implies that $\Sigma(C, D)$ preserves L-equality whenever C and $D(-)$ do.

The indexed container $\Sigma(C, D)$ being ω-finitary follows from ω-compact sets being closed under binary sums. □

Lemma A.2.7. *If C and D preserve L-equality, then $(C + D)$ preserves L-equality.*

If C and D are ω-finitary, then $(C + D)$ is ω-finitary.

Proof. Straightforward. □

Lemma A.2.8. *If the set X has L-equality, then $A(X)$ preserves L-equality.*

For any set X, the indexed container $A(X)$ is ω-finitary.

Proof. Straightforward, note that $[\![F]\!]_{A(X)}$ is constantly X. □

Lemma A.2.9. *If the map $f : X \to I$ is injective, then $R_X(f)$ preserves L-equality.*

For any map $f : X \to I$, the indexed container $R_X(f)$ is ω-finitary.

Proof. It is clear that $R_X(f)$ is ω-finitary, since it has a single position at any shape.

We have $[\![F]\!]_{R_X(f)} \cong ((i : I) \times F(i) \times (x : X \mid f(x) = i))$. If f is injective, then $(x : X \mid f(x) = i)$, which is the fiber of f at i, is a proposition. Therefore its equality is contractible, and is therefore in L. Assuming that $\Sigma_{i:I} F(i)$ has L-equality, then $[\![F]\!]_{R_X(f)}$ has L-equality. Thus $R_X(f)$ preserves L-equality. □

Lemma A.2.10. *If $((x : X) \to -)$ preserves L-equality and C is a family of indexed containers preserving L-equality, then C^X preserves L-equality.*

If X is ω-compact and C is a family of ω-finitary indexed containers, then indexed container C^X is ω-finitary.

Proof. We have an isomorphism $[\![F]\!]_{C^X} \triangleq ((x : X) \to [\![F]\!]_{C(x)})$. If both $((x : X) \to -)$ and $C(x)$ preserve L-equality, then $[\![-]\!]_{C^X}$, which is their composite, also preserves L-equality.

The set of positions of C^X over a shape (x, s) is $(x : X) \times C(x).\mathsf{position}(s(x))$. If both X and $C(x).\mathsf{position}(s(x))$ are ω-compact, then their dependent sum is ω-compact. Thus C^X is ω-finitary when X is ω-compact and $C(x)$ is ω-finitary for any x. □

Lemma A.2.11. *The associated functor $[\![-]\!]_C : \mathbf{Set}^I \to \mathbf{Set}^O$ of any indexed container C preserves monomorphisms.*

Proof. Take a monomorphism $f : X \to Y$ in $\mathbf{Set}^I$ (a levelwise injective natural transformation).

The fiber of $[\![F]\!]_C : [\![X]\!]_C(o) \to [\![Y]\!]_C(o)$ at some $(s, g) : [\![Y]\!]_C(o)$ can be computed to

$$\forall (p : C.\mathsf{position}(s)) \to (x : X(C.\mathsf{query}(p)) \mid f_{C.\mathsf{query}(p)}(x) = g(p)),$$

i.e. to a product of fibers of f_i.

Since the functions f_i are injective, their fibers are propositions. The product of propositions is a proposition, so the fiber of $[\![F]\!]_C$ at (s, g) is a proposition, as needed. □

Theorem A.2.12. *Let L be a good class of propositions and C be an (I, I)-indexed container. If C is ω-finitary and preserves L-equality, then the total space $\Sigma_{i:I} W_C(i)$ has L-equality.*

Proof. This follows from theorem A.1.6, using lemma A.2.4 and lemma A.2.11. □

We can also prove a variant of theorem A.2.12 for nested indexed inductive type. We consider an $(I + T, I)$-indexed container C and a functor $F : \mathbf{Set}^I \to \mathbf{Set}^T$. The set I corresponds to the sorts that are generated by the nested indexed inductive type, and the set T corresponds to the nested sorts. The indexed container C describes the constructors for the non-nested sorts. The functor F describes how the nested sorts are constructed from the non-nested sorts.

Theorem A.2.13. *Let C be an $(I + T, I)$-indexed container and $F : \mathbf{Set}^I \to \mathbf{Set}^T$ be a functor.*

We construct an endofunctor

$$G : \mathbf{Set}^I \to \mathbf{Set}^I,$$
$$G(X) = \lambda i \mapsto [\![[I \ni i \mapsto X(i); T \ni t \mapsto F(X)(t)]]\!]_C(i).$$

If both C and F are ω-finitary and preserve L-equality, and if F preserves monomorphisms, then G is ω-finitary, preserves L-equality and preserves monomorphisms, hence its initial algebra G^ω exists and

$$\Sigma_{i:I} G^\omega(i)$$

has L-equality.

Proof. The functor G is the composition of $[\mathsf{Id}; F] : \mathbf{Set}^I \to \mathbf{Set}^{I+T}$ and $[\![-]\!]_C : \mathbf{Set}^{I+T} \to \mathbf{Set}^I$. Functors that are ω-finitary, preserve L-equality and monomorphisms are closed under copairing and compositions, so the functor G satisfies these semantic conditions, as needed. □

Example A.2.14. As an example, we define a notion of normal form for a type theory extended with a type R with the structure of a ring (with definitional ring laws).

We consider the functor $\mathbb{Z}[-] : \mathbf{Set} \to \mathbf{Ring}$ that constructs a free ring over a set.

As an indexed inductive type, neutral terms and normal forms would be defined as follows (we omit the constructors that would correspond to other type formers):

$$\begin{aligned}
&\mathsf{Ne} : (X : \mathsf{Ty}(1))(x : \mathsf{Tm}(1, X)) \to \mathsf{Set},\\
&\mathsf{var}^{\mathsf{ne}} : (x^{\mathsf{var}} : \mathsf{Var}(X)) \to \mathsf{Ne}(X, \mathsf{var}(x^{\mathsf{var}})),\\
&\mathsf{Nf} : (X : \mathsf{Ty}(1))(a : \mathsf{Tm}(1, X)) \to \mathsf{Set},\\
&\mathsf{ring}^{\mathsf{nf}} : (P : \mathbb{Z}[\Sigma_x \mathsf{Ne}(R, x)]) \to \mathsf{Nf}(R, [\![P]\!]).
\end{aligned}$$

We pose

$$\begin{aligned}
I &= \{\mathsf{ne}(X, x) \mid x : \mathsf{Tm}(1, X)\} + \{\mathsf{nf}(X, x) \mid x : \mathsf{Tm}(1, X)\},\\
T &= \{\mathsf{ring}(P) \mid P : \mathbb{Z}[\mathsf{Tm}(1, R)]\}.
\end{aligned}$$

We define an $(I + T, I)$-indexed container C:

$$\begin{aligned}
C_{\mathsf{var}} &\triangleq A(\mathsf{Var}(X)) \;\&\; \lambda(X, x^{\mathsf{var}}) \mapsto \mathsf{ne}(A, \mathsf{var}(a^{\mathsf{var}})),\\
C_{\mathsf{ring}} &\triangleq R(\lambda P \mapsto \mathsf{ring}(P)) \;\&\; \lambda P \mapsto \mathsf{nf}(R, P),\\
C &\triangleq C_{\mathsf{var}} + C_{\mathsf{ring}}.
\end{aligned}$$

We also define a functor $F : \mathbf{Set}^I \to \mathbf{Set}^T$:

$$F(Y, \mathsf{ring}(P)) = \{Q : \mathbb{Z}[\Sigma_x Y(\mathsf{ne}(R, x))] \mid \mathsf{map}(\mathsf{fst}, Q) = P\}.$$

It should be possible to check that F satisfies the conditions of theorem A.2.13. ⌟

Bibliography

Abel, Andreas (2013). "Normalization by Evaluation: Dependent Types and Impredicativity". Habilitation thesis. Ludwig-Maximilians-Universität München. URL: http://www.cse.chalmers.se/~abela/habil.pdf.

Abel, Andreas, Joakim Öhman, and Andrea Vezzosi (Dec. 2017). "Decidability of conversion for type theory in type theory". In: *Proc. ACM Program. Lang.* 2.POPL. DOI: 10.1145/3158111. URL: https://doi.org/10.1145/3158111.

Adamek, J. and J. Rosicky (1994). *Locally Presentable and Accessible Categories*. London Mathematical Society Lecture Note Series. Cambridge University Press.

Adjedj, Arthur, Meven Lennon-Bertrand, Kenji Maillard, Pierre-Marie Pédrot, and Loïc Pujet (2024). "Martin-Löf à la Coq". In: *Proceedings of the 13th ACM SIGPLAN International Conference on Certified Programs and Proofs, CPP 2024, London, UK, January 15-16, 2024*. Ed. by Amin Timany, Dmitriy Traytel, Brigitte Pientka, and Sandrine Blazy. ACM, pp. 230–245. DOI: 10.1145/3636501.3636951. URL: https://doi.org/10.1145/3636501.3636951.

Agda Developers (2025). *Agda*. Version 2.8.0. URL: https://agda.readthedocs.io/.

Ahrens, Benedikt and Peter LeFanu Lumsdaine (2019). "Displayed Categories". In: *Log. Methods Comput. Sci.* 15.1. DOI: 10.23638/LMCS-15(1:20)2019. URL: https://doi.org/10.23638/LMCS-15(1:20)2019.

Allais, Guillaume, Conor McBride, and Pierre Boutillier (2013). "New equations for neutral terms: a sound and complete decision procedure, formalized". In: *Proceedings of the 2013 ACM SIGPLAN workshop on Dependently-typed programming, DTPICFP 2013, Boston, Massachusetts, USA, September 24, 2013*. Ed. by Stephanie Weirich. ACM, pp. 13–24. DOI: 10.1145/2502409.2502411. URL: https://doi.org/10.1145/2502409.2502411.

Altenkirch, Thorsten, Paolo Capriotti, Thierry Coquand, Nils Anders Danielsson, Simon Huber, and Nicolai Kraus (2017). *Type Theory with Weak J*. 23rd International Conference on Types for Proofs and Programs.

Altenkirch, Thorsten, Neil Ghani, Peter G. Hancock, Conor McBride, and Peter Morris (2015). "Indexed containers". In: *J. Funct. Program.* 25. DOI: 10.1017/S095679681500009X. URL: https://doi.org/10.1017/S095679681500009X.

Altenkirch, Thorsten, Martin Hofmann, and Thomas Streicher (1995). "Categorical reconstruction of a reduction free normalization proof". In: *Category Theory and Computer Science*. Ed. by David Pitt, David E. Rydeheard, and Peter Johnstone. LNCS 953, pp. 182–199.

— (1997). "Reduction-free normalisation for system *F*".

Altenkirch, Thorsten and Ambrus Kaposi (2017). "Normalisation by Evaluation for Type Theory, in Type Theory". In: *Log. Methods Comput. Sci.* 13.4. DOI: 10.23638/LMCS-13(4:1)2017. URL: https://doi.org/10.23638/LMCS-13(4:1)2017.

Annenkov, Danil, Paolo Capriotti, Nicolai Kraus, and Christian Sattler (2023). "Two-level type theory and applications". In: *Math. Struct. Comput. Sci.* 33.8, pp. 688–743. DOI: 10.1017/S0960129523000130. URL: https://doi.org/10.1017/s0960129523000130.

Awodey, Steve (2018). "Natural models of homotopy type theory". In: *Math. Struct. Comput. Sci.* 28.2, pp. 241–286. DOI: 10.1017/S0960129516000268. URL: https://doi.org/10.1017/S0960129516000268.

Awodey, Steve, Nicola Gambino, and Sina Hazratpour (July 2024). "Kripke-Joyal forcing for type theory and uniform fibrations". In: *Selecta Mathematica* 30.4. ISSN: 1420-9020. DOI: 10.1007/s00029-024-00962-2. URL: http://dx.doi.org/10.1007/s00029-024-00962-2.

Bauer, Andrej, Philipp G. Haselwarter, and Peter LeFanu Lumsdaine (2020). *A general definition of dependent type theories*. arXiv: 2009.05539 [math.LO]. URL: https://arxiv.org/abs/2009.05539.

Bednarczyk, Marek, Andrzej Borzyszkowski, Wieslaw Pawlowski, and Michael Barr (Jan. 1999). "Generalized Congruences | Epimorphisms in Cat". In: *Theory and Applications of Categories* 5, pp. 266–280.

Bocquet, Rafaël, Ambrus Kaposi, and Christian Sattler (2023). "For the Metatheory of Type Theory, Internal Sconing Is Enough". In: *8th International Conference on Formal Structures for Computation and Deduction, FSCD 2023, July 3-6, 2023, Rome, Italy*. Ed. by Marco Gaboardi and Femke van Raamsdonk. Vol. 260. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 18:1–18:23. DOI: 10.4230/LIPICS.FSCD.2023.18. URL: https://doi.org/10.4230/LIPIcs.FSCD.2023.18.

Brunerie, Guillaume (2020). *Initiality for Martin-Löf type theory*. https://www.math.uwo.ca/faculty/kapulkin/seminars/hottestfiles/Brunerie-2020-09-10-HoTTEST.pdf.

Capriotti, Paolo (2017). "Models of type theory with strict equality". PhD thesis. University of Nottingham, UK. URL: http://ethos.bl.uk/OrderDetails.do?uin=uk.bl.ethos.719436.

Cartmell, John (Jan. 1986). "Generalised Algebraic Theories and Contextual Categories". In: *Annals of Pure and Applied Logic* 32, pp. 209–243. ISSN: 0168-0072. DOI: 10.1016/0168-0072(86)90053-9. (Visited on 05/29/2024).

Chapman, James (2008). "Type Theory Should Eat Itself". In: *Proceedings of the International Workshop on Logical Frameworks and Metalanguages: Theory and Practice, LFMTPLICS 2008, Pittsburgh, PA, USA, June 23, 2008*. Ed. by Andreas Abel and Christian Urban. Vol. 228. Electronic Notes in Theoretical Computer Science. Elsevier, pp. 21–36. DOI: 10.1016/J.ENTCS.2008.12.114. URL: https://doi.org/10.1016/j.entcs.2008.12.114.

Clairambault, Pierre and Peter Dybjer (2014). "The biequivalence of locally cartesian closed categories and Martin-Löf type theories". In: *Math. Struct. Comput. Sci.* 24.6. DOI: 10.1017/S0960129513000881. URL: https://doi.org/10.1017/S0960129513000881.

Coquand, Thierry (2019). "Canonicity and normalization for dependent type theory". In: *Theor. Comput. Sci.* 777, pp. 184–191. DOI: 10.1016/J.TCS.2019.01.015. URL: https://doi.org/10.1016/j.tcs.2019.01.015.

Corbyn, Nathan, Ohad Kammar, Sam Lindley, Nachiappan Valliappanand, and Jeremy Yallop (2022). "Normalization by Evaluation with Free Extensions". extended abstract.

Curien, Pierre-Louis (1993). "Substitution up to Isomorphism". In: *Fundam. Informaticae* 19.1/2, pp. 51–85.

De Boer, Menno (2020). "A proof and formalization of the initiality conjecture of dependent type theory". Licentiate Thesis. Department of Mathematics, Stockholm University.

de Jong, Tom (2021). "The Scott model of PCF in univalent type theory". In: *Mathematical Structures in Computer Science* 31.10, pp. 1270–1300. DOI: 10.1017/S0960129521000153.

Dybjer, Peter (1995). "Internal Type Theory". In: *Types for Proofs and Programs, International Workshop TYPES'95, Torino, Italy, June 5-8, 1995, Selected Papers*. Ed. by Stefano Berardi and Mario Coppo. Vol. 1158. Lecture Notes in Computer Science. Springer, pp. 120–134. DOI: 10.1007/3-540-61780-9_66. URL: https://doi.org/10.1007/3-540-61780-9%5C_66.

Fiore, Marcelo and Chung-Kil Hur (2010). "Second-order equational logic". In: *Proceedings of the 24th International Conference/19th Annual Conference on Computer Science Logic*. CSL'10/EACSL'10. Brno, Czech Republic: Springer-Verlag, pp. 320–335. ISBN: 364215204X.

Fiore, Marcelo and Ola Mahmoud (2010). "Second-order algebraic theories". In: *Proceedings of the 35th International Conference on Mathematical Foundations of Computer Science*. MFCS'10. Brno, Czech Republic: Springer-Verlag, pp. 368–380. ISBN: 364215154X.

Frey, Jonas (2023). "Duality for Clans: a Refinement of Gabriel-Ulmer Duality". In: *CoRR* abs/2308.11967. DOI: 10.48550/ARXIV.2308.11967. arXiv: 2308.11967. URL: https://doi.org/10.48550/arXiv.2308.11967.

Gabriel, Peter and Friedrich Ulmer (Jan. 1971). *Lokal präsentierbare Kategorien*. Lecture Notes in Mathematics. Springer Berlin, Heidelberg. DOI: https://doi.org/10.1007/BFb0059396.

Garner, Richard (Dec. 2007). "Understanding the Small Object Argument". In: *Applied Categorical Structures* 17. DOI: 10.1007/s10485-008-9137-4.

Gilbert, Gaëtan, Jesper Cockx, Matthieu Sozeau, and Nicolas Tabareau (2019). "Definitional proof-irrelevance without K". In: *Proc. ACM Program. Lang.* 3.POPL, 3:1–3:28. DOI: 10.1145/3290316. URL: https://doi.org/10.1145/3290316.

Gratzer, Daniel, G. A. Kavvos, Andreas Nuyts, and Lars Birkedal (July 2021). "Multimodal Dependent Type Theory". In: *Logical Methods in Computer Science* Volume 17, Issue 3. ISSN: 1860-5974. DOI: 10.46298/lmcs-17(3:11)2021. URL: http://dx.doi.org/10.46298/lmcs-17(3:11)2021.

Gratzer, Daniel and Jonathan Sterling (2020). "Syntactic categories for dependent type theory: sketching and adequacy". In: *CoRR* abs/2012.10783. arXiv: 2012.10783. URL: https://arxiv.org/abs/2012.10783.

Harper, Robert, Furio Honsell, and Gordon Plotkin (Jan. 1993). "A framework for defining logics". In: *J. ACM* 40.1, pp. 143–184. ISSN: 0004-5411. DOI: 10.1145/138027.138060. URL: https://doi.org/10.1145/138027.138060.

Haselwarter, Philipp G. and Andrej Bauer (2023). "Finitary Type Theories With and Without Contexts". In: *J. Autom. Reason.* 67.4, p. 36. DOI: 10.1007/S10817-023-09678-Y. URL: https://doi.org/10.1007/s10817-023-09678-y.

Henry, Simon (2020). *The language of a model category*. www.uwo.ca/math/faculty/kapulkin/seminars/hottestfiles/Henry-2020-01-23-HoTTEST.pdf.

Hofmann, Martin (1994). "On the Interpretation of Type Theory in Locally Cartesian Closed Categories". In: *Computer Science Logic, 8th International Workshop, CSL '94, Kazimierz, Poland, September 25-30, 1994, Selected Papers*. Ed. by Leszek Pacholski and

Jerzy Tiuryn. Vol. 933. Lecture Notes in Computer Science. Springer, pp. 427–441. DOI: 10.1007/BFB0022273. URL: https://doi.org/10.1007/BFb0022273.

Hofmann, Martin (July 1995). "Extensional concepts in intensional type theory". PhD thesis. University of Edinburgh.

— (1999). "Semantical Analysis of Higher-Order Abstract Syntax". In: *14th Annual IEEE Symposium on Logic in Computer Science, Trento, Italy, July 2-5, 1999*. IEEE Computer Society, pp. 204–213. DOI: 10.1109/LICS.1999.782616. URL: https://doi.org/10.1109/LICS.1999.782616.

Hofmann, Martin and Thomas Streicher (1997). *Lifting Grothendieck Universes*.

Hugunin, Jasper (2017). *A Coq development of the theory of Indexed W types with function extensionality*. URL: https://github.com/jashug/IWTypes.

Isaev, Valery (2016). "Algebraic Presentations of Dependent Type Theories". In: *CoRR* abs/1602.08504. arXiv: 1602.08504. URL: http://arxiv.org/abs/1602.08504.

— (2018). "Morita equivalences between algebraic dependent type theories". In: *CoRR* abs/1804.05045. arXiv: 1804.05045. URL: http://arxiv.org/abs/1804.05045.

Johnstone, Peter T (2002). *Sketches of an elephant: a Topos theory compendium*. Oxford logic guides. New York, NY: Oxford Univ. Press. URL: https://cds.cern.ch/record/592033.

Joyal, Andre (2017). *Notes on Clans and Tribes*. arXiv: 1710.10238 [math.CT]. URL: https://arxiv.org/abs/1710.10238.

Kaposi, Ambrus (2017). "Type theory in a type theory with quotient inductive types". PhD thesis. University of Nottingham, UK. URL: https://ethos.bl.uk/OrderDetails.do?uin=uk.bl.ethos.713896.

Kaposi, Ambrus, András Kovács, and Thorsten Altenkirch (Jan. 2019). "Constructing Quotient Inductive-Inductive Types". In: *Proceedings of the ACM on Programming Languages* 3.POPL, pp. 1–24. ISSN: 24751421. DOI: 10.1145/3290315. (Visited on 06/15/2019).

Kaposi, Ambrus and Szumi Xie (2024). "Second-Order Generalised Algebraic Theories: Signatures and First-Order Semantics". In: *9th International Conference on Formal Structures for Computation and Deduction, FSCD 2024, July 10-13, 2024, Tallinn, Estonia*. Ed. by Jakob Rehof. Vol. 299. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 10:1–10:24. DOI: 10.4230/LIPICS.FSCD.2024.10. URL: https://doi.org/10.4230/LIPIcs.FSCD.2024.10.

Kapulkin, Chris and Peter LeFanu Lumsdaine (Sept. 2016). "The homotopy theory of type theories". In: *Advances in Mathematics* 337. DOI: 10.1016/j.aim.2018.08.003.

Kovács, András (2022). "Staged compilation with two-level type theory". In: *Proc. ACM Program. Lang.* 6.ICFP, pp. 540–569. DOI: 10.1145/3547641. URL: https://doi.org/10.1145/3547641.

— (2023). "Type-Theoretic Signatures for Algebraic Theories and Inductive Types". PhD thesis. arXiv: 2302.08837. URL: http://arxiv.org/abs/2302.08837.

Kovács, András and Christian Sattler (2025). *A Generalized Logical Framework*. Work presented at the EuroProofNet WG6 meeting in Genova in April 2025. URL: https://europroofnet.github.io/wg6-genoa/.

Laurent, Théo, Meven Lennon-Bertrand, and Kenji Maillard (2024). "Definitional Functoriality for Dependent (Sub)Types". In: *Programming Languages and Systems - 33rd European Symposium on Programming, ESOP 2024, Held as Part of the European Joint Con-*

ferences on Theory and Practice of Software, ETAPS 2024, Luxembourg City, Luxembourg, April 6-11, 2024, Proceedings, Part I. Ed. by Stephanie Weirich. Vol. 14576. Lecture Notes in Computer Science. Springer, pp. 302–331. DOI: 10.1007/978-3-031-57262-3_13. URL: https://doi.org/10.1007/978-3-031-57262-3%5C_13.

Lawvere, F. William (1963). "Functorial Semantics of Algebraic Theories and Some Algebraic Problems in the context of Functorial Semantics of Algebraic Theories". PhD thesis. Columbia University.

Lumsdaine, Peter LeFanu and Michael A. Warren (2015). "The Local Universes Model: An Overlooked Coherence Construction for Dependent Type Theories". In: *ACM Trans. Comput. Log.* 16.3, 23:1–23:31. DOI: 10.1145/2754931. URL: https://doi.org/10.1145/2754931.

Makkai, Michael (1995). "First Order Logic with Dependent Sorts, with Applications to Category Theory".

Makkai, Michael, Jiří Rosický, and Lukáš Vokřínek (2014). "On a fat small object argument". In: *Advances in Mathematics* 254, pp. 49–68. ISSN: 0001-8708. DOI: https://doi.org/10.1016/j.aim.2013.12.012. URL: https://www.sciencedirect.com/science/article/pii/S0001870813004581.

McBride, Conor and Ross Paterson (2008). "Applicative programming with effects". In: *J. Funct. Program.* 18.1, pp. 1–13. DOI: 10.1017/S0956796807006326. URL: https://doi.org/10.1017/S0956796807006326.

Moeneclaey, Hugo (2022). "Cubical models are cofreely parametric". PhD thesis. Université Paris Cité.

Moura, Leonardo de and Sebastian Ullrich (2021). "The Lean 4 Theorem Prover and Programming Language". In: *Automated Deduction – CADE 28*. Ed. by André Platzer and Geoff Sutcliffe. Cham: Springer International Publishing, pp. 625–635. ISBN: 978-3-030-79876-5. URL: https://link.springer.com/chapter/10.1007/978-3-030-79876-5_37.

Orton, Ian and Andrew M. Pitts (2018). "Axioms for Modelling Cubical Type Theory in a Topos". In: *Log. Methods Comput. Sci.* 14.4. DOI: 10.23638/LMCS-14(4:23)2018. URL: https://doi.org/10.23638/LMCS-14(4:23)2018.

Pfenning, Frank and Carsten Schürmann (1999). "System Description: Twelf - A Meta-Logical Framework for Deductive Systems". In: *Automated Deduction - CADE-16, 16th International Conference on Automated Deduction, Trento, Italy, July 7-10, 1999, Proceedings*. Ed. by Harald Ganzinger. Vol. 1632. Lecture Notes in Computer Science. Springer, pp. 202–206. ISBN: 3-540-66222-7. DOI: 10.1007/3-540-48660-7_14. URL: https://doi.org/10.1007/3-540-48660-7%5C_14.

Pientka, Brigitte (2010). "Beluga: Programming with Dependent Types, Contextual Data, and Contexts". In: *Functional and Logic Programming, 10th International Symposium, FLOPS 2010, Sendai, Japan, April 19-21, 2010. Proceedings*. Ed. by Matthias Blume, Naoki Kobayashi, and Germán Vidal. Vol. 6009. Lecture Notes in Computer Science. Springer, pp. 1–12. ISBN: 978-3-642-12250-7. DOI: 10.1007/978-3-642-12251-4_1. URL: https://doi.org/10.1007/978-3-642-12251-4%5C_1.

Rosolini, Giuseppe (Jan. 1986). "Continuity and effectiveness in topoi". PhD thesis.

Sattler, Christian (2018). "Normalization by evaluation for categories with families".

Shulman, Mike (Aug. 2017). "Brouwer's fixed-point theorem in real-cohesive homotopy type theory". In: *Mathematical Structures in Computer Science* 28.6, pp. 856–941. ISSN: 1469-8072. DOI: 10.1017/s0960129517000147. URL: http://dx.doi.org/10.1017/S0960129517000147.

Shulman, Mike (Sept. 2018). *A Communal Proof of an Initiality Theorem*. URL: https://golem.ph.utexas.edu/category/2018/09/a_communal_proof_of_an_initial.html.

Sterling, Jonathan (Apr. 2022). "First Steps in Synthetic Tait Computability: The Objective Metatheory of Cubical Type Theory". Thesis. Carnegie Mellon University. DOI: 10.1184/R1/19632681.v1. (Visited on 06/04/2024).

Sterling, Jonathan and Carlo Angiuli (2021). "Normalization for Cubical Type Theory". In: *36th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2021, Rome, Italy, June 29 - July 2, 2021*. IEEE, pp. 1–15. DOI: 10.1109/LICS52264.2021.9470719. URL: https://doi.org/10.1109/LICS52264.2021.9470719.

Streicher, Thomas (1991). *Semantics of Type Theory: Correctness, Completeness, and Independence Results*. Birkhäuser Boston, MA.

Swan, Andrew (2018). *Identity Types in Algebraic Model Structures and Cubical Sets*. arXiv: 1808.00915 [math.CT]. URL: https://arxiv.org/abs/1808.00915.

Taylor, Paul (1999). *Practical Foundations Of Mathematics*. ISBN: 0-521-63107-6.

The Rocq Development Team (Apr. 2025). *The Rocq Prover*. Version 9.0. DOI: 10.5281/zenodo.15149629. URL: https://doi.org/10.5281/zenodo.15149629.

Uemura, Taichi (2019). "A General Framework for the Semantics of Type Theory". In: *CoRR* abs/1904.04097. arXiv: 1904.04097. URL: http://arxiv.org/abs/1904.04097.

— (2021). "Abstract and concrete type theories". PhD thesis. Amsterdam: Institute for Logic, Language and Computation. URL: https://hdl.handle.net/11245.1/41ff0b60-64d4-4003-8182-c244a9afab3b.

Univalent Foundations Program, The (2013). *Homotopy Type Theory: Univalent Foundations of Mathematics*. Institute for Advanced Study: https://homotopytypetheory.org/book.

Voevodsky, Vladimir (2014). *B-systems*. arXiv: 1410.5389 [math.LO]. URL: https://arxiv.org/abs/1410.5389.

— (2016). *Subsystems and regular quotients of C-systems*. arXiv: 1406.7413 [math.LO]. URL: https://arxiv.org/abs/1406.7413.