INTERNAL ∞-CATEGORICAL MODELS OF DEPENDENT TYPE THEORY

TOWARDS 2LTT EATING HOTT

NICOLAI KRAUS

Abstract. Using dependent type theory to formalise the syntax of dependent type theory is a very active topic of study and goes under the name of "type theory eating itself" or "type theory in type theory." Most approaches are at least loosely based on Dybjer's *categories with families (CwF's)* and come with a type Con of contexts, a type family Ty indexed over it modelling types, and so on. This works well in versions of type theory where the principle of *unique identity proofs* (UIP) holds. In homotopy type theory (HoTT) however, it is a long-standing and frequently discussed open problem whether the type theory "eats itself" and can serve as its own interpreter. The fundamental underlying difficulty seems to be that categories are not suitable to capture a type theory in the absence of UIP.

In this paper, we develop a notion of ∞*-categories with families (*∞*-CwF's)*. The approach to higher categories used relies on the previously suggested *semi-Segal types*, with a new construction of identity substitutions that allow for both univalent and non-univalent variations. The type-theoretic universe as well as the internalised syntax are models, although it remains a conjecture that the latter is initial. To circumvent the known unsolved problem of constructing semisimplicial types, the definition is presented in *two-level type theory* (2LTT).

Apart from introducing ∞-CwF's, this paper is meant to serve as a "gentle introduction" to shortcomings of 1-categories in type theory without UIP, and to difficulties of and approaches to internal higher-dimensional categories.

Contents

This work has been supported by the Royal Society, grant No. URF\R1\191055, and the European Union, co-financed by the European Social Fund (EFOP-3.6.2-16-2017-00013, Thematic Fundamental Research Collaborations, Grounding Innovation in Informatics and Infocommunication).

1. Introduction: Formalising Type Theory

Dependent type theory in the style of Martin-Löf forms the foundation of various dependently typed programming languages and proof assistants, such as Agda, Coq, Epigram, Idris, and Lean. Numerous variations of type theory have been considered, and models are studied in order to better understand the properties of these theories. Even the study and formalisation of models of type theory *in type theory itself* is an active field of research, involving various different models such as the setoid model implemented by Palmgren [2019] and others, parametric models of type theory [Bernardy et al., 2015], or an implementation of the groupoid interpretation [Sozeau and Tabareau, 2014] by Hofmann and Streicher [1996].

In particular formalisations of the syntax of type theory, i.e. the *syntactic model* or *term model*, have received significant attention. Statements of the form "type theory should be able to handle its own meta-theory" are often made (e.g. the very first sentence by Abel et al. [2017]), which refers to formalising the (intended) *initial* model of type theory.[1] Altenkirch and Kaposi [2016] call this simply *type theory in type theory*. To avoid confusion, we refer to the type theory in which these models are implemented as the *host theory*, and the structure that get implemented as the *object theory* or simply the *model*. The expression "type theory eats itself" for this concept was suggested by Chapman [2009], inspired by Danielsson [2006] and others. Other recent work on the same goal with various techniques includes the studies by Escardó and Xu [2014], the PhD thesis by Kaposi [2016], work by Gylterud et al. [2015] and by Buchholtz [2017].

The closely related idea to formalise the notion of a model of dependent type theory inside dependent type theory itself goes back at least to Dybjer's internal type theory via *categories with families* a.k.a. *CwF's* [Dybjer, 1995] (see also work by Awodey [2018]). Formalisations of various models have since then been pursued many times; a careful comparison of different definitions of models inside type theory has been given by Ahrens et al. [2017, 2018]. A category with families consists of a category C with a terminal object, a presheaf of families on it, and a context extension operation. The presheaf is often split into two functors Ty and

[1] It is often seen as type-theoretic folklore that the initial model of a type theory coincides with the syntax, i.e. that the syntactic model is initial. To the best of my knowledge, this has not yet been proved in full generality, and one might argue that it should be called a "folklore expectation" instead; but proofs for concrete versions of the general claim are available by Streicher [1993], Brunerie and de Boer [2019], and Lumsdaine and Mörtberg [2018].

a semicategory of contexts and substitutions:

$$\mathsf{Con} \;:\; \mathcal{U} \tag{1}$$
$$\mathsf{Sub} \;:\; \mathsf{Con} \to \mathsf{Con} \to \mathcal{U} \tag{2}$$
$$\_ \diamond \_ \;:\; \mathsf{Sub}\,\Theta\,\Delta \to \mathsf{Sub}\,\Gamma\,\Theta \to \mathsf{Sub}\,\Gamma\,\Delta \tag{3}$$
$$\mathsf{assoc} \;:\; (\sigma \diamond \delta) \diamond \nu = \sigma \diamond (\delta \diamond \nu) \tag{4}$$

identity morphisms as identity substitutions:

$$\mathsf{id} \;:\; \mathsf{Sub}\,\Gamma\,\Gamma \tag{5}$$
$$\mathsf{idl} \;:\; \mathsf{id} \diamond \sigma = \sigma \tag{6}$$
$$\mathsf{idr} \;:\; \sigma \diamond \mathsf{id} = \sigma \tag{7}$$

a terminal oject as empty context:

$$\bullet \;:\; \mathsf{Con} \tag{8}$$
$$\epsilon \;:\; \mathsf{Sub}\,\Gamma\,\bullet \tag{9}$$
$$\bullet\eta \;:\; (\sigma : \mathsf{Sub}\,\Gamma\,\bullet) = \epsilon \tag{10}$$

a presheaf of types:

$$\mathsf{Ty} \;:\; \mathsf{Con} \to \mathcal{U} \tag{11}$$
$$\_[\_]^{\mathsf{T}} \;:\; \mathsf{Ty}\,\Delta \to \mathsf{Sub}\,\Gamma\,\Delta \to \mathsf{Ty}\,\Gamma \tag{12}$$
$$[\mathsf{id}]^{\mathsf{T}} \;:\; A[\mathsf{id}]^{\mathsf{T}} = A \tag{13}$$
$$[\diamond]^{\mathsf{T}} \;:\; A[\sigma \diamond \delta]^{\mathsf{T}} = A[\sigma]^{\mathsf{T}}[\delta]^{\mathsf{T}} \tag{14}$$

a (covariant) presheaf on the category of elements as terms:

$$\mathsf{Tm} \;:\; (\Gamma : \mathsf{Con}) \to \mathsf{Ty}\,\Gamma \to \mathcal{U} \tag{15}$$
$$\_[\_]^{\mathsf{t}} \;:\; \mathsf{Tm}\,\Delta\,A \to (\sigma : \mathsf{Sub}\,\Gamma\,\Delta) \to \mathsf{Tm}\,\Gamma\,(A[\sigma]^{\mathsf{T}}) \tag{16}$$
$$[\mathsf{id}]^{\mathsf{t}} \;:\; t[\mathsf{id}]^{\mathsf{t}} = t \qquad \text{over } [\mathsf{id}]^{\mathsf{T}} \tag{17}$$
$$[\diamond]^{\mathsf{t}} \;:\; t[\sigma \diamond \delta]^{\mathsf{t}} = t[\sigma]^{\mathsf{t}}[\delta]^{\mathsf{t}}t \qquad \text{over } [\diamond]^{\mathsf{T}} \tag{18}$$

context extension, modelled by representability:

$$\_ \triangleright \_ \;:\; (\Gamma : \mathsf{Con}) \to \mathsf{Ty}\,\Gamma \to \mathsf{Con} \tag{19}$$
$$\mathsf{p} \;:\; \mathsf{Sub}\,(\Gamma \triangleright A)\,\Gamma \tag{20}$$
$$\mathsf{q} \;:\; \mathsf{Tm}\,(\Gamma \triangleright A)\,(A[\mathsf{p}]^{\mathsf{T}}) \tag{21}$$
$$\_\,,\,\_ \;:\; (\sigma : \mathsf{Sub}\,\Gamma\,\Delta) \to \mathsf{Tm}\,\Gamma\,(A[\sigma]^{\mathsf{T}}) \to \mathsf{Sub}\,\Gamma\,(\Delta \triangleright A) \tag{22}$$
$$\triangleright\beta_1 \;:\; \mathsf{p} \diamond (\sigma, t) = \sigma \tag{23}$$
$$\triangleright\beta_2 \;:\; \mathsf{q}[\sigma, t]^{\mathsf{t}} = tt \qquad \text{over } [\diamond]^{\mathsf{T}} \text{ and } \triangleright\beta_1 \tag{24}$$
$$\triangleright\eta \;:\; (\mathsf{p}, \mathsf{q}) = \mathsf{id} \tag{25}$$
$$,\diamond \;:\; (\sigma, t) \diamond \nu = (\sigma \diamond \nu, t[\nu]^{\mathsf{t}})t \qquad \text{over } [\diamond]^{\mathsf{T}} \tag{26}$$

FIGURE 1. The formulation of a category with families (CwF) as a generalised algebraic theory (GAT), as given by Kaposi et al. [2019b,a]. Presentation-wise, it slightly differs from (but is equivalent to) the similar suggestion by Altenkirch and Kaposi [2016]. Above, the components are reordered and regrouped to make the connection to CwF's more visible. *Over* refers to substitution (a.k.a. transport) in the terminology of the Univalent Foundations Program [2013, Chp. 2.3]. Implicit arguments are omitted. $\mathcal{U}$ is a universe at any level.

Tm. The idea is that C is the category of contexts and substitutions (a.k.a. context morphisms) of a type theory, where the terminal object is the empty context. Ty represents the types in each contexts, and Tm the terms for a given context and type.

CwF's can easily be presented as a *generalised algebraic theory* (GAT) as introduced by Cartmell [1978, 1986]. A GAT consists of *sorts*, *operations*, and *equations*. In the case of CwF's, the sorts are contexts, substitutions, types, and terms. Examples for operations are composition of substitutions, identity substitutions, and context extension. The equalities include the associativity law for composition, identity laws, laws expressing that types and terms form functors, and so on. In type theory, sorts can be represented as types and type families. This includes a type $\mathsf{Con} : \mathcal{U}$ for contexts and families $\mathsf{Sub} : \mathsf{Con} \to \mathsf{Con} \to \mathcal{U}$ as well as $\mathsf{Ty} : \mathsf{Con} \to \mathcal{U}$ for substitutions and types.[2] Operations are functions with sorts as codomains. For example, given a type $A : \mathsf{Ty}\,\Delta$ and a substitution $\sigma : \mathsf{Sub}\,\Gamma\,\Delta$, we need an operation (often written as $\_[\_]$) which gives us a new type $A[\sigma] : \mathsf{Ty}\,\Gamma$. Equalities are stated using Martin-Löf's *identity type*, also known as *equality type*, *path type*, or *identification type*, denoted by $a = b$.[3] As an example, we need an equality of the form $(A[\sigma])[\tau] = A[\sigma \diamond \tau]$. A full presentation of type theory in this form, similar to the one developed by Altenkirch and Kaposi [2016], is shown in Fig. 1.[4] If one works in a type theory that has *quotient inductive-inductive* [Altenkirch et al., 2018] or *higher inductive-inductive types* [Kaposi and Kovács, 2020], then defining a CwF as a GAT automatically gives a formalisation of the initial model (the intended syntax).

Besides the initial model, a very important interpretation is the *standard model* (see e.g. [Altenkirch and Kaposi, 2016, Coquand et al., 2019]), sometimes known as the *meta-circular interpretation*. It is based on the (trivial) observation that the category of [small] types can be equipped with the structure of a CwF in a canonical way. In detail, contexts and substitutions in the standard model are [small] types and functions of the host theory, types of the model are dependent types of the host theory, and terms are dependent functions.

We have a morphism of models from the initial model to any other model, and in particular to the standard model. This means that a universe in the host theory is a model of type theory itself. From a programming perspective, this is interesting as it automatically gives us an interpreter: We formalise the syntax of type theory inside type theory, and anything we do with this syntax can be interpreted as a construction in the host theory. As Shulman [2014] argues, we should expect that this is possible if we want to view type theory as a general purpose programming language, as any such language should be able to implement its own interpreter or compiler.

This motivates us to formulate the following somewhat vague expectation of a general definition of a model in type theory:

---

[2] $\mathcal{U}$ is a universe. We omit universe indices everywhere and implicitly use universe polymorphism.

[3] At this point, it is important to emphasis the difference between the internal equality type $a = b$ and the meta-theoretic *judgmental equality*, also known as *definitional equality*, written as $a \equiv b$.

[4] Regarding notation: In the host theory, we use Agda-style notation for dependent function types and write $(x : A) \to B\,x$ instead of $\Pi(x : A).B(x)$. Dependent pair types are denoted as $\Sigma(x : A).B(x)$. We reserve the symbol $\circ$ for function composition in the host theory and denote composition of substitutions (context morphisms) in the model by $\diamond$. Implicit arguments in the host theory are denoted by $\{x : A\}$, but they are completely omitted in Fig. 1 for readability.

General Problem: In a (given specific version of) dependent type theory, define the notion of *model* such that both the initial model and the standard model can be constructed, and such that the initial model can reasonably be viewed as syntax.

In intensional dependent type theory with the axiom of *unique identity proofs (UIP)* and quotient inductive-inductive types Altenkirch et al. [2018], the problem has been solved by Altenkirch and Kaposi [2016] who construct the initial model and the standard interpretation. Recall that UIP states that parallel equalities are always equal for all types X,

$$(x\, y : X) \to (p\, q : x = y) \to (p = q). \tag{27}$$

Kaposi and Altenkirch [2017] further show that the initial model has *decidable equality*, e.g. for $\Gamma : \mathsf{Con}$ we have

$$(A\, B : \mathsf{Ty}\, \Gamma) \to (A = B) + \neg(A = B). \tag{28}$$

This is certainly expected since the equality type $=$ of the host theory models definitional/judgmental equality, decidability of which is needed for type checking. This arguable covers the part that the initial model can reasonably be viewed as syntax.

We are interested in settings where UIP is not assumed, which makes the problem much harder. In particular, it is a long-standing open problem whether (and how well) *homotopy type theory* (HoTT) [Univalent Foundations Program, 2013], which rejects UIP, is able to handle its own meta-theory. The question was first raised by Shulman [2014] and has since then been discussed very actively. Escardó and Xu [2014], Gylterud et al. [2015], Buchholtz [2017] present work on the question. Progress in the setting without UIP has also been made by Abel et al. [2017]. Nevertheless, the core of the question is still an open problem.

If we define a CwF in homotopy type theory to consist of the components in Fig. 1, maybe with additional base types, universes, or Π- and Σ-types as suggested for example by Kaposi et al. [2019a], we have to decide how we react to the absence of UIP. If we simply ignore it, the equality types will not be well-behaved and destroy expected properties, a very common phenomenon in homotopy type theory. In particular, the initial model will not satisfy UIP and will not have decidable equality, making it unsuitable as "syntax" in any form (in other words: the expected term model will not be initial). The typical strategy to address this problem is to explicitly include a condition which ensures that all involved types are truncated at a certain level. The notion of *category* considered by Ahrens et al. [2015b] requires morphisms to form a set (i.e. are 0-truncated) to ensure well-behavedness. Unfortunately, this would not be a satisfactory solution in our situation. Universes in homotopy type theory are not truncated [Kraus and Sattler, 2015], which means that the standard "model" will cease to be a model.

The fundamental underlying issue is that (ordinary) categories are not suitable to talk about non-truncated structures internally in homotopy type theory. The *laws* of category theory, such as associativity or identity laws, are not appropriately modelled by the internal equality type. In fact, equalities in a theory without UIP are *data* rather than *properties*, which means that an internal equality expressing associativity is much more similar to an isomorphism in a bicategory than a law in a (1-)category. Being even more precise, we should say that an equality behaves like an isomorphism in an $(\infty, 1)$-category (for simplicity called ∞-category; a survey has been given by Bergner [2010]), since universes in homotopy type theory are not truncated at any level.

This motivates the current paper. We define the notion of a model of type theory using ∞-categories rather than ordinary categories. In a nutshell, doing so corresponds to adding all higher coherences which are missing from Fig. 1, turning it into an *∞-category with families*. The main difficulty is that ∞-CwF's need an infinite tower of data, and this data needs to be organised appropriately. It is in particular unknown whether this structure can be represented in the version of homotopy type theory developed by the Univalent Foundations Program [2013]. To circumvent this problem, we work in a *two-level type theory* [Voevodsky, 2013, Annenkov et al., 2019] which makes the construction possible. However, all finite special cases *can* be expressed in "standard HoTT." Thus, we can define what an $(n, 1)$-CwF is, significantly generalising the ordinary definition of a CwF, in HoTT.

The specific approach to ∞-categories that we work with is based on the *semi-Segal types* suggested by Capriotti [2016], Capriotti and Kraus [2017], Annenkov et al. [2019], and Schreiber [2012]. While this approach immediately gives a definition of an *∞-semicategory*, adding the identities in a well-behaved manner is trickier that one might expect. All the authors of the work mentioned above use identities which have *univalence* built-in. This is natural in homotopy type theory, but not necessarily desirable when considering models since we would not expect the syntactic model to be univalent (and even if we do, previous formalisations of the syntax are not univalent and would not be captured by a formulation that requires univalence). We therefore introduce a new definition of identities, namely *idempotent equivalences*. We prove that our identity structure turns out to be unique (propositional) and is therefore automatically fully coherent. A further propositional property expressing univalence can be added optionally. In order to define ∞-CwF's, we naturally have to develop some notions of ∞-category theory inside type theory such as ∞-functors or the ∞-category of elements of an ∞-presheaf.

We prove that several natural constructions and examples which are broken for ordinary CwF's (based on 1-categories) work when we move to ∞-categories, indicating that the latter are much more well-behaved. We also discuss three additional properties which an ∞-CwF can have. First, an ∞-CwF can be purely based on sets (i.e. on types satisfying UIP), such as the syntax. Second, an ∞-CwF can be univalent, such as the standard model. And third, it can be *finite-dimensional*. The latter is particularly interesting since it can be formulated in homotopy type theory, without the need for 2LTT.

Related work. The original connection between higher categories (in the form of spaces) and type theory, discovered independently by Awodey and Warren [2009] and by Voevodsky (see the presentation by Kapulkin and Lumsdaine [2012]), may marked the beginning of the study of homotopy type theory and univalent foundations. While Voevodsky's model of the univalence axiom uses simplicial structures, the superficial similarity to the work in this paper may be somewhat misleading; in fact, the motivation for the appearance of higher categories is reversed. Voevodsky's simplicial set model uses higher categories in order to model the equality type of the type theory that *is* modelled, while we are using higher categories because the *host theory* is unsuitable for working with ordinary categories. While Kapulkin and Lumsdaine [2012] define one particular model, the current paper defines a type ("classifier") of models. Similarly, Barras et al. [2015] construct a model interpreting types as semisimplicial sets. Boulier [2018] and other authors discuss and formalise various models assuming UIP locally or globally. Abel et al. [2017] formalise the syntax in without UIP, but their induced notion of model includes non-well-typed components which are not present in a type-theoretic universe; thus, the standard interpretation is not a model, and their approach does not give a solution to the *General Problem* discussed above. Capriotti [2016], Annenkov et al. [2019]

and Capriotti and Kraus [2017] suggest complete semi-Segal types as a model for ∞-categories, a variation of which we use in this paper (avoiding completeness/univalence). Nguyen and Uemura [2020] propose the development of ∞-*type theories*, where definitional equalities are replaced by homotopies; although their proposal is not worked out in full at the time of writing, we speculate that our ∞-CwF are in principle general enough to also be a suitable notion of model for ∞-type theories. Capriotti and Sattler [2020] build an interpretation based on Segal types for a specific type theory and prove the equivalence between the initiality and induction principle for higher inductive-inductive types.

Overview of the paper. Section 2 discusses the formalisation of 1-categorical models, in the form of CwF's, in a type theory with UIP. This approach does not work anymore when UIP is dropped as we see in Section 3, where several examples that break without UIP are studied. For our higher-categorical approach, we want to use semisimplicial types in two-level type theory, both introduced in Section 4. The heart of the paper is Section 5, where we develop sufficiently much internal ∞-category theory so that we can define ∞-CwF's. In Section 6, we prove that all the examples that were broken without UIP work again when using ∞-CwF's, and in Section 7, we discuss variations including univalent and finite dimensional CwF's. In Section 8, we discuss open problems and conjectures, and conclude.

Specification of the host type theory. We work in dependent type theory (the "host theory") with Σ- and Π-types satisfying their respective judgmental η-rule, and with a hierarchy of universes. We always assume function extensionality. In Section 2, we assume that the host theory satisfies UIP. From Section 3 on, we drop this assumption and work in homotopy type theory with all features introduced by the Univalent Foundations Program [2013]. In particular, we use the *path over* notation that is common in homotopy type theory and expresses substitution/transport ($a =_p b$ means $\mathsf{subst}(p, a) = b$).

From Section 4.2 on, we assume that the host theory is equipped with a second kind of equality type, turning it into a two-level type theory. This theory is further specified in the mentioned section.

2. CwF's: One-Categorical Models of Dependent Type Theory

2.1. **Motivation and categorical definition.** In order to define higher- or ∞-categorical internal models of type theory, we need to find a suitable one-dimensional formulation which can serve as a starting point for generalisations. In the current section, we therefore want to work with types that do not possess non-trivial higher structure. This means that all types in this section are assumed to satisfy the principle of *unique identity proofs (UIP)*, i.e. are *(homotopy) sets*.

In the introduction we have seen Fig. 1 which presents the components of type theory as a generalised algebraic theory and is due to Kaposi et al. [2019b,a]. In detail, a *model of type theory* according to Fig. 1 consists of four types and type families (Con, Ty, Sub, Tm), together with ten terms that inhabit the four type families in various ways, and together with twelve equations. The original work by Altenkirch and Kaposi [2016] (as well as [Kaposi et al., 2019a,b]) contains additional components for various type formers. These and the twenty-six components of Fig. 1 are then viewed as signatures and constructors of a *quotient inductive-inductive type (QIIT)* which defines the initial model (the "syntax"). In the current paper, we are interested in models in general, not only the initial such model. Moreover, the initial model of Fig. 1 is rather uninteresting since we have not added base types, meaning that the initial model has only the empty context and no types.

A more compact description of a model of type theory is given by a CwF [Dybjer, 1995]. Before stating its definition, let us recall the following basic categorical constructions. Let $\mathcal{D}$ be a category and F be a functor from $\mathcal{D}$ to the category of sets Set. The *category of elements* is denoted by $\int_{\mathcal{D}} F$. Its objects are the pairs $\{(d, x) \mid d \in \mathcal{D}_0, x \in F(d)\}$ and a morphism from (d, x) to (e, y) is a morphism $f \in \mathcal{D}_1(x, y)$ such that $F(f)(x) = y$. For $d \in \mathcal{D}_0$, the *slice category* $\mathcal{D}/d$ has as objects those morphisms of $\mathcal{D}$ which have d as codomain, while morphisms are those of $\mathcal{D}$ which make the evident triangles commute. Finally, the functor F is *represented* by $d \in \mathcal{D}_0$ if F is naturally isomorphic to $\mathcal{D}_1(d, \_)$. By the Yoneda lemma, this is equivalent to having an element $x \in F(d)$ such that (d, x) is initial in $\int_{\mathcal{D}} F$. We follow standard terminology and call d the *representing object* and x *universal element*.

Definition 1 (CwF)**.** A *category with families (CwF)* is given by:

(i) a category $\mathcal{C}$ with a terminal object,
(ii) a presheaf $\mathsf{Ty} : \mathcal{C}^{\mathsf{op}} \to \mathsf{Set}$; we will write $A[\sigma]^{\mathsf{T}}$ instead of $\mathsf{Ty}(\sigma)(A)$;
(iii) a functor $\mathsf{Tm} : \left(\int_{\mathcal{C}^{\mathsf{op}}} \mathsf{Ty}\right) \to \mathsf{Set}$; for $t \in \mathsf{Tm}(\Gamma, A)$, we will write $t[\sigma]^{\mathsf{t}}$ instead of $\mathsf{Tm}(\sigma)(t)$,
(iv) and, for every $\Delta \in \mathcal{C}_0$ and $A \in \mathsf{Ty}(\Delta)$, an object $\Delta \triangleright A$ together with a morphism $p_A : \mathcal{C}_1(\Delta \triangleright A, \Delta)$ which represents the functor

$$(\mathcal{C}/\Delta)^{\mathsf{op}} \to \mathsf{Set} \tag{29}$$

$$(\Gamma, \sigma) \mapsto \mathsf{Tm}(\Gamma, A[\sigma]^{\mathsf{T}}). \tag{30}$$

2.2. **CwF's as generalised algebraic theories.** Fig. 1 can be seen as a direct implementation of Definition 1 in type theory. The correspondence is made clear by the sub-headlines in the figure. Let us go through the points in Definition 1:

(i) The category $\mathcal{C}$ in Definition 1 is the category consisting of the first two groups of data in Fig. 1; the reason for splitting this category in a semicategory and separate identities will become clear later. The terminal object is self-explanatory.
(ii) The presheaf Ty is implemented in the straighforward way.
(iii) For the functor $\mathsf{Tm} : \left(\int_{\mathcal{C}^{\mathsf{op}}} \mathsf{Ty}\right) \to \mathsf{Set}$, the object part is directly given by Eq. (15). The type-theoretic version of the morphism part makes use of a standard simplification: Having two objects (Γ, B) and (Δ, A) in $\int_{\mathcal{C}^{\mathsf{op}}} \mathsf{Ty}$ together with a morphism $(\sigma : \mathcal{C}_1(\Gamma, \Delta), e : B = \mathsf{Ty}(\sigma)(A))$ is as good as having only Γ, Δ, σ, and A. Thus, the morphism part of Tm can be stated as

$$\begin{aligned} \_[\_]^{\mathsf{t}} : \quad & \{\Gamma\ \Delta : \mathsf{Con}\} \to \{A : \mathsf{Ty}(\Delta)\} \to \\ & (\sigma : \mathsf{Sub}\,\Gamma\,\Delta) \to \mathsf{Tm}\,\Delta\,A \to \mathsf{Tm}\,\Gamma\,(A[\sigma]^{\mathsf{T}}) \end{aligned} \tag{31}$$

The version (16) omits the implicit arguments and swaps the explicit arguments.
(iv) Given Δ and A, the pair $(\Delta \triangleright A, p)$ is the representing object, while q is the universal element. Initiality of the object $(\Delta \triangleright A, p, q)$ in the category of elements of Tm translates in type theory to: For all $\sigma : \mathsf{Sub}\,\Gamma\,\Delta$ and $t : \mathsf{Tm}\,\Gamma\,(A[\sigma]^{\mathsf{T}})$, the type

$$\Sigma(\gamma : \mathsf{Sub}\,\Gamma\,(\Delta \triangleright A)).\Sigma(e : \sigma = p \diamond \gamma).(q[\sigma]^{\mathsf{t}} =_e t) \tag{32}$$

is contractible (i.e. has a unique inhabitant). The inhabitant ("centre of contraction") is given by $\gamma :\equiv (\sigma, t)$ together with the equations $\triangleright\beta_1$ and $\triangleright\beta_2$. Having shown that there is a suitable γ, we need to check that it is the only one; thus, assume we have γ' satisfying the equations. It then follows

that

$$\begin{array}{lcl} & & \gamma' \\ \textit{by } (\mathsf{idl}) \textit{ and } (\triangleright\eta) & = & \gamma' \diamond (p, q) \\ \textit{by } (,\diamond) & = & (p \diamond \gamma', q[\gamma']^{\mathsf{t}}) \\ \textit{by assumption} & = & (\sigma, t). \end{array} \tag{33}$$

Thanks to the assumption that all involved types satisfy UIP, the equations are automatically unique. Vice versa, from contractibility of (32), one can derive the components $(,\diamond)$ and $(\triangleright\eta)$.

2.3. **Examples.** The literature covers many examples for CwF's. For us, the following standard and well-known examples are particularly interesting:

Example 2 (syntax/initial model)**.** In a version of type theory with *quotient inductive-inductive types (QIITs)* [Altenkirch et al., 2018, Kaposi and Kovács, 2018], we can read Fig. 1 directly as a QIIT signature. There is an implied notion of model morphism, and the QIIT is automatically the initial such model. This is how the construction is presented by Altenkirch and Kaposi [2016]. The initial CwF without base types consists of only the empty context, but they demonstrate that it is easy to add further components: Unit, Bool, Σ- and Π-types as well as a "universe" $(\mathsf{U}, \mathsf{El})$.

Kaposi and Altenkirch [2017] show that the QIIT of the type theory with all the mentioned components has decidable equality. This is important if we want to view the initial model as formalised syntax, and from this point of view, the internal equality type plays the role of the meta-theoretic definitional equality, which in most type theories is required to be decidable.

Example 3 (standard model)**.** The *standard model* is the CwF of types and functions: each expression in Fig. 1 is interpreted by its canonical "semantic counterpart". We need a fixed [small] universe $\mathcal{U}$ to define this internally. Then, the standard model is given by:

$$\mathsf{Con} :\equiv \mathcal{U} \tag{34}$$
$$\mathsf{Sub}\,\Gamma\,\Delta :\equiv \Gamma \to \Delta \tag{35}$$
$$\delta \diamond \sigma :\equiv \delta \circ \sigma \tag{36}$$
$$\mathsf{id} :\equiv \lambda x.x \tag{37}$$
$$\bullet :\equiv \mathsf{Unit} \tag{38}$$
$$\mathsf{Ty}\,\Gamma :\equiv \Gamma \to \mathcal{U} \tag{39}$$
$$A[\sigma]^{\mathsf{T}} :\equiv A \circ \sigma \tag{40}$$
$$\mathsf{Tm}\,\Gamma\,A :\equiv \Pi(x : \Gamma).(A\,x) \tag{41}$$
$$t[\sigma]^{\mathsf{t}} :\equiv t \circ \sigma \tag{42}$$
$$\Gamma \triangleright A :\equiv \Sigma(x : \Gamma).(A\,x) \tag{43}$$
$$\mathsf{p} :\equiv \mathsf{proj}_1 \tag{44}$$
$$\mathsf{q} :\equiv \mathsf{proj}_2 \tag{45}$$
$$(\sigma, t) :\equiv \lambda x.(\sigma\,x, t\,x) \tag{46}$$
$$\cdots$$

Several authors (e.g. Altenkirch and Kaposi [2016], Escardó and Xu [2014]) have noticed that all equations in this model hold definitionally (i.e. by refl): function composition is definitionally associative (assuming η for Π-types), and so on. This will play a role later in the current paper.

Example 4 (modelling an axiom)**.** Assume we are given a set-CwF $\mathcal{C}$. This model may have additional types and type formers such as Π-types and universes. We now may want a model where a global assumption is satisfied, i.e. the axiom of function extensionality. Such a model can be created by fixing the context Γ_0 to contain a term which witnesses function extensionality, and constructing the slice $\mathcal{C}/\Gamma_0$. As usual, contexts of $\mathcal{C}/\Gamma_0$ are pairs $(\Delta : \mathsf{Con}, \delta : \mathsf{Sub}\,\Delta\,\Gamma_0)$, and a morphism between (Δ, δ) and (Φ, φ) is a pair $(f : \mathsf{Sub}\,\Delta\,\Phi, e : \delta = \varphi \diamond f)$. The other components of the new model are those given by $\mathcal{C}$.

Section summary. Dybjer's notion of a model of type theory, a *category with families (CwF)*, can be represented as a generalised algebraic theory and implemented in dependent type theory. The most important examples of models are the *initial model*, which one expects to coincide with an internal representation of the syntax, and the *standard model* of types and functions, which provides the "obvious semantics". Both models have been studied and work well in a type theory with UIP.

3. Challenges in Type Theory without UIP

As discussed in the previous section, there are clear and extensively studied strategies for the development of models of type theory via CwF's in a dependent type theory which satisfies the principle of unique identity proofs (UIP).

The purpose of the section is to demonstrate that 1-categorical methods may be unsatisfactory in a setting without UIP. The underlying question is: *What is a good notion of a category?* We can add a truncation condition explicitly and accept that this may exclude cases that of interest, as it is done e.g. by Ahrens et al. [2015b]. If we choose not to add such a condition, the types may behave differently than expected.

In this section, we assume that we work in the version of homotopy type theory that is developed by Univalent Foundations Program [2013]. In principle, the examples that we give apply in any version of dependent type theory which does not (or not necessarily) validate the principle UIP.

3.1. Deficiency of Truncated Structure.

In order to reproduce the constructions from Section 2 in HoTT or other dependent type theories without UIP, we could simply modify the definition of a model (i.e. of a CwF) by adding the requirement that the involved types and type families are sets anyway:

Definition 5 (set-CwF)**.** A *set-CwF* is a 30-tuple $(\mathsf{Con}, \mathsf{Sub}, \ldots)$ of the 26 components given in Fig. 1, together with four components:

$$\mathsf{conset}: \ \mathsf{isSet}(\mathsf{Con}) \tag{47}$$

$$\mathsf{tyset}: \ (\Gamma : \mathsf{Con}) \to \mathsf{isSet}(\mathsf{Ty}\,\Gamma) \tag{48}$$

$$\mathsf{subset}: \ (\Gamma\,\Delta : \mathsf{Con}) \to \mathsf{isSet}(\mathsf{Sub}\,\Gamma\,\Delta) \tag{49}$$

$$\mathsf{tmset}: \ (\Gamma : \mathsf{Con}) \to (A : \mathsf{Ty}\,\Gamma) \to \mathsf{isSet}(\mathsf{Tm}\,\Gamma\,A) \tag{50}$$

Formally, a set-CwF is an element of an iterated Σ-type, or, if supported by the theory, an element of a record type with 30 entries.

As long as we only care about the "sub-theory of sets", we can do everything with set-CwF's that we could do in the setting with UIP. In particular, assuming the type theory supports QIITs, we can define the initial set-CwF as in Example 2. The main drawback is the observation that the *standard model* will no longer work:

Example 6 (standard model; continuing Example 3)**.** In HoTT, the [lowest or higher] universe $\mathcal{U}$ is not a set. Thus, the standard model presented in Example 3 is not a set-CwF.

A very weak version of the standard model is still possible: We can define a "universe of propositions" as $\mathsf{Prop} :\equiv \Sigma(X : \mathcal{U}).\mathsf{isProp}(X)$. Since this is a set, it can be used to build a model as in Example 3. However, as explained in the introduction, the non-existence of the general "standard model" is unsatisfying. One possible view is that, unlike many other programming languages, HoTT cannot serve as its own interpreter with this approach.

The situation does not become better if we work with *univalent* 1-categories [Ahrens et al., 2015b] instead, where substitutions have to form a set and contexts a 1-type. Any requirement for components to be of restricted truncation level leads to the issue in Example 6.

3.2. **Deficiency of Wild/Incoherent Structure.** Another approach would be to "ignore" the fact that we do not have UIP, and simply use the definition which works well in a setting with UIP.

Definition 7 (wild CwF)**.** A *wild CwF* is a 26-tuple $(\mathsf{Con}, \mathsf{Sub}, \ldots)$ of all the components in Fig. 1. In other words, a wild CwF is a set-CwF without the four components in Definition 5.

The universe forms a wild CwF: the construction in Example 3 works word for word, since the assumption of UIP was never used. Unfortunately, the absence of UIP breaks the other constructions:

Example 8 (initial model; continuing Example 2)**.** In order for the initial model to be non-trivial, let us assume that we add a base type or a family of base types to the specification of a CwF, as for example demonstrated by Altenkirch and Kaposi [2016] or Kaposi et al. [2019a]. Assuming we work in HoTT with higher inductive-inductive types, we can take the specification as the signature for a higher inductive-inductive type with the rules suggested by Kaposi and Kovács [2018], which directly gives us the initial wild CwF. However, compared to the setting with UIP, we lose decidability of equality. If we take a non-empty context Θ, we have the trivial substitution $\mathsf{id} : \mathsf{Sub}\,\Theta\,\Theta$. We also have $\mathsf{idl} : \mathsf{id} \diamond \mathsf{id} = \mathsf{id}$ as well as $\mathsf{idr} : \mathsf{id} \diamond \mathsf{id} = \mathsf{id}$. One can then prove that these two equalities are not equal, and $\mathsf{Sub}\,\Theta\,\Theta$ does therefore *not* have unique equality proofs, i.e. is not a set. By Hedberg's theorem [Hedberg, 1998], this means that $\mathsf{Sub}\,\Theta\,\Theta$ does not have decidable equality.

If we consider wild CwF's with all the type formers and base types considered by Altenkirch and Kaposi [2016], it is easy to see that not only substitutions, but also contexts, types, and terms lose decidable equality.

Example 9 (modelling an axiom; continuing Example 4)**.** The construction of the *slice CwF* is broken for wild CwF's. To be precise, the slice construction is already broken for wild categories (and even wild semicategories) before even considering types, terms, or context extension. In a nutshell, when doing the slice construction of a (higher) category, a component at "level" n needs components of the original category at level n and $n + 1$: slice-objects need objects and morphisms, slice-morphisms need morphisms and composition, slice-composition needs composition and associativity, slice-associativity needs associativity and the coherence known as *Mac Lane's pentagon* for bicategories. In Example 4, UIP makes this coherence automatic. Here, it is simply impossible to prove associativity.

The two issues that we have identified in Examples 8 and 9 are instances of the same underlying problem, namely data that is missing from the definition of a wild CwF. In the first case, we need one datum that gives us $\mathsf{idl} =_{\mathsf{id}\diamond\mathsf{id}=\mathsf{id}} \mathsf{idr}$, and in the second case, we need a datum giving us the "pentagon coherence". Unsurprisingly, naïvely adding these to the definition of a wild CwF creates the need to add even more data, and so on. The approach advocated in this paper, i.e. to use higher categories, means that we essentially add *all* data.

Section summary. When implementing models of type theory inside homotopy type theory using CwF's, one has to choose whether or not to impose a condition on the truncation levels of the involved types. Both possibilities can be unsatisfactory.

4. Infinite Structures in Type Theory

Definition 5 defines a wild CwF to be a tuple with 26 components. Even more extreme, Definition 7 defines a set-CwF to consist of 30 components. While this is already tedious to express in type theory, it is entirely straightforward to express such a definition simply by listing the components. Formally, a CwF is an element of a nested Σ-type or, if supported by the type theory, a record type (we view records as syntactic sugar for nested Σ-types).

However, what happens if we want to define a structure to consist of not only 30, but infinitely many components? Often, this can be encoded in a finite way. To give a trivial example, an infinite sequence of elements of a type A is simply given as the function type $\mathbb{N} \to A$ (or as a coinductive type of streams, although such coinductive definitions can be translated to homotopy type theory without coinduction [Ahrens et al., 2015a]).

A non-trivial example for a structure with infinitely many components is a *semisimplicial type* [Voevodsky et al., 2013], and it is a long-standing open problem whether a type capturing this structure can be defined in homotopy type theory. Nevertheless, this structure is important for the approach to ∞-categories that we chose in this paper. In this section, we first define what semisimplicial types are and then introduce the setting of *two-level type theory (2LTT)*, an extension of HoTT which allows the treatment of certain structures with infinitely many components.

4.1. **Semisimplicial Types.** A semisimplicial type of level 2 is a tuple (A_0, A_1, A_2) of types and type families as follows:

$$A_0 : \mathcal{U} \tag{51}$$

$$A_1 : A_0 \to A_0 \to \mathcal{U} \tag{52}$$

$$A_2 : \{x_0\, x_1\, x_2 : A_0\} \to (A_1\, x_0\, x_1) \to (A_1\, x_1\, x_2) \to (A_1\, x_0\, x_2) \to \mathcal{U} \tag{53}$$

We think of A_0 as a type of *points* or *vertexes*. For two vertexes x_0 and x_1 we think of $A_1\, x_0\, x_1$ as a type of *lines* between the vertexes. For three vertexes and three lines forming a triangle, A_2 is the type of triangle fillers.

A semisimplicial type of level 3 is a tuple (A_0, A_1, A_2, A_3) with A_0, A_1, A_2 as above, and A_3 being a family of types that is indexed over four vertexes, six lines, and four triangle fillers which form the boundary of a tetrahedron. Although harder to imagine geometrically, it is clear on an intuitive level how the next step would look like: a semisimplicial type of level 4 adds a component A_4, which is a type family indexed over the boundary of a 4-dimensional tetrahedron, consisting of five vertexes, ten lines, ten triangle fillers, and five tetrahedron fillers.

The mentioned long-standing open problem of homotopy type theory asks whether it is possible to *define* semisimplicial types internally, i.e. whether it is possible to write down a type family $F : \mathbb{N} \to \mathcal{U}_1$ (where $\mathcal{U}_1$ is a universe that contains $\mathcal{U}$) such that $F(n)$ encodes the type of tuples $(A_0, \dots, A_n)$. The problem was first discussed between Voevodsky, Lumsdaine, and others during the Univalent Foundations special year program at the IAS in Princeton 2012/13 and has since then acquired significant attention. Proposals for partial solutions, or extensions of type theory where solutions are possible, have been suggested by Voevodsky [2013], Herbelin [2015], Part and Luo [2015], Altenkirch et al. [2016], and others.

Meta-theoretically, a semisimplicial type represents a certain contravariant functor from the category Δ_+ to the category of (small) types. This works as follows. For a universe $\mathcal{U}$, the category of types in $\mathcal{U}$ has types as objects and functions as morphisms. The categorical laws hold judgmentally/definitionally due to the η-law for Π-types. Overloading notation, we denote this category by $\mathcal{U}$. The category Δ_+ has natural numbers as objects. For a number k, we write $[k]$ for the finite set of numbers $\{0, 1, \ldots, k\}$. Morphisms of Δ_+ are strictly monotone functions,

$$\Delta_+(i,j) :\equiv \{\, f : [i] \to [j] \mid f \text{ strictly increasing} \,\}. \tag{54}$$

We denote the full subcategory of Δ_+ with objects [0], [1], [2] by $\Delta_+^{\leq 2}$. Drawing only generating morphisms, this category can be depicted as

$$[0] \rightrightarrows [1] \;\substack{\rightarrow\\ \rightarrow\\ \rightarrow}\; [2] \tag{55}$$

Given a semisimplicial type of level 2 as above, the functor $\left(\Delta_+^{\leq 2}\right)^{\mathsf{op}} \overset{s}{\Rightarrow} \mathcal{U}$ that it represents is given by

$$A_0 \;\substack{\leftarrow\\ \leftarrow}\; \Sigma(x_0\, x_1 : A_0).A_1\, x_0\, x_1 \;\substack{\leftarrow\\ \leftarrow\\ \leftarrow}\; \begin{array}{r}\Sigma(x_0\, x_1\, x_2 : A_0).\\ \Sigma(x_{01} : A_1\, x_0\, x_1).\\ \Sigma(x_{12} : A_1\, x_1\, x_2).\\ \Sigma(x_{02} : A_1\, x_0\, x_2).\\ A_2\, x_{01}\, x_{12}\, x_{02}\end{array} \tag{56}$$

The function between the types are simply projections, ensuring that the functor laws hold definitionally (thanks to η for Σ-types).

Given a semisimplicial type A, we can consider an *A-indexed family* of semisimplicial types. With the view of a simplicial type as a functor, we can also call this a semisimplicial type *E over A*. Concretely, if we have $A \equiv (A_0, A_1, A_2)$ as above, a semisimplicial type E of level 2 over A consists of:

$$E_0 : A_0 \to \mathcal{U} \tag{57}$$

$$E_1 : \{x_0\, x_1 : A_0\} \to (x_{01} : A_1\, x_0\, x_1) \to (x'_0 : E_0\, x_0) \to (x'_1 : E_1\, x_1) \to \mathcal{U} \tag{58}$$

$$\begin{aligned} E_2 : {} & \{x_0\, x_1\, x_2 : A_0\} \to \{x_{01} : A_1\, x_0\, x_1\} \to \{x_{12} : A_1\, x_1\, x_2\} \to \{x_{02} : A_1\, x_0\, x_2\} \to \\ & (x_{012} : A_2\, x_{01}\, x_{12}\, x_{02}) \to \{x'_0 : E_0\, x_0\} \to \{x'_1 : E_0\, x_1\} \to \{x'_2 : E_0\, x_2\} \to \\ & (E_1\, x_{01}\, x'_0\, x'_1) \to (E_1\, x_{12}\, x'_1\, x'_2) \to (E_1\, x_{02}\, x'_0\, x'_2) \to \mathcal{U} \end{aligned} \tag{59}$$

Analogously to (55), the tuple $(A_0, A_1, A_2, E_0, E_1, E_2)$ can be viewed as a type-valued presheaf on the category

$$\begin{array}{ccccc} [0'] & \rightrightarrows & [1'] & \rightrightarrows & [2'] \\ \uparrow & & \uparrow & & \uparrow \\ [0] & \rightrightarrows & [1] & \rightrightarrows & [2] \end{array} \tag{60}$$

This presentation of type-theoretic structure as type families described by a *one-way* or *inverse* category is of course well-known since at least Makkai's FOLDS [Makkai, 1995]. For the theory of model categories, a similar concept has already been studied by Reedy [1974]. The theory of *Reedy fibrant diagrams* in type theory have been developed by Shulman [2015]. For the specific case of Δ_+^{op} (semisimplicial types), we have studied the relevance of the concept for homotopy type theory in earlier work [Kraus, 2015a,b].

4.2. **Two-Level Type Theory.** Voevodsky's *Homotopy Type System (HTS)* [Voevodsky, 2013] is a version of HoTT which makes it possible to make certain meta-theoretical statements in the language of type theory. *Two-Level Type Theory (2LTT)* [Capriotti, 2016, Altenkirch et al., 2016, Annenkov et al., 2019] develops the idea of HTS further and provides a more general framework. 2LTT allows us to work with infinite structures such as semisimplicial types. The same can be done in other frameworks such as Shulman's inverse diagrams [Shulman, 2015], which would be completely suitable for the work of this paper as well; and the translation of the current paper to that setting is, in fact, straightforward. For concreteness and to avoid leaving the type-theoretical setting, we nevertheless choose to work in 2LTT in the current paper.

The fundamental idea of HTS and 2LTT is very simple. They contain a new type former, written $=^{\mathsf{s}}$, which expresses *strict equality*.[5] Strict equality can be viewed as an "internalised" version of judgmental equality. In fact, one can assume that $=^{\mathsf{s}}$ satisfies equality reflection and thus $=^{\mathsf{s}}$ and $\equiv$ coincide; this is an axiom which is part of HTS, and it has been shown that this assumption can be made without destroying the good property [conservativity] of 2LTT [Annenkov et al., 2019].

However, it is not as simple as adding $=^{\mathsf{s}}$ with all the usual rules for equality. This would make $=^{\mathsf{s}}$ and $=$ coincide, which is undesirable and incompatible with HoTT. To resolve this issue, only some types of the theory will come with the equality type $=$, and these types are called *fibrant*. In other words, there are two "layers", or "levels", of types: fibrant types and not-necessarily-fibrant a.k.a. *strict* types.

Fibrant types are the ones which are of interest and which correspond to types in HoTT. Non-fibrant types can be thought of as auxiliary components which make the language more expressive, but which are of little interest themselves since they might not even exist in HoTT. If A is a fibrant type and $x, y : A$ elements, then $(x = y)$ is a fibrant type while $(x =^{\mathsf{s}} y)$ is (in general) not: the first is a perfectly fine type of HoTT, while the second is something that cannot be expressed in HoTT. In contrast, $\Sigma(x' : A).x' =^{\mathsf{s}} x$ *is* fibrant, since it is strictly isomorphic to the (fibrant) unit type; i.e. this exists in HoTT up to strict isomorphism. Similar as for equality, 2LTT has two types of natural numbers: First, the standard type $\mathbb{N}$ of fibrant natural numbers which one works with in type theory; and second, the type $\mathbb{N}^{\mathsf{s}}$ of *strict* natural numbers. Similarly, we have fibrant finite types $\mathsf{Fin}\, n$ (for $n : \mathbb{N}$) and strict finite types $\mathsf{Fin}^s\, n$ (for $n : \mathbb{N}^{\mathsf{s}}$). HTS assumes that $\mathbb{N}$ and $\mathbb{N}^{\mathsf{s}}$ coincide, which is justified by the model in simplicial sets [Kapulkin and Lumsdaine, 2012].

2LTT allows us to make the description of Section 4.1 of a semisimplicial type as a functor precise. We briefly recall the construction by Annenkov et al. [2019]. Using strict equality, we can define all the standard categorical concepts.

Definition 10 (strict categories and their theory)**.** A *strict category* is a tuple $(\mathsf{Ob}, \mathsf{Hom}, \diamond, \mathsf{assoc}, \mathsf{id}, \mathsf{idl}, \mathsf{idr})$ of standard categorical structure, where the identity laws (idl, idr, assoc) are formulated using strict equality ($=^{\mathsf{s}}$). Strict functors, strict natural transformations, and so on are analogously defined.

Given a universe $\mathcal{U}$ of fibrant types, the strict category of types in $\mathcal{U}$ and functions is denoted by $\mathcal{U}$ as well. Similarly, we can construct the category Δ_+^{op} using strict components (objects are $\mathbb{N}^{\mathsf{s}}$). We can then consider the (non-fibrant) type of functors $\Delta_+^{\mathsf{op}} \overset{s}{\Rightarrow} \mathcal{U}$, and define the (non-fibrant) structure of being *Reedy fibrant*.

[5] Like Altenkirch et al. [2016], Ahrens et al. [2020] but unlike Annenkov et al. [2019], we leave the fibrant components unannotated and annotate the strict components instead. For simplicity, we do not distinguish between *inner* and *fibrant*; this can be achieved by assuming the axiom (T3) by Annenkov et al. [2019].

We can then define the type of Reedy fibrant diagrams over Δ_+^{op}, i.e. pairs of a functor and a proof that it is Reedy fibrant, for which we write $\Delta_+^{\mathsf{op}} \overset{RF}{\Longrightarrow} \mathcal{U}$.

Given $A, E : \Delta_+^{\mathsf{op}} \overset{RF}{\Longrightarrow} \mathcal{U}$ and a natural transformation $\eta : E \to A$, one can define what it means for η to be a *Reedy fibration*, which means that E is a semisimplicial type over A as explained in the previous subsection. It is standard to denote fibrations by $\eta : E \twoheadrightarrow A$ (although attention is required when there are multiple different notions of fibrations). We refer to Annenkov et al. [2019] for the precise definition.

The type $\Delta_+^{\mathsf{op}} \overset{RF}{\Longrightarrow} \mathcal{U}$ encodes the intuitive idea of a type of "infinite tuples" $(A_0, A_1, A_2, \ldots)$. This type is (in general) not fibrant. Given a number $n : \mathbb{N}^{\mathsf{s}}$, we can take the full subcategory of Δ_+^{op} of objects $\{[0], [1], \ldots, [n]\}$. The corresponding type $\left(\Delta_+^{\leq n}\right)^{\mathsf{op}} \overset{RF}{\Longrightarrow} \mathcal{U}$ is fibrant and can be understood as the nested Σ-type of finite tuples $(A_0, A_1, \ldots, A_n)$, i.e. as a semisimplicial type of level n. It is possible to equip 2LTT with an axiom which ensures that the type $\Delta_+^{\mathsf{op}} \overset{RF}{\Longrightarrow} \mathcal{U}$ is fibrant as well, e.g. by assuming axiom (A2) of Annenkov et al. [2019], and the models of 2LTT discussed there all satisfy this axiom. The original suggestion by Voevodsky [2013], corresponding to the strongest form of (A1) by Annenkov et al. [2019], has the same effect, but unnecessarily invalidates the last of the models discussed by [Annenkov et al., 2019, Chp 2.4].

More generally, everything we have said for the type $\Delta_+^{\mathsf{op}} \overset{RF}{\Longrightarrow} \mathcal{U}$ is also the case for the type of semisimplicial types over a give semisimplicial type A.

Our presentation of 2LTT above is significantly simplified but sufficient for the current paper. For details, we refer to Annenkov et al. [2019]. Apart from axiom (A2), we assume the axioms (M1, M2, T1, T2, T3), although this is only to simplify the presentation; what we do could be done without the latter five axioms.

Section summary. By *infinite structure*, we mean what can intuitively be described as a record type with infinitely many components, or an infinitely nested Σ-type. Semisimplicial types are an example of such a structure where it is unclear whether it can be encoded in homotopy type theory.
In 2LTT, we can define the type $\Delta_+^{\mathsf{op}} \overset{RF}{\Longrightarrow} \mathcal{U}$, elements of which correspond to the intuitive idea of an infinite tuple $(A_0, A_1, A_2, \ldots)$ describing a semisimplicial type. There is a reasonable axiom which makes the type $\Delta_+^{\mathsf{op}} \overset{RF}{\Longrightarrow} \mathcal{U}$ fibrant, i.e. lets it behave like a type in homotopy type theory.

5. Higher Dimensional Categories and Internal ∞-CwF's

We now have all the tools to define *∞-categories with families*.

5.1. **Contexts and substitutions, first part: semi-Segal types.** *Segal spaces* [Rezk, 2001], also called *Rezk spaces*, are a model for higher categories. Translating Rezk's Segal condition to type theory is straightforward and has been presented previously by Capriotti [2016] and others, but note that type theory only allows us to talk about *semi*simplicial types. We briefly sketch the definition as presented by Annenkov et al. [2019].

On the strict level, and for $n : \mathbb{N}^{\mathsf{s}}$, we can define the semisimplicial set $\Delta[n]$ as the functor represented by $[n]$. This is the n-dimensional tetrahedron. A trivial application of the Yoneda lemma shows that, given a semisimplicial type A, the type of natural transformations $\Delta[n] \to A$ is strictly isomorphic to the total space of A_n (a collection of $n+1$ points in A_0, $\binom{n+1}{2}$ lines, and so on). Given another

number $k : \mathbb{N}^{\mathsf{s}}$, we can define the *horn* $\Lambda^k[n]$ as a semisimplicial set; $\Lambda^k[n]$ can be obtained from $\Delta[n]$ by removing the single cell at level n, and the cell at level $(n-1)$ which does not depend on the k-th point.

There is a canonical inclusion $\Lambda^k[n] \hookrightarrow \Delta[n]$. A semisimplicial type A satisfies the Segal condition if, for any given n and $0 < k < n$, a given $\eta : \Lambda^k[n] \to A$ can uniquely be extended to a natural transformation $\Delta[n] \to A$. Here, "uniquely extending" means that the type of extensions is contractible. A is also called *local* with respect to η.

Definition 11. A *semi-Segal type* is a semisimplicial type which satisfies the Segal condition.

Capriotti and Kraus [2017] unfold the Segal condition explicitly for small n, k (and actually fix $k \equiv 1$, which is sufficient). Given a semisimplicial type (A_0, A_1, A_2) of level 2, the Segal condition can be expressed as:

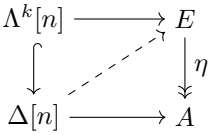
$$\begin{aligned} h_2 : (x_0\, x_1\, x_2 : A_0) \to (x_{01} : A_1\, x_0\, x_1) \to (x_{12} : A_1\, x_1\, x_2) \to \\ \mathsf{isContr}\,(\Sigma(x_{02} : A_1\, x_0\, x_2).A_2\, x_{01}\, x_{12}\, x_{02}) \end{aligned} \tag{61}$$

We also call this a *horn-filling condition*.

Capriotti and Kraus [2017] construct an explicit translation between horn-filling conditions and the structure one expects of (higher) semicategories. They show that having the pair (A_2, h_2) is equivalent to having a composition operator $\_ \circ \_ : A_1\, x_0\, x_1 \to A_1\, x_1\, x_2 \to A_1\, x_0\, x_2$. On the next level, we require that any map $\Lambda^1[3] \to A$ can uniquely be extended to $\Delta[3] \to A$, and Capriotti and Kraus [2017] show explicitly that this (together with A_3) is equivalent to stating that $\_ \circ \_$ is associative. The next level corresponds to the pentagon coherence of associativity familiar from the definition of a bicategory. This explains why semi-Segal types can be seen as ∞-semicategories.

The generalisation from semisimplicial types to maps between semisimplicial types is canonical:

Definition 12 (inner fibration)**.** An *inner fibration*, or a *semi-Segal type over a semisimplicial type*, is a Reedy fibration $\eta : E \twoheadrightarrow A$ such that, for all n, $0 < k < n$, and squares of the form shown on the right, the type of fillers (i.e. the type of the dashed map) is contractible.



(62)

By definition, a semisimplicial type A satisfies the Segal condition exactly if $A \twoheadrightarrow 1$ is an inner fibration.

5.2. **Contexts and substitutions, second part: identities.** Since a semi-Segal type possesses the structure we need for an ∞-semicategory, the natural next question is how we can extend this definition to capture ∞-categories, i.e. how we can add identities.

Similar questions have been studied outside the field of type theory long before HoTT was researched. Rourke and Sanderson [1971] have shown that a semisimplicial set with (not necessarily unique) fillers for all (not necessarily inner) horns can be extended to a simplicial set. This is however non-constructive and relies on choice. Our situation is closer to semisimplicial *spaces* with an inner horn filling condition, for which Lurie [2014] and Harpaz [2015] have suggested a simple condition which expresses the presence of suitable identities.

In type theory, the following points have to be taken into consideration:

(1) Naturally, we want to be fully constructive and not use a definition that relies on excluded middle or any version of choice.
(2) It is not satisfactory for us to only state a definition which says that an ∞-category has identities. We also want it to be suitable for defining a

type of ∞-categories. Therefore, we do *not* want a definition which allows an ∞-semicategory to be an ∞-category in more than one way. In other words, we want that "having identities" is a propositional property of ∞-semicategories.

(3) From a HoTT perspective, it is natural to want ∞-CwF's to be *univalent*, i.e. their isomorphisms should coincide with their identities. Unfortunately, this means that (in all non-trivial cases) the type of context Con cannot be a set. In particular, the initial model of a non-trivial theory (representing "syntax") cannot have decidable equality for contexts, which contradicts the expectation by Altenkirch [2018] and others. However, Eric Finster has pointed out (in a private conversation with the current author) that type-checking algorithms do not rely on context equality being decidable, and univalence might therefore be not be as problematic as it appears. This paper considers both versions by formulating univalence as a property on top of ∞-categories.

A minimalistic and simple definition of univalent higher categories based on the suggestion by Harpaz [2015] has been given by Capriotti [2016], also presented by Annenkov et al. [2019] (see also the independent earlier wiki entry by Schreiber [2012]). There, univalence is built into the definition of identities, which means it is not possible to define not-necessarily-univalent higher categories with this approach. A *direct replacement* of the full simplex category has been suggested by Kraus and Sattler [2017], which gives rise to (a replacement of) simplicial types and thereby ∞-categories. Another direct replacement was given by Kock [2006]. These approaches both work by adding an infinite tower of data.

The current paper gives a different definition of identities in ∞-semicategories. The idea is based on what is known as *dunce's hat* in topology [Zeeman, 1963], which is the simplest example of a space that is contractible but not collapsible (in type-theoretic terms: a space that is contractible but not of the form $\Sigma(a : A).a = a_0$). Concretely, we define identities to be *idempotent equivalences*. This is both minimalistic and, as we prove in this section, well-behaved. It is similar to a characterisation of identities that was independently suggested by Lai [2018], who considered $(\infty, 1)$-categories in a slightly different version of type theory.

This author expects that the definition of identities via idempotent equivalences is equivalent to both of the "elaborate" definitions using the constructions by Kock [2006] or Kraus and Sattler [2017]. If we add univalence, the equivalence with the version of Capriotti [2016] is obvious.

Our definition of an identity does not need the whole infinite structure of a semi-Segal type, but only the first four levels (A_0, A_1, A_2, A_3). Using the translation explained in Section 5.1, we phrase this subsection in the (maybe more familiar) language of a semicategory $(\mathsf{Ob}, \mathsf{Hom}, \diamond, \mathsf{assoc})$.

Definition 13 (identities via idempotent equivalences)**.** A morpism $e : \mathsf{Hom}(x, y)$ is an *equivalence* (*neutral morphism* in [Capriotti and Kraus, 2017]), written $\mathsf{iseqv}(e)$, if both composition operations

$$(\_ \diamond e) : \Pi\{z : \mathsf{Ob}\}.\mathsf{Hom}(y, z) \to \mathsf{Hom}(x, z) \tag{63}$$

$$(e \diamond \_) : \Pi\{w : \mathsf{Ob}\}.\mathsf{Hom}(w, x) \to \mathsf{Hom}(w, y) \tag{64}$$

are equivalences of types. A morphism $f : \mathsf{Hom}(x, x)$ is *idempotent* if $f \diamond f = f$. A morphism $i : \mathsf{Hom}(x, x)$ is an *identity* if it is an equivalence and idempotent. A semicategory $C \equiv (\mathsf{Ob}, \mathsf{Hom}, \diamond, \mathsf{assoc})$ has an *identity structure* if there is an identity for every object,

$$\mathsf{hasIdStruc}(C) :\equiv \Pi(x : \mathsf{Ob}).\Sigma(i : \mathsf{Hom}(x, x)).\mathsf{iseqv}(i) \times (i \diamond i = i). \tag{65}$$

Remark 14. Note that "being an equivalence" is a propositional property, while "being idempotent" and "being an identity" are data. The non-trivial result of this section is that "having an identity structure" is a propositional property.

The following is a useful and certainly expected characterisation of identities:

Lemma 15. *A morphism $i : \mathsf{Hom}(x, x)$ is an identity if and only if, for all composable morphisms $\xrightarrow{f}\xrightarrow{i}\xrightarrow{g}$, we have $i \diamond f = f$ and $g \diamond i = g$.*

Proof. Let i be an idempotent equivalence and $f : \mathsf{Hom}(w, x)$. We have $i \diamond i \diamond f = i \diamond i \diamond f$ trivially, thus by idempotence (and implicitly associativity) also $i \diamond i \diamond f = i \diamond f$, and by applying $(i \diamond \_)^{-1}$ to both sides $i \diamond f = f$. The proof of $g \diamond i = g$ is analogous.

Conversely, if i is such that both compositions are the identity, it is clearly an equivalence and idempotent. □

Corollary 16. *If $i_1, i_2 : \mathsf{Hom}(x, x)$ are both identities, then $i_1 = i_2$.*

Note that Lemma 15 does not state an equivalence of types, but only claims "functions in both directions". We have seen in Example 8 that the "naïve" characterisation causes coherence issues. In contrast, the following shows that the definition via idempotent equivalences is fully coherent:

Theorem 17. *For a given semicategory $\mathcal{C}$, the type $\mathsf{hasIdStruc}(C)$ is a proposition.*

Before proving this theorem, we formulate several very simple auxiliary statements. Given any equivalence $e : \mathsf{Hom}(x, y)$, we define $I(e) : \mathsf{Hom}(x, x)$ by

$$I(e) :\equiv (e \diamond \_)^{-1}(e). \tag{66}$$

This is how identities (modulo the translation of Section 5.1) are constructed by Harpaz [2015] and by Capriotti and Kraus [2017], where sufficiently many equivalences are assumed as given.

Lemma 18. *For any equivalence $e : \mathsf{Hom}(x, y)$, the morphism $I(e)$ is an identity.*

Proof. $I(e) \diamond f = f$ and $g \diamond I(e) = g$ are both shown in (the Agda formalisation accompanying) [Capriotti and Kraus, 2017]. The claim then follows by Lemma 15. Alternatively, we can show idempotence directly by setting $f :\equiv I(e)$ in the following calculation:

$$\begin{aligned} I(e) \diamond f &= (e \diamond \_)^{-1}\left((e \diamond \_)(I(e) \diamond f)\right) && (67)\\ &= (e \diamond \_)^{-1}\left((e \diamond I(e)) \diamond f\right) && (68)\\ &= (e \diamond \_)^{-1}\left(e \diamond f\right) && (69)\\ &= f && (70) \end{aligned}$$

Furthermore, $I(e)$ is an equivalence since equivalences satisfy *2-out-of-3* and $e \diamond I(e) = e$. □

Lemma 19. *Let $e : \mathsf{Hom}(x, x)$ be an equivalence. The type witnessing that e equals $I(e)$ is equivalent to the type witnessing that e is idempotent,*

$$(e = I(e)) \simeq (e \diamond e = e) \tag{71}$$

Proof. The equivalence is $\mathsf{ap}_{e \diamond \_}$ (i.e. applying $(e \diamond \_)$ on both sides of the equation). □

We are ready to show that, given a semicategory, there is at most one identity structure.

Proof of Theorem 17. Let us fix $x : \mathsf{Ob}$; by function extensionality, it suffices to prove that $\Sigma(i : \mathsf{Hom}(x,x)).\mathsf{iseqv}(i) \times (i \diamond i = i)$ is propositional. Assume we are given an element (i_0, p, q); then:

$$\Sigma(i : \mathsf{Hom}(x,x)).\mathsf{iseqv}(i) \times (i \diamond i = i) \tag{72}$$

$$\text{by Lemma 19} \quad \simeq \quad \Sigma(i : \mathsf{Hom}(x,x)).\mathsf{iseqv}(i) \times (i = I(i)) \tag{73}$$

$$\text{by Corollary 16 and Lemma 18} \quad \simeq \quad \Sigma(i : \mathsf{Hom}(x,x)).\mathsf{iseqv}(i) \times (i = i_0) \tag{74}$$

This type is easily seen to be contractible, with (i_0, p, refl) as centre of contraction. □

Phrased in the language of a semi-Segal type $(A_0, A_1, A_2, \ldots)$, a morphism $e : A_1\, x\, y$ is an equivalence if any horn of the form $1 \overset{e}{\leftarrow} 0 \to 2$ or $0 \to 2 \overset{e}{\leftarrow} 1$ has a contractible type of fillers. A proof that the morphism $f : A_1\, x\, x$ is idempotent is an element of $A_2\, f\, f\, f$.

Definition 20. An ∞-category is an ∞-semicategory with an identity structure.

Univalence for ∞-categories is an optional additional property which we discuss in Section 7.

5.3. **The empty context: a terminal object.** Modelling the empty context as a terminal object in the category of contexts is standard.

Definition 21 (terminal object)**.** Given an ∞-category $(A_0, A_1, A_2, \ldots)$, an object $x : A_0$ is *terminal* if, for all $y : A_0$, the type $A_1\, y\, x$ is contractible.

5.4. **Types: a presheaf.** Following Definition 1, types should be a presheaf on the category of contexts. This author is aware of two differently looking (but of course in a suitable sense equivalent) ways to define what an ∞*-presheaf* (or *prestack*) is in this setting. The first possibility is to:

(1) define the *opposite category* $\mathcal{A}^{\mathsf{op}}$;
(2) define the ∞-category $\mathcal{T}$ of types and functions;
(3) and define what an ∞*-functor* between ∞-categories is.

The second possibility, suggested and studied by Christian Sattler in unpublished work, is to consider *right fibrations* over the category of contexts. We will discuss this second approach and its advantages below in Remark 26. For now, we concentrate on the first possibility which is closer to the 1-categorical formulation discussed in Section 2.

Let us start with the last point (3). A morphism in a functor category is a natural transformation, and ∞-categories are certain functors (equipped with structure). Manually unfolded to type theory, a natural transformation between semisimplicial types of level 2, from (A_0, A_1, A_2) to (B_0, B_1, B_2) is a tuple (F_0, F_1, F_2) where:

$$F_0 : A_0 \to B_0 \tag{75}$$

$$F_1 : \{x_0\, x_1 : A_0\} \to (A_1\, x_0\, x_1) \to (B_1\, (F_0\, x_0)\, (F_0\, x_1)) \tag{76}$$

$$F_2 : \{x_0\, x_1\, x_2 : A_0\} \to \{x_{01} : A_1\, x_0\, x_1\} \to \{x_{12} : A_1\, x_1\, x_2\} \to \{x_{02} : A_1\, x_0\, x_2\} \to (A_2\, x_{01}\, x_{12}\, x_{02}) \to (B_2\, (F_1\, x_{01})\, (F_1\, x_{12})\, (F_1\, x_{02})) \tag{77}$$

Being a map between the underlying semisimplicial types already ensures that F preserves the compositionality stucture. We can make this transparent on level

2 via the translation explained in Section 4.1, under which the last component F_2 translates to:

$$F_2' : \{x_0\, x_1\, x_2 : A_0\} \to \{x_{01} : A_1\, x_0\, x_1\} \to \{x_{12} : A_1\, x_1\, x_2\} \to \\ (F_1\, x_{12}) \diamond (F_1\, x_{01}) = F_1(x_1 2 \diamond x_0 1) \tag{78}$$

∞-categories come with a proof that the underlying functors are Reedy fibrant, but the natural transformation takes care of this automatically. Thus, the only task that is left is to ensure that the functor F preserves identities, i.e. if $i : A_1\, x\, x$ is an idempotent equivalence, then $F_1\, i$ has to be an idempotent equivalence as well. Stating it in this way would not give a well-behaved notion of ∞-functor, since "being idempotent" is not a propositional property. Fortunately, idempotence is already preserved automatically by F_2, and "being an equivalence" *is* a propositional property.

Definition 22 (∞-functor)**.** An ∞-*functor* between ∞-categories is a natural transformation $F \equiv (F_0, F_1, F_2, \ldots)$ which maps identities to equivalences, i.e. for which we have:

$$\mathsf{id\text{-}preserving}(F) : (x : A_0) \to (i : A_1\, x\, x) \to \mathsf{isId}(i) \to \mathsf{iseqv}(F_1\, i) \tag{79}$$

Remark 23. A natural transformation between semi-Segal types does not automatically preserve equivalences or even identities. Let $\top$ be the terminal semi-Segal type, which is the unit type $\mathsf{1}$ on every level. Let A be the semi-Segal type defined by $A_0 :\equiv \mathsf{1}$, $A_1\, \_ :\equiv (\mathsf{2} \to \mathsf{2})$, $A_2\, f\, g\, h :\equiv (g \circ f = h)$, and A_{k+3} being constantly $\mathsf{1}$. There are three maps $\top \to A$, each of them even an inner fibration, but only a single of them preserves identities.

Given an ∞-category $A \equiv (A_0, A_1, A_2, \ldots)$, we need to define the opposite category $A^{\mathsf{op}} \equiv (A_0^{\mathsf{op}}, A_1^{\mathsf{op}}, A_2^{\mathsf{op}}, \ldots)$. There is really only one possible construction: of course we want $A_0^{\mathsf{op}} :\equiv A_0$ and $A_1^{\mathsf{op}}\, x\, y :\equiv A\, y\, x$, and after this, everything is determined; e.g. we have $A_2^{\mathsf{op}}\, f\, g\, h :\equiv A_2\, g\, f\, h$, since nothing else would type-check. The concrete combinatorial description for the representation as functors is identical to the one for simplicial sets given by Lurie [2009, Sec 1.2.1].

Finally, we need the ∞-category $\mathcal{T}$ of types and functions, starting with $\mathcal{T}_0 \equiv \mathcal{U}$, $\mathcal{T}_1\, X\, Y :\equiv (X \to Y)$, and $\mathcal{T}_2\, f\, g\, h \equiv (g \circ f = h)$. The complete construction has been given by Annenkov et al. [2019], which we briefly sketch here. Given any *strict* category C, such as the category of types and functions, one can define $\mathsf{N}_+(C) : \Delta_+^{\mathsf{op}} \overset{\mathsf{s}}{\Rightarrow} \mathcal{U}$ via the usual nerve construction which defines $\mathsf{N}_+(C)_n$ to be sequences $X_0 \to X_1 \to \ldots \to X_n$. This functor is not Reedy fibrant, but via a general Reedy fibrant replacement construction [Annenkov et al., 2019, Sec. 4], one can find a levelwise equivalent and Reedy fibrant functor $R(\mathsf{N}_+(C))$. The identity structure is obvious and simply comes from $X \xrightarrow{\mathsf{id}} X$.

Definition 24 (∞-category of small types)**.** The ∞-category $\mathcal{T}$ is given as $R(\mathsf{N}_+(\mathcal{U}))$.

5.5. **Terms: diagrams over a category of elements.** Recall from Definition 1 that terms are in 1-CwF's modelled by a functor $\mathsf{Tm} : \int_{\mathcal{C}^{\mathsf{op}}} \mathsf{Ty} \to \mathsf{Set}$. Our treatment of types has already covered several of the components that are needed in order to translate this to the ∞-categorical setting. The remaining missing part is the construction of the *category of elements*, which is what we explain in this subsection.

Let $\mathcal{A}$ be an ∞-category and $F : \mathcal{A} \to \mathcal{T}$ be an ∞-functor. As one may expect, $\int_{\mathcal{A}} F$ can be viewed as an ∞-category *over* $\mathcal{A}$: Given an n-simplex in $\mathcal{A}$ consisting of vertexes $x_0, x_1, \ldots$, lines $x_{01}, \ldots$, and so on, then we can build an n-simplex in $\int_{\mathcal{A}} F$ if we have something in $F_0\, x_0$, in $F_0\, x_1$, in $F_1\, x_{01}$, and so on, where all the new data has to "match". Our goal is to define $\int_{\mathcal{A}} F$ on all levels.

We first solve the problem for the case that A is a strict category and $F : A \to \mathcal{U}$ a strict functor with the following ad-hoc construction, which has been (partially) discussed by Annenkov et al. [2019]. We define $\mathsf{N}^\bullet_+(A, F)$ to be the functor $\Delta_+^{\mathsf{op}} \overset{s}{\Rightarrow} \mathcal{U}$ given by the nerve together with a single point: An element of $\mathsf{N}^\bullet_+(A, F)_n$ is a pair (s, p) of a chain $s : x_0 \to x_1 \to \ldots \to x_n$ in A and a point $p : F\, x_0$. There is a canonical natural transformation $\eta : \mathsf{N}^\bullet_+(A, F) \to \mathsf{N}_+(A)$ which, on each level, forgets the point. Using the Reedy fibrant replacement by [Annenkov et al., 2019, Sec 4] twice, we get a Reedy fibration between Reedy fibrant diagrams.

For the terminal ("universal") case, where A is $\mathcal{U}$ and F the identity functor, we denote the constructed Reedy fibration by $\mathcal{T}^\bullet \twoheadrightarrow \mathcal{T}$. We call $\mathcal{T}^\bullet$ the *∞-category of pointed types*. For the first few levels, its "over $\mathcal{T}$" representation (cf. 57–59) is the following, where we annotate arguments with their types for readability and happly is the function given by Univalent Foundations Program [2013, Eq. 2.9.2]:

$$\mathcal{T}^\bullet_0\,(X : \mathcal{U}) \quad :\equiv \quad X \tag{80}$$

$$\mathcal{T}^\bullet_1\,(f : X \to Y)\,(x : X)\,(y : Y) \quad :\equiv \quad (f\,x = y) \tag{81}$$

$$\mathcal{T}^\bullet_2\,(\alpha : g \circ f = h)\,(e_0 : f\,x = y)\,(e_1 : g\,y = z)\,(e_2 : h\,x = z) \quad :\equiv \quad e_2 = (\mathsf{happly}\,\alpha\,x) \cdot (\mathsf{ap}_g\,e_0) \cdot e_1 \tag{82}$$

We can now come back to the general case of an ∞-category $\mathcal{A}$ and an ∞-functor $F : \mathcal{A} \to \mathcal{T}$. Recall (from [Annenkov et al., 2019]) that Reedy fibrations are closed under pullback.

Definition 25 (∞-category of elements)**.** For an ∞-category $\mathcal{A}$ and a functor $F : \mathcal{A} \to \mathcal{T}$, the *∞-category of elements of* F, written $\int_{\mathcal{A}} F$, is defined to be the strict pullback of F along the Reedy fibration $\mathcal{T}^\bullet \twoheadrightarrow \mathcal{T}$, as shown on the right.

$$\begin{array}{ccc} \int_{\mathcal{A}} F & \dashrightarrow & \mathcal{T}^\bullet \\ {\scriptstyle \pi_1}\downarrow & \lrcorner & \downarrow \\ \mathcal{A} & \xrightarrow{F} & \mathcal{T} \end{array} \tag{83}$$

It is standard that pullbacks are closed under [left] fibrations (given a lifting problem for $\int_{\mathcal{A}} F \twoheadrightarrow \mathcal{A}$, we extend it to a lifting problem for $\mathcal{T}^\bullet \twoheadrightarrow \mathcal{T}$ and use the pullback property). The compisition of a left and and inner fibration is an inner fibration by a similar argument. Therefore, $\int_{\mathcal{A}} F$ is an ∞-category.

Remark 26 (left and right fibrations)**.** Let $\eta : E \twoheadrightarrow A$ be a Reedy fibration between semisimplicial types. η is a *left fibration* if it fulfils the condition of Definition 12 for $0 \leq k < n$ (that means that all left fibrations are inner fibrations, but only some inner fibration are left fibrations). Analogously, η is a *right fibration* if the condition of Definition 12 holds for for $0 < k \leq n$.

One can show that the map $\eta : \mathsf{N}^\bullet_+(A, F) \to \mathsf{N}_+(A)$ constructed above, and in particular $\mathcal{T}^\bullet \twoheadrightarrow \mathcal{T}$, is a left fibration. One can further show that $\mathcal{T}^\bullet \twoheadrightarrow \mathcal{T}$ is a *homotopical left fibration classifier*: For any left fibration $E \twoheadrightarrow A$, the type of tuples (g, h, q) with $g : A \to \mathcal{T}$, $h : E \to \mathcal{T}^\bullet$, and q witnessing that the resulting square commutes up to homotopy and is a homotopy pullback, is contractible.

This implies that left fibrations over A are in a suitable sense equivalent to ∞-functors $A \to \mathcal{T}$. Thus, ∞-presheaves on A correspond to right fibrations over A. If $\mathcal{C}$ is the ∞-category of contexts, we can define types to be given as a right fibration $\mathsf{Ty} \twoheadrightarrow \mathcal{C}$. Since Ty already is (the opposite of) the ∞-category of elements, terms are then simply given by a second right fibration $\mathsf{Tm} \twoheadrightarrow \mathsf{Ty}$. This formulation is due to Christian Sattler.

Note that the strict pullback (83) is also a homotopy pullback since the vertical maps are Reedy fibrations.

5.6. **Context extension.** Context extension is arguable the most involved component of a CwF in the 1-categorical setting. For what it is worth, context extension contributes the largest number of components in the type theoretic representation in Fig. 1. Even so, when generalising from CwF's to ∞-CwF's, context extension greatly benefits from the formulation of representabiliy via initiality in a category of elements (see Section 2.2. Even for ∞-categories, representability can be stated referring only to the lowest levels of the category. This means that context extension can be stated using only finitely many components. Still, the presentation of Fig. 1 does not quite work (recall that we relied on UIP in Section 2.2). Let us start by formulating representability.

Definition 27. Let $F : \mathcal{A} \to \mathcal{T}$ be an ∞-functor. A *representation* for F is a tuple (x, u, r) where $x : \mathcal{A}_0$, $u : F_0\, x$, and $r : (y : \mathcal{D}_0) \to (v : F_0\, y) \to \mathsf{isContr}\,(\Sigma(f : \mathcal{A}_1\, x\, y).F_1\, f\, u = v)$.

Applying this definition to our case of interest leads us to:

Definition 28 (context extension structure)**.** Let an ∞-category $\mathcal{C}$ be given together with ∞-functors $\mathsf{Ty} : \mathcal{C}^{\mathsf{op}} \to \mathcal{T}$ and $\mathsf{Tm} : \left(\int_{\mathcal{C}^{\mathsf{op}}} \mathsf{Ty}\right) \to \mathcal{T}$. A *context extension structure* consists, for all $\Gamma : \mathcal{C}_0$ and $A : \mathsf{Ty}_0\, \Gamma$, of the following data:

(1) an object $\Gamma \triangleright A : \mathcal{C}_0$,
(2) a morphism $\mathsf{p}_A : \mathcal{C}_1\, (\Gamma \triangleright A)\, \Gamma$,
(3) and a term $\mathsf{q}_A : \mathsf{Tm}_0((\Gamma \triangleright A), (\mathsf{Ty}_1\, \mathsf{p}_A\, A))$.

Whenever, in addition to Γ and A, we have $\Theta : \mathcal{C}_0$ and $\sigma : \mathcal{C}_1\, \Theta\, \Gamma$ and $t : \mathsf{Tm}_0(\Theta, (\mathsf{Ty}_1\, \sigma\, A))$, then we also have the following data:

(4) a morphism $(\sigma, t) : \mathcal{C}_1\, \Theta\, (\Gamma \triangleright A)$,
(5) a triangle filler $\triangleright\beta_1^{A,\sigma,t} : \mathcal{C}_2\, (\sigma, t)\, \mathsf{p}_A\, \sigma$
(6) an equality $\triangleright\beta_2^{A,\sigma,t} : \mathsf{Tm}_1\, ((\sigma, t), \mathsf{refl})\, \mathsf{q}_A = t$ over the equality we get from $\mathsf{Ty}_2\, \triangleright\, \beta_1^{A,\sigma,t}$.
(7) for any 3-tuple (τ, f, e) of the same type as $((\sigma, t), \triangleright\beta_1^{A,\sigma,t}, \triangleright\beta_2^{A,\sigma,t})$, the two tuples are equal.

Section summary. An ∞-CwF has the following components:

(1) An ∞-category $\mathcal{C}$, cf. Definition 20. This is a semisimplicial type $(\mathcal{C}_0, \mathcal{C}_1, \mathcal{C}_2, \ldots)$ such that every inner horn has a contractible type of fillers and, for every object, we have an idempotent equivalence.
(2) $\mathcal{C}$ has a terminal object, cf. Definition 21. An object x is terminal if every $\mathcal{C}_1\, y\, x$ is contractible.
(3) An ∞-functor $\mathsf{Ty} : \mathcal{C}^{\mathsf{op}} \to \mathcal{T}$. Here, $\mathcal{T}$ (cf. Definition 24) is the semisimplicial type of (small) types and functions, with $\mathcal{T}_0 \equiv \mathcal{U}$ and $\mathcal{T}_1\, X\, Y :\equiv (X \to Y)$. An ∞-functor (cf. Definition 22) is a natural transformation between semisimplicial types, i.e. a sequence $(F_0, F_1, F_2, \ldots)$, which maps identities to equivalences.
(4) A second ∞-functor $\mathsf{Ty} : \int_{\mathcal{C}^{\mathsf{op}}} \mathsf{Ty} \to \mathcal{T}$. The ∞-category of elements (cf. Definition 25) is defined by first constructing the ∞-category of pointed types.
(5) A context extension structure, cf. Definition 28. Context extension can be represented in a finite way and is essentially the same as in the UIP case.

Christian Sattler has suggested an alternative where the above functors Ty and Tm are replace by a sequence of right fibrations, $\mathsf{Tm} \twoheadrightarrow \mathsf{Ty} \twoheadrightarrow \mathcal{C}$.

6. Examples of ∞-Categories with Families

6.1. **The Syntax as a QIIT.** We will discuss later (Theorem 33) that every set-CwF in the sense of Definition 5 and Fig. 1 can be presented as an ∞-CwF. Therefore, the quotient inductive-inductive construction by Altenkirch and Kaposi [2016], discussed in Example 2, is an ∞-CwF.

The concrete construction of the ∞-CwF $\mathcal{C}$ from Fig. 1 can be described as follows. One starts by defining $\mathcal{C}_0$ to be Con and by setting $\mathcal{C}_1\,\Gamma\,\Delta$ to be $\mathsf{Sub}\,\Gamma\,\Delta$. The next level is given by $\mathcal{C}_2\,f\,g\,h :\equiv (g \diamond f = h)$, and all higher components of $\mathcal{C}$ are contractible: $\mathcal{C}_{3+n}\,\_ :\equiv \mathsf{1}$. The other parts are constructed analogously.

It is possible to define what a morphism between ∞-CwF's is, and thereby to state what it means for an ∞-CwF $\mathcal{C}$ to be *initial* – to be understood in the usual sense of *homotopy initial*, i.e. there is a contractible type of morphisms from $\mathcal{C}$ to any other given ∞-CwF.

While the syntax in Example 2 is initial among set-CwF's, it is highly unclear whether, if viewed as an ∞-CwF (Section 6.1), it is also initial among higher CwF's. Instead, we can attempt to define the initial such ∞-CwF directly in the next section.

6.2. **The Initial Model as a HIIT.** The definition of an ∞-CwF is fully algebraic in the sense of Cartmell [1986], although with infinitely many sorts, operations, and equations, and this is still true if we add components such as base types or Π-types. Clearly, semisimplicial types themselves are an (infinite) generalised algebraic theory, as seen from the presentation (51–53).

For the Segal condition, this is not as obvious, but still easy to see; in fact, we have in Section 5.1 chosen the specific formulation of the Segal condition via horn filling in order to simplify this part.

Assume we have a semisimplicial type $B \equiv (B_0, B_1, B_2, \ldots)$. We can generate the *localisation*[6] of B at inner horn inclusions, i.e. the *free* ∞*-semicategory* $A \equiv (A_0, A_1, A_2, \ldots)$ generated by B via a huge higher inductive-inductive construction. First, we want to ensure that A contains B, so we unsurprisingly start with:

$$\eta_0 : \quad B_0 \to A_0 \tag{84}$$

$$\eta_1 : \quad (x_0\,x_1 : B_0) \to (B_1\,x_0\,x_1) \to A_1(\eta_0\,x_0)(\eta_0\,x_1) \tag{85}$$

$$\eta_2 : \quad \{x_0\,x_1\,x_2 : B_0\} \to (x_{01} : B_1\,x_0\,x_1) \to (x_{12} : B_1\,x_1\,x_2) \to$$
$$(x_{02} : B_1\,x_0\,x_2) \to (B_2\,x_{01}\,x_{12}\,x_{02}) \to A_2\,(\eta_1\,x_{01})(\eta_1\,x_{12})(\eta_1\,x_{02}) \tag{86}$$

Next, each level of the Segal-condition needs to be split into four parts. For example, to state that Λ^3_1-horns have contractible types of fillers, we need (we omit implicit arguments):

$$h^e_2 : \quad (x_{01} : A_1\,x_0\,x_1) \to (x_{12} : A_1\,x_1\,x_2) \to A_1\,x_0\,x_2 \tag{87}$$

$$h^e_3 : \quad (x_{01} : A_1\,x_0\,x_1) \to (x_{12} : A_1\,x_1\,x_2) \to A_2\,x_{01}\,x_{12}\,(h^e_2\,x_{01}\,x_{12}) \tag{88}$$

$$h^u_2 : \quad (x_{01} : A_1\,x_0\,x_1) \to (x_{12} : A_1\,x_1\,x_2) \to$$
$$(f : A_1\,x_0\,x_2) \to (A_2\,x_{01}\,x_{12}\,x_{02}) \to f = (h^e_2\,x_{01}\,x_{12}) \tag{89}$$

$$h^u_3 : \quad (x_{01} : A_1\,x_0\,x_1) \to (x_{12} : A_1\,x_1\,x_2) \to$$
$$(f : A_1\,x_0\,x_2) \to (F : A_2\,x_{01}\,x_{12}\,x_{02}) \to F =_{h^u_2} (h^e_3\,x_{01}\,x_{12}) \tag{90}$$

The intuition behind the name $h^{e/u}_i$ above is as follows: i is the level, e stands for *existence* [of a filler], u stands for *uniqueness* [of the existing filler].

[6] Localisations in homotopy type theory have been studied by Rijke et al. [2017] and Christensen et al. [2018].

The other components of an ∞-CwF can be written as a GAT as well. If we assume that our host type theory has an infinite version of higher inductive-inductive types as specified by Kaposi and Kovács [2020], then this gives us the initial ∞-CwF. The semisimplicial type B can be used to specify base types, or it can be taken to be empty with base types added as part of the induction.

We do not know under which conditions this HIIT turns out to have decidable equality, but if it does, then it in particular is set-based in the sense of Section 7.1 below. This means that it would be equivalent to a set-CwF (cf. Theorem 33), namely the syntax QIIT.

6.3. **Higher Models from Strict Models and the Standard Interpretation.** We show that, from a *strict* CwF such as the standard model, we get an ∞-CwF.

Definition 29 (strict CwF)**.** In 2LTT, a *strict category with families (sCwF)* is a CwF as in Fig. 1 such that Con, Sub, Ty and Tm are all fibrant types, while all the stated equalities for contexts, substitutions, types, and terms hold up to strict equality ($=^{\mathsf{s}}$). For context extension, we need the contractibility condition (32), stated using the usual fibrant equality type ($=$).

Lemma 30. *Given a sCwF, we can construct an ∞-CwF.*

Proof sketch. The first part of the proof is given by the construction of semi-Segal types from strict categories, as described by Annenkov et al. [2019] and sketched in Section 5.4. To construct the remaining components, we first generate strict semisimplicial diagrams by taking nerves N_+ and $\mathsf{N}^\bullet_+$ as in Section 5.5.

The Reedy fibrant replacement operation R constructed by Annenkov et al. [2019] has the property that, from a strict natural transformation $\tau : X \to Y$ between strict diagrams over Δ_+^{op}, we get a strict natural transformation $R(\tau) : R(X) \to R(Y)$ between the respective Reedy fibrant replacements, i.e. semisimplicial types. This follows from [Annenkov et al., 2019, Cor 4.28], applied on the diagram on the right.

$$\begin{array}{ccc} X & \xrightarrow{\quad\tau\quad} & Y \\ \downarrow & & \downarrow \\ R(X) & \xrightarrow{\quad R(\tau)\quad} & R(Y) \\ \downarrow\!\downarrow & & \downarrow\!\downarrow \\ 1 & \longrightarrow & 1 \end{array}$$

Simply fibrantly replacing everything does not fully work for the ∞-functor Tm since the fibrant replacement does not commute with the required strict pullback, but switching to right fibrations and back (as discussed in Remark 26) resolves the issue. Context extension can be checked manually, since it only concerns the lowest levels. □

Applying Lemma 30 on the 1-categorical standard model (which is a strict CwF, see Example 3) shows that the standard model is also an ∞-CwF.

6.4. **Slicing in ∞-CwF's.** As a final example, let us show that the slice construction analogous to Example 4 works for ∞-CwF's. Slicing of ∞-categories in type theory works in essentially the same way as slicing in Kan simplical sets or similar settings [Lurie, 2009].

Given an ∞-category $\mathcal{C}$ with an object $\Gamma : \mathcal{C}_0$, we want to construct the *slice ∞-category* $(\mathcal{C}/\Gamma)$. Recall that elements of $\mathcal{C}_n$ can, by Yoneda, be seen as n-simplexes $\Delta[n] \overset{\mathsf{s}}{\Rightarrow} \mathcal{C}$. We define $(\mathcal{C}/\Gamma)_n$ to be the type of $(n{+}1)$-simplexes where the last vertex strictly equals Γ; in other words, $(\mathcal{C}/\Gamma)_n$ is defined to be $\mathcal{C}_{n+1}$ with a condition involving a strict equality. This type is fibrant nevertheless: $(\mathcal{C}/\Gamma)_n$ is (strictly isomorphic to) a nested Σ-type of $(n{+}2)$ vertexes including the last vertex $x_{n+1} : \mathcal{C}_0$, lines, triangle fillers, ..., and the strict equality $e : x_{n+1} = \Gamma$. But the type of pairs (x_{n+1}, e) is strictly isomorphic to the unit type and therefore fibrant.

The morphism part of $(\mathcal{C}/\Gamma)$ is given as those morphisms in $\mathcal{C}_{n+1}$ which do not touch the last vertex. Thinking of $\mathcal{C}_{n+1}$ as simplexes and face maps as projections as in (56), face maps are those projections which do not remove the last vertex x_{n+1}.

It is easy to see that $(\mathcal{C}/\Gamma)$ is a semi-Segal type. If i is an identity in $\mathcal{C}$ and $f : \mathcal{C}_1\,\Theta\,\Gamma$ an object of $(\mathcal{C}/\Gamma)$, then the identity on this object is given by the cell in $\mathcal{C}_2\,f\,i\,f$ of Lemma 15. The functors Ty and Tm are not affected by the slicing operation.

Section summary. In Section 3, we have seen several examples that show why the notion of a CwF implemented in type theory without UIP is unsatisfactory. In the current section, we have seen that all these examples work as expected as soon as we move to ∞-CwF's.

7. Variations: Set-Based, Univalent, and Finite-Dimensional Models

We introduce three possible properties that an ∞-CwF can have. Each of these three properties is a proposition.

7.1. Set-Based ∞-CwF's.

Recall that a type is said to be a *set* if it satisfies UIP.

Definition 31 (set-based ∞-CwF)**.** An ∞-CwF is *set-based* if, for all Γ, Δ, A, each of the types $\mathcal{C}_0$, $\mathcal{C}_1\,\Gamma\,\Delta$, $\mathsf{Ty}_0\,\Gamma$, and $\mathsf{Tm}_0(\Gamma, A)$ is a set.

This can in more detail be described as:

Lemma 32. *In a set-based ∞-Cwf,* all *of the following apply:*

- *By definition, the types and type families $\mathcal{C}_0$, $\mathcal{C}_1$, Ty_0, and Tm_0 are all set-truncated;*
- *it follows that $\mathcal{C}_2$, Ty_1, Tm_1 are families of propositions;*
- *and all remaining occurring components ($\mathcal{C}_{3+i}$, Ty_{2+i}, Tm_{2+i}) are families of contractible types.*

Proof. For $\mathcal{C}$, this follows from the Segal condition in the same way as in [Kraus, 2015b, Lem 8.9.3]. For Ty and Tm, this is follows from the fact that $\mathcal{T}$, $\mathcal{T}^\bullet$ have the corresponding property. □

We can now conclude:

Theorem 33. *Set-based ∞-CwF's are equivalent to the set-CwF's of Definition 5.*

Proof. The above lemma shows that almost all components of a set-based ∞-CwF are trivial. Together with the translation by Capriotti and Kraus [2017], this proves that the data of the semi-Segal type is equivalent to the representation of a semi-category in Fig. 1. All remaining parts are easy to check. □

Note that Theorem 33 also implies that the formalised syntax using UIP, for example as given by Altenkirch and Kaposi [2016], can be formulated as an ∞-CwF (cf. Section 6.1).

Example 34. The syntax by Altenkirch and Kaposi [2016] is an example for a set-based ∞-CwF, and a non-example is the standard model as in Example 3.

7.2. **Univalent ∞-CwF's.** Since an identity in an ∞-CwF $\mathcal{C}$ is by definition an idempotent equivalence, we have, for all $\Gamma, \Delta : \mathcal{C}_0$, an obvious map

$$(\Gamma = \Delta) \to \Sigma(e : \mathcal{C}_1\,\Gamma\,\Delta).\mathsf{iseqv}(e). \tag{91}$$

The consequent formulation of univalence is standard:

Definition 35 (univalent ∞-CwF)**.** An ∞-category is *univalent* if the function (91) is an equivalence.

This allows us to connect our version of higher categories to the complete semi-Segal types suggested by Annenkov et al. [2019] and others. Recall that a semi-Segal type is *complete* if, for every object Γ, the type $\Sigma(\Delta : \mathcal{C}_0).\Sigma(e : \mathcal{C}_1\,\Gamma\,\Delta).\mathsf{iseqv}(e)$ is contractible.

Lemma 36. *A semi-Segal type A is a univalent ∞-category if and only if it is* complete *in the sense of Annenkov et al. [2019].*

Proof. From a complete semi-Segal type, we can construct an explicit identity structure as in (66) using Lemma 18. Given the identity structures, univalence and completeness both say that the total space $\Sigma(e : \mathcal{C}_1\,\Gamma\,\Delta).\mathsf{iseqv}(e)$ is contractible, making them equivalent. □

Theorem 37. *In an ∞-CwF that is both set-based and univalent, the identities are the only auto-equivalences.*

Proof. By univalence, equivalences and equalities coincide; and by assumption, refl is the only equality of type $\Gamma = \Gamma$. □

Theorem 37 implies that only trivial models can be set-based and univalent at the same time, and the representations of the syntax e.g. by Altenkirch and Kaposi [2016] is definitely not univalent. Already the simple context $(b : A, c : A)$ has the non-trivial auto-equivalence which swaps b and c.

Example 38. The standard model in Example 3 is a univalent ∞-CwF, while the syntax by Altenkirch and Kaposi [2016] is not univalent.

7.3. **Finite-Dimensional Models.** We have seen that set-based CwF's correspond to 1-categories. From the point of view of homotopy type theory, it is somewhat unusual that objects and morphisms ($\mathcal{C}_0$ and $\mathcal{C}_1$) are restricted to the same truncation level (recall that univalent 1-categories have 1-truncated types of objects). Thus, we say that an ∞-CwF are *of dimension* 1 if $\mathcal{C}_0$ is a 1-type and $\mathcal{C}_1$ is a family of sets. We also simply say that $\mathcal{C}$ is a 1-CwF. The canonical generalisation is the following:

Definition 39 (finite-dimensional ∞-CwF *or* n-CwF)**.** Let $n : \mathbb{N}^{\mathsf{s}}$ be a number. An ∞-CwF is of dimension n if $\mathcal{C}_0$ is an n-type, and $\mathcal{C}_1$, Ty_0, as well as Tm_0 are families of $(n-1)$-types. In this case, we say that $\mathcal{C}$ is an *n-CwF*.

Note that the condition on Ty_0 amounts to requiring that $\mathsf{Ty} : \mathcal{C}^{\mathsf{op}} \overset{RF}{\Longrightarrow} \mathcal{U}$ factors through $\mathcal{U}^{\leq n}$, the semi-Segal type of n-types. The analogous observation holds for Tm_0.

Lemma 40. *If an ∞-Cwf is of dimension n, then we have:*

- *For all $i : \mathbb{N}^{\mathsf{s}}$, the type family $\mathcal{C}_{n+i}$ is $(n-i)$-truncated (or contractible, if $n - i \leq -2$);*
- *Ty_0 is an $(n-1)$-truncated family and, for all $i \geq 1$, Ty_{n+i} is $(n-i)$-truncated;*
- *and finally, Tm_0 is an $(n-1)$-truncated family, while Tm_{n+i} are $(n-i)$-truncated.*

Proof. Analogous to Lemma 32. □

Remark 41. The awkward special cases for Ty_0 and Tm_0 stem from the fact that we are working with functors into $\mathcal{T}$. If we work with right fibrations (see Remark 26) instead, the truncation conditions are more regular.

As in the set-based case, almost all components of an n-dimensional CwF are contractible families and can be omitted since they do not carry any information. The only exception to this is $\mathcal{C}_{n+2}$, which is a contractible family but which is required to formulate the Segal condition on the level that affects $\mathcal{C}_{n+1}$, i.e. non-contractible components depends on it.[7] Each of the finitely many components can be formulated purely in the inner/fibrant fragment of 2LTT. This observation implies:

Theorem 42. *In "plain" homotopy type theory as developed by the Univalent Foundations Program [2013], and for an externally fixed natural number n, we can define a type of n-CwF's. In 2LTT (even without the axiom that $\mathbb{N}^{\mathsf{s}}$ is cofibrant), for a number $n : \mathbb{N}^{\mathsf{s}}$, the type of n-CwF's is fibrant.*

Proof. This follows from a result by Shulman [2015, Lem 11.8] and the analogous formulation by Annenkov et al. [2019, Thm 4.8], respectively. □

Example 43. Since an n-CwF is the more general version of a set-based ∞-CwF, what we said in Example 34 can be transferred. The standard model from Example 3 is not an n-CwF for any $n : \mathbb{N}^{\mathsf{s}}$, but by replacing the universe $\mathcal{U}$ in its construction by the type $\Sigma(X : \mathcal{U}).\mathsf{is}\text{-}n\text{-}\mathsf{type}(X)$, we get the n-CwF of n-types.

Section summary. We have identified three possible additional properties that an ∞-CwF can have. It can be *univalent*, which is natural in homotopy type theory; an obvious example is the standard model. An ∞-CwF can further be *finite-dimensional*. This gives rise to the notion of an *n-CwF* which can be expressed in homotopy type theory and vastly generalises the internal CwF's that the current literature considers. In particular, this notion is general enough to allow non-trivial versions of the standard model, albeit restricted to n-types. *Set-based ∞-CwF's* are a very weak special case and even less general than 1-CwF's, but a notion that captures many developments that have been done assuming UIP.

8. Open Problems and Future Directions

We have demonstrated that higher-dimensional categories lead to a notion of model of type theory that is very well-behaved and works in situations where 1-categorical models are unsuitable. Our work inspires numerous new questions. First of all, it seems natural to equip the definition of an ∞-CwF with additional components such as Π-types, Σ-types, universes, or simple base types. Developing a notion of *∞-natural transformation*, it is easy to define what a *morphism* between ∞-categories is; but the natural expectation would be that [small] ∞-CwF's form a [large] $(\infty, 2)$-category, and it is much less clear how this can be formulated. The intermediate goal would be to show that ∞-CwF's form a large ∞-CwF.

A very important question is whether the syntax (defined as a quotient inductive-inductive type) is initial as an ∞-CwF (after adding base types and other type

[7] This is identical to the situation encountered by Capriotti and Kraus [2017] who show that a univalent 1-category corresponds to a semi-Segal type (A_0, A_1, A_2, A_3) with structure, where it is necessary to include A_3 even though it is always contractible (unless, of course, one wants to replace it by another special case).

formers). Phrased differently, we can ask whether the initial ∞-CwF has decidable equality. This is very closely related to the open problem whether HoTT can "eat" itself [Shulman, 2014]. Our conjecture is that HoTT cannot eat itself, but that 2LTT can eat itself. Of course, the intermediate goal that this paper is working towards is the statement that 2LTT can eat HoTT. Even an even much weaker question, namely whether the initial ∞-CwF has trivial fundamental group, would be very interesting; but even this seemingly much simpler problem appears to be highly non-trivial, and similar results for seemingly simpler situations [Kraus and Altenkirch, 2018, Kraus and von Raumer, 2020] suggest that a solution will require new techniques.

If it turns out that the (set-truncated) syntax is indeed the initial ∞-CwF it would show that, if we give ourselves a way to talk about semisimplicial types, then HoTT can eat itself. The other direction seems much more in reach: If HoTT can eat itself, then semisimplicial types can be defined. This has already been conjectured by Shulman [2014] but not been made precise.

Another question is whether the higher inductive-inductive definition described in Section 6.2 really requires what is described as *HIITs with infinitely many components*. We would expect that such HIITs are not required and ordinary HIITs, albeit indexed over $\mathbb{N}^{\mathbb{S}}$, suffice to encode the desired initial model. It also would be interesting to formulate and study *higher categories with attributes* (which one would expect to be equivalent to ∞-CwF's), the more general *higher comprehension categories*, and other internal ∞-categorical formulations of models that have been considered in 1-category theory.

The connection to *directed type theory* [Licata and Harper, 2011, Riehl and Shulman, 2017, Nuyts, 2015, North, 2019] is intriguing. Future directed type theories may allows us to develop ∞-category theory in a synthetic way, and the relationship to the half-synthetic approach of this paper will be interesting to see.

Finally, 2LTT appears to receive more and more support by proof assistants. At the time of writing, it seems likely that future versions of Agda will be make it possible to formalise results such as those of the current paper.

Acknowledgements. This paper has benefited from many discussions that I had with various people during the last couple of years.

I am particularly grateful for the numerous discussions with Christian Sattler and Paolo Capriotti. I have learned many ∞-categorical ideas from Christian and Paolo, and their support allowed me to understand much more of the higher categorical literature than I would have managed to understand on my own. Moreover, Paolo pointed me to the more abstract and alternative definitions of CwF's and representability, which proved to be very useful for the development in this paper. I also thank Christian for very helpful comments on an earlier draft of this paper.

Additional special thanks go to Ambrus Kaposi and Thorsten Altenkirch for explaining me their intrinsically well-typed syntax until the main points sunk in.

Furthermore, I thank the type theory communities in Birmingham, Budapest, and Nottingham for helpful discussions.

References

Andreas Abel, Joakim Öhman, and Andrea Vezzosi. Decidability of conversion for type theory in type theory. *Proceedings of the ACM on programming languages*, 2(POPL):23, 2017.

Benedikt Ahrens, Paolo Capriotti, and Régis Spadotti. Non-wellfounded trees in homotopy type theory. In *Typed Lambda Calculi and Applications (TLCA 2015)*, volume 38 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages

17–30. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2015a. ISBN 978-3-939897-87-3. doi: http://dx.doi.org/10.4230/LIPIcs.TLCA.2015.17. URL http://drops.dagstuhl.de/opus/volltexte/2015/5152.

Benedikt Ahrens, Krzysztof Kapulkin, and Michael Shulman. Univalent categories and the Rezk completion. *Mathematical Structures in Computer Science (MSCS)*, pages 1–30, Jan 2015b. ISSN 1469-8072. doi: 10.1017/S0960129514000486. URL http://journals.cambridge.org/article_S0960129514000486.

Benedikt Ahrens, Peter LeFanu Lumsdaine, and Vladimir Voevodsky. Categorical structures for type theory in univalent foundations. In Valentin Goranko and Mads Dam, editors, *26th EACSL Annual Conference on Computer Science Logic (CSL 2017)*, volume 82 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 8:1–8:16, Dagstuhl, Germany, 2017. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. ISBN 978-3-95977-045-3. doi: 10.4230/LIPIcs.CSL.2017.8. URL http://drops.dagstuhl.de/opus/volltexte/2017/7696.

Benedikt Ahrens, Peter LeFanu Lumsdaine, and Vladimir Voevodsky. Categorical structures for type theory in univalent foundations. *Logical Methods in Computer Science*, Volume 14, Issue 3, September 2018. doi: 10.23638/LMCS-14(3:18)2018. URL https://lmcs.episciences.org/4814.

Benedikt Ahrens, Paige Randall North, Michael Shulman, and Dimitris Tsementzis. A higher structure identity principle. In *Proceedings of the 35th Annual ACM/IEEE Symposium on Logic in Computer Science*, LICS '20, pages 53–66, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450371049. doi: 10.1145/3373718.3394755. URL https://doi.org/10.1145/3373718.3394755.

Thorsten Altenkirch. Towards the syntax and semantics of higher dimensional type theory, 2018. Talk abstract for HoTT/UF, Oxford.

Thorsten Altenkirch and Ambrus Kaposi. Type theory in type theory using quotient inductive types. *SIGPLAN Not.*, 51(1):18–29, January 2016. ISSN 0362-1340. doi: 10.1145/2914770.2837638. URL http://doi.acm.org/10.1145/2914770.2837638.

Thorsten Altenkirch, Paolo Capriotti, and Nicolai Kraus. Extending homotopy type theory with strict equality. In *25th EACSL Annual Conference on Computer Science Logic (CSL 2016)*, volume 62 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 21:1–21:17, Dagstuhl, Germany, 2016. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. ISBN 978-3-95977-022-4. doi: http://dx.doi.org/10.4230/LIPIcs.CSL.2016.21. URL http://drops.dagstuhl.de/opus/volltexte/2016/6561.

Thorsten Altenkirch, Paolo Capriotti, Gabe Dijkstra, Nicolai Kraus, and Fredrik Nordvall Forsberg. Quotient inductive-inductive types. In Christel Baier and Ugo Dal Lago, editors, *Foundations of Software Science and Computation Structures (FoSSaCS 2018)*, pages 293–310. Springer International Publishing, 2018. doi: https://doi.org/10.1007/978-3-319-89366-2_16.

Danil Annenkov, Paolo Capriotti, Nicolai Kraus, and Christian Sattler. Two-level type theory and applications. *ArXiv*, 2019. Available online at https://arxiv.org/abs/1705.03307.

Steve Awodey. Natural models of homotopy type theory. *Mathematical Structures in Computer Science*, 28(2):241–286, 2018. doi: 10.1017/S0960129516000268.

Steve Awodey and Michael A. Warren. Homotopy theoretic models of identity types. *Mathematical Proceedings of the Cambridge Philosophical Society*, 146:45–55, 2009. ISSN 1469-8064. doi: 10.1017/S0305004108001783. URL http://journals.cambridge.org/article_S0305004108001783.

Bruno Barras, Thierry Coquand, and Simon Huber. A generalization of the takeuti-gandy interpretation. *Mathematical Structures in Computer Science*, 25(5):1071–1099, 2015. doi: 10.1017/S0960129514000504.

Julia E Bergner. A survey of (infinity, 1)-categories. In *Towards higher categories*, pages 69–83. Springer, 2010.

Jean-Philippe Bernardy, Thierry Coquand, and Guilhem Moulin. A presheaf model of parametric type theory. *Electronic Notes in Theoretical Computer Science*, 319:67 – 82, 2015. ISSN 1571-0661. doi: https://doi.org/10.1016/j.entcs.2015.12.006. URL `http://www.sciencedirect.com/science/article/pii/S1571066115000730`. The 31st Conference on the Mathematical Foundations of Programming Semantics (MFPS XXXI).

Simon Boulier. *Extending Type Theory with Syntactic Models*. PhD thesis, École des Mines de Nantes, Nantes, France, 2018.

Guillaume Brunerie and Menno de Boer. A formalization of the initiality conjecture in agda, 2019. Talk given at the HoTT 2019 conference, slides available at `https://guillaumebrunerie.github.io/pdf/initiality.pdf`.

Ulrik Buchholtz. Formalizing type theory in type theory using nominal techniques, 2017. Talk at HoTT/UF, Oxford.

Paolo Capriotti. *Models of Type Theory with Strict Equality*. PhD thesis, School of Computer Science, University of Nottingham, Nottingham, UK, 2016. Available online at `https://arxiv.org/abs/1702.04912`.

Paolo Capriotti and Nicolai Kraus. Univalent higher categories via complete semi-segal types. *Proceedings of the ACM on Programming Languages*, 2(POPL): 44:1–44:29, December 2017. ISSN 2475-1421. doi: 10.1145/3158132. URL `http://doi.acm.org/10.1145/3158132`.

Paolo Capriotti and Christian Sattler. Higher categories of algebras for higher inductive definitions. 2020. Abstract for the conference TYPES'20.

John Cartmell. *Generalised algebraic theories and contextual categories*. PhD thesis, Oxford, 1978.

John Cartmell. Generalised algebraic theories and contextual categories. *Annals of pure and applied logic*, 32:209–243, 1986.

James Chapman. Type theory should eat itself. *Electronic Notes in Theoretical Computer Science*, 228:21–36, 2009.

Daniel Christensen, Morgan Opie, Egbert Rijke, and Luis Scoccola. Localization in homotopy type theory. *arXiv preprint arXiv:1807.04155*, 2018.

Thierry Coquand, Simon Huber, and Christian Sattler. Homotopy Canonicity for Cubical Type Theory. In Herman Geuvers, editor, *4th International Conference on Formal Structures for Computation and Deduction (FSCD 2019)*, volume 131 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 11:1–11:23, Dagstuhl, Germany, 2019. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. ISBN 978-3-95977-107-8. doi: 10.4230/LIPIcs.FSCD.2019.11. URL `http://drops.dagstuhl.de/opus/volltexte/2019/10518`.

Nils Anders Danielsson. A formalisation of a dependently typed language as an inductive-recursive family. In *International Workshop on Types for Proofs and Programs*, pages 93–109. Springer, 2006.

Peter Dybjer. Internal type theory. In Stefano Berardi and Mario Coppo, editors, *Types for Proofs and Programs (TYPES)*, volume 1158 of *Lecture Notes in Computer Science*, pages 120–134. Springer-Verlag, 1995. ISBN 978-3-540-70722-6. doi: 10.1007/3-540-61780-9_66.

Martín Hötzel Escardó and Chuangjie Xu. Autophagia – type theory eating itself?, 2014. Agda project, available at `https://www.cs.bham.ac.uk/~mhe/TT-perhaps-eating-itself/TT-perhaps-eating-itself.html`.

Hakon Gylterud, Peter LeFanu Lumsdaine, and Erik Palmgren. Formalising semantics of dependent type theory in dependent type theory, 2015. Talk at DMV, Hamburg.

Yonatan Harpaz. Quasi-unital ∞–categories. *Algebraic & Geometric Topology*, 15 (4):2303–2381, 2015.

Michael Hedberg. A coherence theorem for Martin-Löf's type theory. *Journal of Functional Programming*, 8(4):413–436, 1998.

Hugo Herbelin. A dependently-typed construction of semi-simplicial types. *Mathematical Structures in Computer Science*, pages 1–16, Mar 2015. ISSN 1469-8072. doi: 10.1017/S0960129514000528. URL `http://journals.cambridge.org/article_S0960129514000528`.

Martin Hofmann and Thomas Streicher. The groupoid interpretation of type theory. In *In Venice Festschrift*, pages 83–111. Oxford University Press, 1996. URL `www.mathematik.tu-darmstadt.de/~streicher/venedig.ps.gz`.

Ambrus Kaposi. *Type theory in a type theory with quotient inductive types*. PhD thesis, School of Computer Science, University of Nottingham, Nottingham, UK, 2016. Available online at `http://eprints.nottingham.ac.uk/41385/1/th.pdf`.

Ambrus Kaposi and Thorsten Altenkirch. Normalisation by evaluation for type theory, in type theory. *Logical Methods in Computer Science*, 13, 2017.

Ambrus Kaposi and András Kovács. A syntax for higher inductive-inductive types. In *3rd International Conference on Formal Structures for Computation and Deduction (FSCD 2018)*, volume 108 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 20:1–20:18, Dagstuhl, Germany, 2018. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. ISBN 978-3-95977-077-4. doi: 10.4230/LIPIcs.FSCD.2018.20. URL `http://drops.dagstuhl.de/opus/volltexte/2018/9190`.

Ambrus Kaposi and András Kovács. Signatures and induction principles for higher inductive-inductive types. *Logical Methods in Computer Science*, Volume 16, Issue 1, February 2020. doi: 10.23638/LMCS-16(1:10)2020. URL `https://lmcs.episciences.org/6100`.

Ambrus Kaposi, Simon Huber, and Christian Sattler. Gluing for type theory. In Herman Geuvers, editor, *4th International Conference on Formal Structures for Computation and Deduction (FSCD 2019)*, volume 131 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 25:1–25:19, Dagstuhl, Germany, 2019a. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. ISBN 978-3-95977-107-8. doi: 10.4230/LIPIcs.FSCD.2019.25. URL `http://drops.dagstuhl.de/opus/volltexte/2019/10532`.

Ambrus Kaposi, András Kovács, and Nicolai Kraus. Shallow embedding of type theory is morally correct. In *Mathematics of Program Construction (MPC'19)*, pages 329–365, 2019b. ISBN 978-3-030-33636-3. Available online at `https://arxiv.org/abs/1907.07562`.

Chris Kapulkin and Peter LeFanu Lumsdaine. The simplicial model of univalent foundations (after voevodsky). *ArXiv e-prints*, November 2012. To appear in the Journal of the European Mathematical Society.

Joachim Kock. Weak identity arrows in higher categories. *International Mathematics Research Papers*, 2006, 2006.

Nicolai Kraus. The general universal property of the propositional truncation. In Hugo Herbelin, Pierre Letouzey, and Matthieu Sozeau, editors, *20th International Conference on Types for Proofs and Programs (TYPES 2014)*, volume 39 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 111–145, Dagstuhl,

Germany, 2015a. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. ISBN 978-3-939897-88-0. doi: http://dx.doi.org/10.4230/LIPIcs.TYPES.2014.111. URL http://drops.dagstuhl.de/opus/volltexte/2015/5494.

Nicolai Kraus. *Truncation Levels in Homotopy Type Theory*. PhD thesis, School of Computer Science, University of Nottingham, Nottingham, UK, 2015b. Available online at http://eprints.nottingham.ac.uk/28986/.

Nicolai Kraus and Thorsten Altenkirch. Free higher groups in homotopy type theory. In *Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science*, LICS '18, pages 599–608, New York, NY, USA, 2018. ACM. ISBN 978-1-4503-5583-4. doi: 10.1145/3209108.3209183. URL http://doi.acm.org/10.1145/3209108.3209183.

Nicolai Kraus and Christian Sattler. Higher homotopies in a hierarchy of univalent universes. *ACM Transactions on Computational Logic (TOCL)*, 16(2):18:1–18:12, April 2015.

Nicolai Kraus and Christian Sattler. Space-valued diagrams, type-theoretically (extended abstract). *ArXiv e-prints*, 2017. Available online at https://arxiv.org/abs/1704.04543.

Nicolai Kraus and Jakob von Raumer. Coherence via well-foundedness: Taming set-quotients in homotopy type theory. In *Symposium on Logic in Computer Science (LICS 2020)*, pages 662–675, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450371049. doi: 10.1145/3373718.3394800. URL https://doi.org/10.1145/3373718.3394800.

Luhang Lai. $(\infty, 1)$-categories in infinitary HoTT. Unpublished note, 2018.

Daniel R. Licata and Robert Harper. 2-dimensional directed type theory. *Electronic Notes in Theoretical Computer Science*, 276:263 – 289, 2011. ISSN 1571-0661. doi: http://dx.doi.org/10.1016/j.entcs.2011.09.026. URL http://www.sciencedirect.com/science/article/pii/S1571066111001174. Twenty-seventh Conference on the Mathematical Foundations of Programming Semantics (MFPS XXVII).

Peter LeFanu Lumsdaine and Anders Mörtberg. Formalising the initiality conjecture in coq, 2018. Talk given at the Göteborg-Stockholm Joint Type Theory Seminar, slides available at http://peterlefanulumsdaine.com/research/Lumsdaine-2018-Goteborg-Initiality.pdf.

Jacob Lurie. *Higher Topos Theory*. Annals of Mathematics Studies. Princeton University Press, Princeton, 2009. Also avaialabe online at http://arxiv.org/abs/math/0608040.

Jacob Lurie. Higher algebra. Available at http://www.math.harvard.edu/~lurie/, Sep 2014.

Michael Makkai. First order logic with dependent sorts, with applications to category theory. Available at http://www.math.mcgill.ca/makkai/folds/, 1995.

Hoang Kim Nguyen and Taichi Uemura. ∞-type theories. 2020. Abstract presented at the online workshop HoTT/UF'20.

Paige Randall North. Towards a directed homotopy type theory. *Electronic Notes in Theoretical Computer Science*, 347:223 – 239, 2019. ISSN 1571-0661. doi: https://doi.org/10.1016/j.entcs.2019.09.012. URL http://www.sciencedirect.com/science/article/pii/S1571066119301288. Proceedings of the Thirty-Fifth Conference on the Mathematical Foundations of Programming Semantics.

Andreas Nuyts. Towards a directed homotopy type theory based on 4 kinds of variance. 2015. Master thesis, available online at https://anuyts.github.io/.

Erik Palmgren. From type theory to setoids and back. *ArXiv*, 2019. Available online at https://arxiv.org/abs/1909.01414.

Fedor Part and Zhaohui Luo. Semi-simplicial types in logic-enriched homotopy type theory. *ArXiv*, 2015. Available online at `http://arxiv.org/abs/1506.04998`.

Chris Reedy. Homotopy theory of model categories. 1974. Unpublished, date estimated, available at `http://www-math.mit.edu/~psh/`.

Charles Rezk. A model for the homotopy theory of homotopy theory. *Transactions of the American Mathematical Society*, 353(3):973–1007, 2001.

Emily Riehl and Michael Shulman. A type theory for synthetic ∞-categories. *Higher Structures*, 1(1), 2017. URL `https://journals.mq.edu.au/index.php/higher_structures/article/view/36`.

Egbert Rijke, Michael Shulman, and Bas Spitters. Modalities in homotopy type theory. *arXiv preprint arXiv:1706.07526*, 2017.

Colin Patrick Rourke and Brian Joseph Sanderson. Δ-sets I: Homotopy theory. *The Quarterly Journal of Mathematics*, 22(3):321–338, 1971.

Urs Schreiber. Category object in an (infinity,1)-category, revision 20, November 2012. nLab entry, `https://ncatlab.org/nlab/revision/category+object+in+an+%28infinity%2C1%29-category/20`; newest version available at `https://ncatlab.org/nlab/show/category+object+in+an+%28infinity%2C1%29-category`.

Michael Shulman. Homotopy type theory should eat itself (but so far, it's too big to swallow), 2014. Blog post, `homotopytypetheory.org/2014/03/03/hott-should-eat-itself`.

Michael Shulman. Univalence for inverse diagrams and homotopy canonicity. *Mathematical Structures in Computer Science*, pages 1–75, Jan 2015. ISSN 1469-8072. doi: 10.1017/S0960129514000565. URL `http://journals.cambridge.org/article_S0960129514000565`.

Matthieu Sozeau and Nicolas Tabareau. Internalization of the groupoid interpretation of type theory. In *TYPES 2014*, 2014.

Thomas Streicher. Investigations into intensional type theory, 1993. Habilitationsschrift, Ludwig-Maximilians-Universität München.

The Univalent Foundations Program. *Homotopy Type Theory: Univalent Foundations of Mathematics*. `http://homotopytypetheory.org/book/`, Institute for Advanced Study, 2013.

Vladimir Voevodsky. A simple type system with two identity types, 2013. Unpublished note.

Vladimir Voevodsky, Peter LeFanu Lumsdaine, et al. Semi-simplicial types, 2013. Discussion preserved on the nLab, `https://ncatlab.org/homotopytypetheory/show/semi-simplicial+types`.

EC Zeeman. On the dunce hat. *Topology*, 2(4):341–358, 1963.