Internal ∞-Categorical Models of Dependent Type Theory

Towards 2LTT Eating HoTT

Nicolai Kraus
University of Nottingham

***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 (set-level) 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, the paper explains the shortcomings of 1-categories in type theory without UIP as well as the difficulties of and approaches to internal higher-dimensional categories.

I. 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 [5] and others, parametric models of type theory (Bernady et al. [6]), or an implementation of the groupoid interpretation (Sozeau and Tabareau [7]) by Hofmann and Streicher [8].

The author was 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)

978-1-6654-4895-6/21/$31.00 ©2021 IEEE

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. [9]), which refers to formalising the (intended) *initial* model of type theory.[1]

Altenkirch and Kaposi [3] 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 [16], inspired by Danielsson [17] and others. Other recent work on the same goal with various techniques includes the studies by Escardó and Xu [18], the PhD thesis by Kaposi [19], the work by Gylterud, Lumsdaine, and Palmgren [20], and the presentation by Buchholtz [21].

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*, often just referred to as *CwF's* [22] (see also formulation by Awodey [23]). 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. [24], [25]. A category with families consists of a category with a terminal object, a presheaf of families on it, and a context extension operation. The objects of the category are usually denoted by Con ("contexts") and the morphisms by Sub ("substitutions" or "context morphisms"). Many presentations further split the presheaf into two functors Ty and Tm, with the idea being that Ty gives the types in a context and Tm the terms of a type. The terminal object represents the empty context.

CwF's can easily be presented as a *generalised algebraic theory* (GAT) as introduced by Cartmell [26], [27]. A GAT consists of *sorts*, *operations*, and *equations*. In the case of

[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. While a proof for the calculus of constructions has been known for a while (cf. Streicher's work [10]), precise and formalised proofs for intensional Martin-Löf type theory became available only recently (cf. the talks presenting work by de Boer, Brunerie, Lumsdaine, and Mörtberg [11]–[14] as well as the licanciate thesis by de Boer [15].

a semicategory of contexts and substitutions:

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

identity morphisms as identity substitutions:

$$
\begin{array}{lcll}
\mathsf{id} & : & \mathsf{Sub}\,\Gamma\,\Gamma & (5)\\
\mathsf{idl} & : & \mathsf{id} \diamond \sigma = \sigma & (6)\\
\mathsf{idr} & : & \sigma \diamond \mathsf{id} = \sigma & (7)
\end{array}
$$

a terminal oject as empty context:

$$
\begin{array}{lcll}
\bullet & : & \mathsf{Con} & (8)\\
\epsilon & : & \mathsf{Sub}\,\Gamma\,\bullet & (9)\\
\bullet\eta & : & \forall(\sigma : \mathsf{Sub}\,\Gamma\,\bullet).\ \sigma = \epsilon & (10)
\end{array}
$$

a presheaf of types:

$$
\begin{array}{lcll}
\mathsf{Ty} & : & \mathsf{Con} \to \mathcal{U} & (11)\\
\_[\_]^{\mathsf{T}} & : & \mathsf{Ty}\,\Delta \to \mathsf{Sub}\,\Gamma\,\Delta \to \mathsf{Ty}\,\Gamma & (12)\\
[\mathsf{id}]^{\mathsf{T}} & : & A[\mathsf{id}]^{\mathsf{T}} = A & (13)\\
[\diamond]^{\mathsf{T}} & : & A[\sigma \diamond \delta]^{\mathsf{T}} = A[\sigma]^{\mathsf{T}}[\delta]^{\mathsf{T}} & (14)
\end{array}
$$

a (covariant) presheaf representing terms:

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

context extension, modelled by representability:

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

Fig. 1: The formulation of a category with families (CwF) as a generalised algebraic theory (GAT), as given by Kaposi and others [1], [2]. Presentation-wise, it slightly differs from (but is equivalent to) the similar suggestion by Altenkirch and Kaposi [3]. 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 HoTT book [4, Chp. 2.3]. Implicit arguments are omitted. $\mathcal{U}$ is a universe at any level.

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 [3], is shown in Fig. 1.[4] If one works in a type theory that has *quotient inductive-inductive* [28] or *higher inductive-inductive types* [29], 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. the work by Altenkirch and Kaposi [3] and Coquand et al. [30]), sometimes known as the *meta-circular interpretation*. It is based on the 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. Therefore, anything we do in the syntax (the object theory) can be interpreted as a construction in the host theory. Shulman [31] argues that we should expect this to be possible if type theory is to be seen as a general purpose programming language, as any such language should be able to implement its own interpreter or compiler. This motivates the formulation of the following somewhat vague task:

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.

[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.

It seems difficult, but also slightly besides the point, to make the above *General Problem* completely precise to exclude trivial solutions; a satisfactory solution is something of the category "we know it when we see it". The expectation is that a *model* should allow at least the constructions listed in Fig. 1, i.e. each component of that figure should be definable. An (algebraic) definition of a model will give rise to the definitions of *model morphism* and thus *initiality* in a canonical way. Since we don't want the initial model to be trivial, the notion of model should contain base types; these are omitted in Fig. 1 (and in most parts of the current paper) for simplicity, but easy to add, cf. [3].

The most interesting question is arguable what it means that the initial model can be viewed as syntax. Note that the equality type of the host theory plays the role of definitional/judgmental equality in the (initial) model. Therefore, we expect that the equality of the initial model is *proof-irrelevant*, i.e. Con, Sub, Ty, Tm are *(homotopy) sets* in the terminology of homotopy type theory (but see footnote Footnote 6).

If we want to model an intensional type theory (which is the primary case of interest here), it is further desirable that the equality in the initial model is *decidable*, corresponding to decidable type checking. This means for example that, for $\Gamma : \mathsf{Con}$, we have

$$(A\, B : \mathsf{Ty}\, \Gamma) \to (A = B) + \neg(A = B). \tag{27}$$

If we instead model extensional Martin-Löf type theory, we cannot expect this (cf. Clairambault and Dybjer's work [32]), but the requirement of proof-irrelevance remains.

In intensional dependent type theory with the axiom of *unique identity proofs (UIP)*, i.e. the assumption that *all* equalities are proof-irrelevant, and quotient inductive-inductive types [28], a solution to the problem has been given by Altenkirch and Kaposi. They construct the initial model with the standard interpretation [3] and show that the initial model has decidable equality [33].

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 homotopy type theory (HoTT) [4], which rejects UIP, is able to handle its own meta-theory. The question was first raised by Shulman [31] and has since then been discussed very actively, in particular by Escardó and Xu [18], Gylterud, Lumsdaine, and Palmgren [20], Buchholtz [21], and Altenkirch [34]. Progress in the setting without UIP has also been made by Abel, Öhman, and Vezzosi [9]. The core task, as described by the *General Problem* above, is still unsolved.

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, Huber, and Sattler [2], 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 expected properties will not hold, a very common phenomenon in homotopy type theory. In particular, the initial model will not have proof-irrelevant (or even decidable) equality, making it unsuitable as "syntax" in any form. Phrased differently, the expected quotiented 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 (i.e. that equality has to be irrelevant at some level). As a prominent example, the notion of *category* considered by Ahrens, Kapulkin, and Shulman [35] requires morphisms to form a set to ensure well-behavedness. Unfortunately, this would not be a satisfactory solution in our situation. Since univalent universes in homotopy type theory are provably not set-truncated [4, Ex. 3.1.9], any such truncatedness assumption would mean 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 or other versions of dependent type theory without UIP. 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*. Thus, 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 [36]), as higher equalities of any levels will be possibly non-trivial data [37].

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 in the standard reference, the "HoTT book" [4]. To circumvent this problem, we work in a *two-level type theory* [38], [39] which makes the construction possible. This nevertheless has significant implications for "ordinary" homotopy type theory, where all finite special cases *can* be expressed. Therefore, our construction automatically gives a definition in HoTT of $(2, 1)$-CwF's, $(3, 1)$-CwF's, and more generally $(n, 1)$-CwF's for any externally fixed natural number n. This already strongly generalises ordinary CwF's, which are merely $(1, 1)$-CwF's in this scheme.

The specific approach to ∞-categories that we work with is based on the *semi-Segal types* suggested independently by Capriotti [40] and Schreiber [41], further studied by the current author in joint work with Capriotti [42] as well as with Annenkov, Capriotti, and Sattler [39]. While this approach immediately gives a definition of an *∞-semicategory*, adding the identities in a well-behaved manner is trickier than 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 is unique (i.e. a propositional property rather than data) and therefore automatically fully coherent. A further propositional property expressing univalence can be added optionally on top.

Related work: The original connection between higher categories (in the form of spaces) and type theory, discovered by Awodey and Warren [43] and by Voevodsky (see the presentation by Kapulkin and Lumsdain [44]), marks 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 [44] define one particular model, the current paper defines a type ("classifier") of models. Similarly, Barras, Coquand, and Huber [45] construct a (concrete) model interpreting types as semisimplicial sets. Boulier [46] and other authors discuss and formalise various models assuming UIP locally or globally. Abel, Öhman, and Vezzosi [9] formalise the syntax in a type theory 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 the current author did not manage to use their approach for a solution of the *General Problem* discussed above. Complete semi-Segal types are a known approach to univalent ∞-categories [39], [40], [42], and the current paper uses a variation of those in order to avoid completeness/univalence. Nguyen and Uemura [47] 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's are in principle general enough to also be a suitable notion of model for ∞-type theories. Capriotti and Sattler [48] 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 II 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 several examples break. For our higher-categorical approach, we want to use semisimplicial types to construct semi-Segal types with a new notion of identity, introduced in Section III. The heart of the paper is Section IV, where we develop enough internal ∞-category theory in order to define ∞-CwF's. In Section V, we will see that all the examples that were broken without UIP work when using ∞-CwF's. In Section VI, 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 II, we assume that the host theory satisfies UIP in order to discuss the standard approaches and why they fail without UIP. From Section III on, we drop this assumption and work in homotopy type theory with all definitions and features introduced in the "HoTT book" [4]. In particular, we use the notions of *proposition* (-1*-type*), *set* (*0-type*) and more generally n*-type*, as well as the *path over* notation expressing substitution/transport ($a =_p b$ means $\mathsf{subst}(p, a) = b$).

From Section III on, we assume that the host theory is equipped with a second kind of equality type, turning it into a two-level type theory. A very brief introduction is given in Section III-A, but all ideas of the current paper should be understandable without knowledge of two-level type theory as it behaves essentially like "ordinary" Martin-Löf type theory. For details, we refer to [39].

Agda formalisation: The discussed *identities via idempotent equivalences*, which turn an ∞-semicategory into an ∞-category, are a small part of the paper but represent a core idea for the construction of ∞-CwF's. The idea and all proofs can be formulated in a very simple setting that can be formalised without having to refer to 2LTT. An Agda implementation is available on GitHub.[5]

II. CwF's: 1-Categorical Models of Dependent Type Theory

A. 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, due to Kaposi and others [1], [2], which presents the components of type theory as a generalised algebraic theory. 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 [3] (as well as [1], [2]) 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

[5]https://github.com/nicolaikraus/idempotentEquivalences; and as `html`: nicolaikraus.github.io/docs/html-idempotentequivalences/Identities.html

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 the categorical formulation of a CwF [22]. 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 Set of sets and functions. 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 $f \in \mathcal{D}_1(d,e)$ 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 triangle 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

$$(\mathcal{C}/\Delta)^{\mathsf{op}} \to \mathsf{Set} \tag{28}$$
$$(\Gamma, \sigma) \mapsto \mathsf{Tm}(\Gamma, A[\sigma]^{\mathsf{T}}). \tag{29}$$

Fig. 1 can be seen as a direct implementation of Definition 1 in type theory. The correspondence is made clear by the subheadlines in the figure.

B. 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)* [28], [49], 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 [3], and we refer to their construction as the *syntax QIIT*. 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 [33] further show that the syntax QIIT 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:

$$\mathsf{Con} :\equiv \mathcal{U} \tag{30}$$
$$\mathsf{Sub}\,\Gamma\,\Delta :\equiv \Gamma \to \Delta \tag{31}$$
$$\delta \diamond \sigma :\equiv \delta \circ \sigma \tag{32}$$
$$\mathsf{id} :\equiv \lambda x.x \tag{33}$$
$$\bullet :\equiv \mathsf{Unit} \tag{34}$$
$$\mathsf{Ty}\,\Gamma :\equiv \Gamma \to \mathcal{U} \tag{35}$$
$$A[\sigma]^{\mathsf{T}} :\equiv A \circ \sigma \tag{36}$$
$$\mathsf{Tm}\,\Gamma\,A :\equiv \Pi(x:\Gamma).(A\,x) \tag{37}$$
$$t[\sigma]^{\mathsf{t}} :\equiv t \circ \sigma \tag{38}$$
$$\Gamma \triangleright A :\equiv \Sigma(x:\Gamma).(A\,x) \tag{39}$$
$$\mathsf{p} :\equiv \mathsf{proj}_1 \tag{40}$$
$$\mathsf{q} :\equiv \mathsf{proj}_2 \tag{41}$$
$$(\sigma, t) :\equiv \lambda x.(\sigma\,x, t\,x) \tag{42}$$
$$\ldots$$

Several authors (see e.g. [3], [18]) 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 $\mathcal{C}$.

The above examples work very well in a type theory where every type satisfies UIP. If we drop this assumption, be it because we want to be more general or other assumptions would be inconsistent with UIP (such as the univalence axiom/principle), we have to choose whether we adapt the definition of a CwF. The first canonical possibility is to keep the previous definition, which leads to what we call a *wild CwF*:

Definition 5 (wild CwF)**.** A *wild CwF* is a 26-tuple $(\mathsf{Con}, \mathsf{Sub}, \ldots)$ of all the components in Fig. 1.

The other canonical possibility is to simply require that all the involved types satisfy UIP, i.e. are sets; this leads to the definition of a *set-CwF*:

Definition 6 (set-CwF)**.** A *set-CwF* is a wild CwF together with the following four components:

$$\mathsf{conset} : \mathsf{isSet}(\mathsf{Con}) \tag{43}$$
$$\mathsf{tyset} : (\Gamma : \mathsf{Con}) \to \mathsf{isSet}(\mathsf{Ty}\,\Gamma) \tag{44}$$
$$\mathsf{subset} : (\Gamma\,\Delta : \mathsf{Con}) \to \mathsf{isSet}(\mathsf{Sub}\,\Gamma\,\Delta) \tag{45}$$
$$\mathsf{tmset} : (\Gamma : \mathsf{Con}) \to (A : \mathsf{Ty}\,\Gamma) \to \mathsf{isSet}(\mathsf{Tm}\,\Gamma\,A) \tag{46}$$

Unfortunately, neither suggestion is satisfactory with the *General Problem* (cf. the introduction) in mind. Without UIP, the [lowest or higher] universe $\mathcal{U}$ is not necessarily a set anymore, and even provably not a set in HoTT. Thus, the standard model Example 3 is not a set-CwF.

While the standard model is a wild CwF, the remaining two examples break in the absence of UIP. Altenkirch and Kaposi's *syntax QIIT* (Example 2) is a wild CwF and remains a valid example, but is not the *initial* wild CwF. For example, the (higher) equality $\mathsf{idl}_{\mathsf{id}} = \mathsf{idr}_{\mathsf{id}}$ holds in the syntax QIIT, but there is no way to prove it from the components in Fig. 1 (assuming the type theory has base types). The situation is even worse for the slice model (Example 4), which cannot even be constructed. Both constructions fail for the same underlying problem, namely *coherence* data that is missing from the definition of a wild CwF. For the initiality, we have discussed above that at least a datum that gives us $\mathsf{idl}(\mathsf{id}) = \mathsf{idr}(\mathsf{id})$ would be needed, and for the slice construction, "MacLane's pentagon coherence" is missing. 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. the use of ∞-categories, amounts to adding *all* this data.

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. Without UIP, neither the unmodified definition of a CwF nor the definition extended with the requirement that all involved types satisfy UIP is satisfactory.

III. Internal ∞-Dimensional Categories

Definition 5 defines a wild CwF to be a tuple with 26 components. Even more extreme, Definition 6 defines a set-CwF to consist of 30 components. While this is tedious to write these down in type theory, expressing such a definition is entirely straightforward to 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 [50]).

A non-trivial example for a structure with infinitely many components are *semisimplicial types* [51], [52], and these are important for the approach to ∞-categories that we chose in this paper. It is a long-standing open problem whether they can be defined in homotopy type theory. Therefore, we work in *two-level type theory* (2LTT [38], [39], [53]), an extension of HoTT which allows us to work with semisimplicial types and similar structures.

In order to understand the later ideas of this paper, semisimplicial types are important. Therefore, we first briefly recall what they are. Two-level type theory is more of a technical framework which is required to perform the constructions in full generality, but it is not crucial in order to understand the ideas; a brief explanation is given below and, for details, we refer to [39].

A. Semisimplicial Types

A semisimplicial type of dimension 2 is a tuple (A_0, A_1, A_2) of types and type families as follows:

$$A_0 : \mathcal{U} \tag{47}$$

$$A_1 : A_0 \to A_0 \to \mathcal{U} \tag{48}$$

$$A_2 : \{x_0\, x_1\, x_2 : A_0\} \to (A_1\, x_0\, x_1) \to (A_1\, x_1\, x_2) \to (A_1\, x_0\, x_2) \to \mathcal{U} \tag{49}$$

We think of A_0 as a type of *points* or *vertices*, of $A_1\, x_0\, x_1$ as *lines*, of A_2 with three lines as arguments as a type of *triangle fillers*, and it is (at least on an intuitive level) clear what a semisimplicial type of dimension 3 (extend the above with a family A_3 of tetrahedron fillers) would be, and so on.

Meta-theoretically (and in 2LTT), the semisimplicial type (A_0, A_1, A_2) represents a type-valued presheaf on the index category

$$\begin{tikzcd} {[0]} \arrow[r, shift left] \arrow[r, shift right] & {[1]} \arrow[r, shift left=2] \arrow[r] \arrow[r, shift right=2] & {[2]} \end{tikzcd} \tag{50}$$

which is given by

$$\begin{tikzcd} A_0 & \Sigma(x_0\, x_1 : A_0).A_1\, x_0\, x_1 \arrow[l, shift left] \arrow[l, shift right] & \begin{array}{c} \Sigma(x_0\, x_1\, x_2 : A_0). \\ \Sigma(x_{01} : A_1\, x_0\, x_1). \\ \Sigma(x_{12} : A_1\, x_1\, x_2). \\ \Sigma(x_{02} : A_1\, x_0\, x_2). \\ A_2\, x_{01}\, x_{12}\, x_{02} \end{array} \arrow[l, shift left=2] \arrow[l] \arrow[l, shift right=2] \end{tikzcd} \tag{51}$$

The morphism part of this presheaf (i.e. the functions in (51)) are simply projections. It works out such that the usual functor laws hold *definitionally* thanks to η for Σ-types. This style of presentations of structure as type families described by an index category are known since at least Makkai's FOLDS [54].

A "full" semisimplicial type is a presheaf on the simplex category without degeneracies, Δ_+, of which (50) is an initial segment. 2LTT is a version of type theory where this can be expressed. The main idea of 2LTT that we need to keep in mind for the rest of this paper is that it has a *strict* layer on top of "normal HoTT". The strict layer contains *strict natural numbers* $\mathbb{N}^{\mathsf{s}}$ as well as *strict equality* ($=^{\mathsf{s}}$), and more generally contains constructions that would usually only work in the meta-theory; e.g. a number $n : \mathbb{N}^{\mathsf{s}}$ behaves like an externally fixed natural number (a *numeral*), and strict equality plays the role of judgmental/definitional equality (at least from the point of view of "normal HoTT"). For clarity, we refer to types in "normal HoTT" as *fibrant* and types in the strict layer as *strict*.

Strict equality allows us to talk about strict categories, strict functors, and strict natural transformations. A natural example of a strict category is the universe. The strict category Δ_+ has $n : \mathbb{N}^{\mathsf{s}}$ as objects, and a morphism from m to n is an injective and monotone function $\mathsf{Fin}_m \to \mathsf{Fin}_n$. Semisimplicial types are strict functors satisfying the so-called *Reedy fibrancy* [55] condition which essentially says that the functor can be described by type families as in (51). In a similar fashion, it is possible to define a family of semisimplicial types indexed by another semisimplicial type, which can also be described as a *semisimplicial type over a semisimplicial type* or a *displayed semisimplicial type*:

Definition 7 (semisimplicial type in 2LTT [39])**.** A semisimplicial type is a strict Reedy fibrant functor $A : \Delta_+^{\mathsf{op}} \overset{RF}{\Longrightarrow} \mathcal{U}$. A semisimplicial type B over A is a strict natural transformation from B to A that is also a Reedy fibration.

Intuitively, Definition 7 describes a semisimplicial type as an infinite tuple $(A_0, A_1, \ldots)$, and a semisimplicial type over it as a sequence $(B_0, B_1, \ldots)$ with $B_0 : A_0 \to \mathcal{U}$, $B_1 : (x_0\, x_1 : A_0) \to B_0\, x_0 \to B_0\, x_1 \to \mathcal{U}$, and so on.

B. Contexts and substitutions, first part: semi-Segal types

Segal spaces (Rezk [56]), 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 [40] and others, but note that the degeneracies are missing. We briefly sketch the definition as presented in [39], [40].

On the strict level, and for $n : \mathbb{N}^{\mathsf{s}}$, we have the representable functor $\Delta[n]$. It can be viewed as 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 $k : \mathbb{N}^{\mathsf{s}}$, the *horn* $\Lambda^k[n]$ is the semisimplicial set obtained from the representable $\Delta[n]$ by removing the identity morphism on Fin_n and the monotone injection $\mathsf{Fin}_{n-1} \to \mathsf{Fin}_n$ which does not have k in its image. There is an obvious inclusion $\Lambda^k[n] \hookrightarrow \Delta[n]$ that is used in the following definition:

Definition 8 (Segal condition [39], [56])**.** A semisimplicial type A satisfies the Segal condition if, for any given n and $0 < k < n$, any given $\eta : \Lambda^k[n] \to A$ can uniquely be extended to a natural transformation $\Delta[n] \to A$, in the sense that the type of extensions is contractible.

We remark that A is often called *local* with respect to the inclusion $\Lambda^k[n] \hookrightarrow \Delta[n]$.

In previous work with Capriotti [42], we 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:

$$\begin{aligned} h_2 :\ & \{x_0\, x_1\, x_2 : A_0\} \to (x_{01} : A_1\, x_0\, x_1) \to \\ & (x_{12} : A_1\, x_1\, x_2) \to \\ & \mathsf{isContr}\,(\Sigma(x_{02} : A_1\, x_0\, x_2).A_2\, x_{01}\, x_{12}\, x_{02}) \end{aligned} \tag{52}$$

We also call this a *horn-filling condition* or the condition that the horn $x_0 \xrightarrow{x_{01}} x_1 \xrightarrow{x_{12}} x_2$ has a *contractible type of fillers*.

In [42], an explicit translation between horn-filling conditions and the structure one expects of (higher) semicategories is constructed. It not hard to see that having the pair (A_2, h_2) is equivalent to having a composition operator $(\_ \circ \_) : A_1\, x_0\, x_1 \to A_1\, x_1\, x_2 \to A_1\, x_0\, x_2$: having an element of a type (here $A_1\, x_0\, x_2$) is equivalent to having a family over the type (here $A_2\, x_{01}\, x_{12} : A_1\, x_0\, x_2 \to \mathcal{U}$) and a proof that it is inhabited at exactly one point (here $h_2\, x_{01}\, x_{12}$). On the next level, we require that any map $\Lambda^1[3] \to A$ can uniquely be extended to $\Delta[3] \to A$, and 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 [42]. This motivates the following definition:

Definition 9 (∞-semicategory [40])**.** A semisimplicial type A is an ∞-*semicategory* (also known as a *semi-Segal type*) if it satisfies the Segal condition.

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

Definition 10 (inner fibration)**.** An *inner fibration*, or an ∞*-semicategory over an* ∞*0semicategory*, is a Reedy fibration $\eta : E \twoheadrightarrow A$ such that, for all n, $0 < k < n$, and squares

$$\begin{tikzcd} \Lambda^k[n] \arrow[r] \arrow[d, hook] & E \arrow[d, two heads, "\eta"] \\ \Delta[n] \arrow[r] \arrow[ur, dashed] & A \end{tikzcd} \tag{53}$$

of the form shown on the right, the type of fillers (i.e. the type of the dashed map) is contractible. It is worth noting that, by definition, a semisimplicial type A is an ∞-semicategory exactly if $A \twoheadrightarrow 1$ is an inner fibration.

C. Contexts and substitutions, second part: identities

The natural next question is how the definition of an ∞-semicategory can be extended to capture ∞-categories. To do this, we need to add a property which ensures that the ∞-semicategory has identities. We want this property to satisfy the following conditions:

1) Of course, the property should give us (i.e. allow us to construct) the "naïve" identities that are shown in Fig. 1: For every object, we want an endomorphism id such that $\mathsf{id} \diamond \sigma = \sigma$ and $\sigma \diamond \mathsf{id} = \sigma$.
2) The property should be a proposition. In other words, we want an ∞-semicategory to be an ∞-category in at most one way. This is important because we do not only want to talk about a single ∞-category in isolation, but about a *type* of ∞-categories.

3) From the perspective of HoTT, it is natural to want *univalent* (or *saturated*) ∞-categories, meaning that identities/equalities and isomorphisms coincide (analogous to univalent 1-categories [35]). As one might expect, the standard model will indeed turn out to be a *univalent* ∞-CwF. However, not all examples that we want can possibly be univalent: In a type theory, contexts can be isomorphic in non-trivial ways, meaning that the type Con in a univalent ∞-CwF cannot be a set. This would rule out the "syntax" (Example 2). Phrased differently, the initial model of a non-trivial type theory would not be on the level of sets (and will in particular not have decidable equality for contexts), which contradicts the expectation by Altenkirch [34] and others.[6] Therefore, we want a more general definition of "being an ∞-category" (a.k.a. "having identities") which allows us to study both univalent and non-univalent variants.

The question of adding identities to ∞-semicategories has been studied outside the field of type theory long before HoTT was researched. It is known [57] 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 [58] and Harpaz [59] have suggested a simple condition that expresses the presence of suitable identities. The translation of their idea into type theory leads to the *complete semi-Segal types* by Capriotti [40] and others. Complete semi-Segal types can be taken as the definition of type-theoretic univalent ∞-categories, but the univalence condition is built into the definition of identities.

A *direct replacement* of the full simplex category has been suggested by Sattler and the current author [60], which gives rise to (a replacement of) simplicial types and thereby (not necessarily univalent) ∞-categories. Another direct replacement was given by Kock [61]. These approaches both work by adding an infinite tower of data.

The current paper gives a new and different definition of identities in ∞-semicategories. The idea is motivated by [60] and inspired by what is known as *dunce's hat* in topology [62], 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 satisfies the three properties listed above. This definition corresponds to a characterisation of identities that was independently suggested by Lai [63], who considered $(\infty, 1)$-categories in a slightly different version of type theory.

[6]In a private conversation with the current author, Eric Finster has pointed out that type-checking algorithms do not rely on context equality being decidable. In other words, one could potentially drop the condition that contexts of the syntactic model form a set and only keep the condition for substitutions, types, and terms. This may make a univalent syntactic model possible.

This author expects that the definition of identities via idempotent equivalences is equivalent to both of the mentioned "elaborate" definitions [60], [61]. If we add univalence, the equivalence with complete semi-Segal types is obvious.

Our definition of an identity does not need the whole infinite structure of an ∞-semicategory, but only the first four levels (A_0, A_1, A_2, A_3). Using the translation explained in Section III-B, we phrase this subsection in the (maybe more familiar) language of a semicategory $(\mathsf{Ob}, \mathsf{Hom}, \diamond, \mathsf{assoc})$, without any requirement of set-truncation. This allows us to present the development in a very elementary way, without the need to allude to infinite structures. Below, we state the precise definitions and theorems; the detailed proofs themselves are reasonably elementary and can be found in the Agda formalisation (see footnote 5 on page 4).

Definition 11 (identities via idempotent equivalences)**.** In a wild semicategory $(\mathsf{Ob}, \mathsf{Hom}, \diamond, \mathsf{assoc})$, a morphism $e : \mathsf{Hom}(x, y)$ is an *equivalence* (*neutral morphism* in [42]), written $\mathsf{iseqv}(e)$, if both composition operations

$$(\_ \diamond e) : \Pi\{z : \mathsf{Ob}\}.\mathsf{Hom}(y, z) \to \mathsf{Hom}(x, z) \tag{54}$$

$$(e \diamond \_) : \Pi\{w : \mathsf{Ob}\}.\mathsf{Hom}(w, x) \to \mathsf{Hom}(w, y) \tag{55}$$

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 a *good identity* if it is an equivalence and idempotent. The wild semicategory $C \equiv (\mathsf{Ob}, \mathsf{Hom}, \diamond, \mathsf{assoc})$ has a *good identity structure* if there is an identity for every object,

$$\begin{aligned}\mathsf{hasGoodIdStruc}(C) :\equiv \Pi(x : \mathsf{Ob}).\Sigma(i : \mathsf{Hom}(x, x)).\\ \mathsf{iseqv}(i) \times (i \diamond i = i).\end{aligned} \tag{56}$$

Theorem 12 (Identities are good iff left and right neutral)**.** *A morphism $i : \mathsf{Hom}(x, x)$ is a good identity if and only if, for all composable morphisms $\xrightarrow{f}\xrightarrow{i}\xrightarrow{g}$, we have $i \diamond f = f$ and $g \diamond i = g$.*

Theorem 13 (Uniqueness of good identities)**.** *For a given semicategory $\mathcal{C}$, the type $\mathsf{hasGoodIdStruc}(C)$ is a proposition.*

Phrased in the language of an ∞-semicategory $(A_0, A_1, A_2, \ldots)$, a morphism $e : A_1\,x\,y$ is an equivalence if any horn of the form $1 \xleftarrow{e} 0 \to 2$ or $0 \to 2 \xleftarrow{e} 1$ 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 14. An ∞-category is an ∞-semicategory with a good identity structure.

Univalence for ∞-categories is an optional additional property that one can add; it is not discussed further in this paper. From the next subsection on, we will drop the attribute *good* and simply speak of *identities* rather than *good identities*. There is no risk of confusion, especially since Theorem 12 essentially shows that *every* identity is good.

Section summary. A *semisimplicial type* is a type-valued presheaf on the category Δ_+, the category of finite (non-empty) sets and strictly increasing functions. Intuitively, a semisimplicial type is an infinite tuple $(A_0, A_1, A_2, \ldots)$; this definition can be formulated in 2LTT.
A semisimplicial type forms the "raw structure" of a higher category. If we equip it with the *Segal condition* (which is itself a propositional property), we get an ∞-semicategory.
Finally, an ∞-semicategory with an idempotent equivalence for every object is a not-necessarily-univalent notion of ∞-category in type theory.

IV. Internal ∞-CwF's

A. The empty context: a terminal object

Modelling the empty context as a terminal object is standard.

Definition 15 (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.

B. 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 20. For now, we concentrate on the first possibility which is closer to the 1-categorical formulation discussed in Section II.

Let us start with the last point (3). A morphism in a (strict) functor category is a (strict) natural transformation, and ∞-categories are certain (strict) 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:

$$F_0 : A_0 \to B_0 \tag{57}$$

$$F_1 : \{x_0\, x_1 : A_0\} \to (A_1\, x_0\, x_1) \to (B_1\, (F_0\, x_0)\, (F_0\, x_1)) \tag{58}$$

$$\begin{aligned} F_2 : {} & \{x_0\, x_1\, x_2 : A_0\} \to \{x_{01} : A_1\, x_0\, x_1\} \to \\ & \{x_{12} : A_1\, x_1\, x_2\} \to \{x_{02} : A_1\, x_0\, x_2\} \to \\ & (A_2\, x_{01}\, x_{12}\, x_{02}) \to (B_2\, (F_1\, x_{01})\, (F_1\, x_{12})\, (F_1\, x_{02}) \end{aligned} \tag{59}$$

Being a strict map between the underlying semisimplicial types already ensures that F preserves the compositionality structure. We can make this transparent on level 2 via the translation explained in Section III-B, under which the last component F_2 translates to:

$$\begin{aligned} F_2' : {} & \{x_0\, x_1\, x_2 : A_0\} \to \{x_{01} : A_1\, x_0\, x_1\} \to \\ & \{x_{12} : A_1\, x_1\, x_2\} \to \\ & (F_1\, x_{12}) \diamond (F_1\, x_{01}) = F_1 (x_{12} \diamond x_{01}) \end{aligned} \tag{60}$$

∞-categories come with a proof that the underlying functors are Reedy fibrant, but the strict 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 16 (∞-functor)**.** An ∞-*functor* between ∞-categories is a strict natural transformation which maps identities to equivalences.

Remark 17. A strict natural transformation between ∞-categories does not automatically preserve equivalences or even identities. Let $\top$ be the terminal ∞-category, which is the unit type Unit on every level. Let A be the ∞-category defined by $A_0 :\equiv \mathsf{Unit}$, $A_1\, \_\, \_ :\equiv (2 \to 2)$, $A_2\, f\, g\, h :\equiv (g \circ f = h)$, and A_{k+3} being constantly Unit. 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 [64, 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 $\mathcal{T}_2\, f\, g\, h \equiv (g \circ f = h)$. We briefly sketch the complete construction given in [39]. Given any strict category C, one can define a strict functor $\mathsf{N}_+(C) : \Delta_+^{\mathsf{op}} \overset{s}{\Rightarrow} \mathcal{U}$ via the usual nerve construction which defines $\mathsf{N}_+(C)_n$ to be (fibrant) type of sequences $X_0 \to X_1 \to \ldots \to X_n$. This strict functor is not Reedy fibrant, but via a general Reedy fibrant replacement construction [39, 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 18 (∞-category of small types)**.** The ∞-category $\mathcal{T}$ is given as $R(\mathsf{N}_+(\mathcal{U}))$, where $\mathcal{U}$ is the strict category of types and functions.

C. 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 vertices $x_0, x_1, \ldots$, lines $x_{01}, \ldots$, and so on, 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 that has been (partially) discussed in [39]. We define $\mathsf{N}^{\bullet}_{+}(A, F)$ to be the strict 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 strict natural transformation $\eta : \mathsf{N}^{\bullet}_{+}(A, F) \to \mathsf{N}_{+}(A)$ which, on each level, forgets the point. Using the mentioned Reedy fibrant replacement [39, 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 is the following, where we annotate arguments with their types for readability and happly is the function given in [4, Eq. 2.9.2]:

$$\mathcal{T}^{\bullet}_0\,(X : \mathcal{U}) \quad :\equiv \quad X \tag{61}$$

$$\mathcal{T}^{\bullet}_1\,(f : X \to Y)\,(x : X)\,(y : Y) \quad :\equiv \quad (f\, x = y) \tag{62}$$

$$\begin{aligned} &\mathcal{T}^{\bullet}_2\,(\alpha : g \circ f = h)\,(e_0 : f\, x = y) \\ &\qquad (e_1 : g\, y = z)\,(e_2 : h\, x = z) \quad :\equiv \\ &\qquad\qquad e_2 = (\mathsf{happly}\,\alpha\, x) \cdot (\mathsf{ap}_g\, e_0) \cdot e_1 \end{aligned} \tag{63}$$

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 [39]) that Reedy fibrations are closed under pullback.

Definition 19 (∞-category of elements)**.** For an ∞-category $\mathcal{A}$ and a functor $F : \mathcal{A} \to \mathcal{T}$, the *$\infty$-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.

$$\begin{tikzcd} \int_{\mathcal{A}} F \arrow[r, dashed] \arrow[d, dashed, two heads, "\pi_1"'] \arrow[dr, phantom, "\lrcorner", very near start] & \mathcal{T}^{\bullet} \arrow[d, two heads] \\ \mathcal{A} \arrow[r, "F"] & \mathcal{T} \end{tikzcd} \tag{64}$$

It is standard that pullbacks are closed under 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). Therefore, $\int_{\mathcal{A}} F$ is an ∞-category.

Remark 20 (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 10 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 10 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 (64) is also a homotopy pullback since the vertical maps are Reedy fibrations.

D. Context extension

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 II). Even for ∞-categories, representability can be stated referring only to the lowest levels of the category, i.e. using finitely many components. However, the presentation of Fig. 1 relies on UIP and does not work out of the box; instead, we define:

Definition 21. 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 and unfolding leads us to:

Definition 22 (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 14. 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 15. 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 18) 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 16) is a strict natural transformation between semisimplicial types 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 19) is defined by first constructing the ∞-category of pointed types.
5) A context extension structure, cf. Definition 22. Context extension can be represented in a finite way even without referring to the strict layer of 2LTT.

Christian Sattler has suggested an alternative, where the above functors Ty and Tm are replace by a sequence of right fibrations, $\mathsf{Tm} \twoheadrightarrow \mathsf{Ty} \twoheadrightarrow \mathcal{C}$. Context extension can then be described as a right adjoint of the first fibration, in line with Awodey's presentation [23].

V. Examples of ∞-Categories with Families

A. The Syntax as a QIIT

Every set-CwF in the sense of Definition 6 and Fig. 1 can be presented as an ∞-CwF. Therefore, the *syntax QIIT* by Altenkirch and Kaposi [3], 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: $\mathcal{C}_{3+n}(\_) :\equiv \mathsf{Unit}$. The other parts are constructed analogously. Summarised, the construction of an ∞-CwF from a set-CwF is easy because almost all components of the ∞-CwF become trivial.

Although we do not explain it in this paper, we can define what a morphism between ∞-CwF's is, which in turn lets us speak about the property of being the *initial* ∞-CwF (again, note that this only makes sense if we consider base types). We conjecture that the syntax is the initial ∞-CwF, but this seems to be highly non-trivial.

B. The Initial Model as a HIIT

The definition of an ∞-CwF is fully algebraic in the sense of Cartmell [27], 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 (47–49). For the Segal condition, this might be slightly harder to see; however, the specific formulation of the Segal condition via horn filling that we have chosen in Section III-B makes it doable: for every horn, we can generate a filler (with two operations), and that filler is equal to every other filler parallel to it (using two equations). This construction makes it possible to build the *free ∞-category* of a semisimplicial type if the type theory has an "infinite" version of higher inductive-inductive types as specified by Kaposi and Kovács [29], i.e. a HIIT where constructors are indexed over strict natural numbers. The other components of an ∞-CwF can be written as a GAT as well, giving us the initial ∞-CwF.

We do not know under which conditions this HIIT turns out to have decidable equality, but if it does, then it is in particular based on sets. If this is the case, it is also initial among the set-CwF's and thus equivalent to the syntax QIIT.

C. Higher Models from Strict Models and the Standard Interpretation

A *strict* CwF is a (1-categorical) CwF where most equations hold up to strict equality. As remarked in Example 3, this is in particular the case for the standard model. We show that, from a strict CwF, we get an ∞-CwF. Let us begin with a precise definition:

Definition 23 (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 of Definition 21 stated using the usual fibrant equality type ($=$), i.e. it suffices to have the conditions 5,6,7 in Definition 22 as fibrant equalities.

Lemma 24. *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 in [39] and sketched in Section IV-B. To construct the remaining components, we first generate strict semisimplicial diagrams by taking nerves N_+ and $\mathsf{N}_+^\bullet$ as in Section IV-C.

The Reedy fibrant replacement operation R constructed in [39] 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 [39, Cor 4.28], applied on the diagram on the right.

$$\begin{tikzcd}
X \arrow[r, "\tau"] \arrow[d] & Y \arrow[d] \\
R(X) \arrow[r, dashed, "R(\tau)"] \arrow[d, two heads] & R(Y) \arrow[d, two heads] \\
1 \arrow[r] & 1
\end{tikzcd}$$

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 20) resolves the issue. Context extension can be checked manually, since it only concerns the lowest levels. □

Applying Lemma 24 on the 1-categorical standard model (which is a strict CwF, see Example 3) shows that the standard model is also an ∞-CwF.

D. 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 simplical sets [64].

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-simplices $\Delta[n] \overset{s}{\Rightarrow} \mathcal{C}$. We define $(\mathcal{C}/\Gamma)_n$ to be the type of $(n+1)$-simplices 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 $(n+2)$ vertices 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 simplices and face maps as projections as in (51), face maps are those projections which do not remove the last vertex x_{n+1}.

One can check that $(\mathcal{C}/\Gamma)$ is a semi-Segal type. Further, if i is an identity on Θ 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 Theorem 12. Thus, $(\mathcal{C}/\Gamma)$ is an ∞-category. As in the 1-categorical case, the functors Ty and Tm as well as context extension are not affected by the slicing operation.

Section summary. In Section II, we have discussed that several 1-categorical CwF constructions break in the absence of UIP. In the current section, we have seen that the examples work (again) in the ∞-categorical setting.

VI. 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 Altenkirch-Kaposi *syntax QIIT* is initial as an ∞-CwF (after adding base types and other type 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 [31]). 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 a much weaker question, namely whether the initial ∞-CwF has trivial fundamental group, would be very interesting; but already this seemingly much simpler problem appears to be highly non-trivial, and similar results for seemingly simpler situations [65]–[67] suggest that a solution will require new techniques.

If it turns out that the *syntax QIIT* is indeed the initial ∞-CwF, we still have to face the problem of defining semisimplicial types before we can hope for a solution to the *General Problem* of whether "HoTT eats itself". The other direction seems in reach: If HoTT can eat itself, then we can define semisimplicial types. This has already been conjectured by Shulman [31] but not yet been made precise.

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* [68]–[72] is intriguing. Future directed type theories may allow us to develop ∞-category theory in a synthetic way, and the relationship to the semi-synthetic approach of this paper will be interesting to see.

Acknowledgements

This paper has benefited from many conversations 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 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 many 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 (the *syntax QIIT*) until the main points sunk in.

I am grateful to the research communities in Birmingham, Budapest, Nottingham, and Pittsburgh (CMU) for the opportunities to give talks and discuss the main ideas of this paper. These opportunities have led to valuable feedback and various improvements.

Finally, I thank the anonymous reviewers for their comments which have helped me to improve this paper.

References

[1] A. Kaposi, A. Kovács, and N. Kraus, "Shallow embedding of type theory is morally correct," in *Mathematics of Program Construction (MPC'19)*, 2019, pp. 329–365, available online at https://arxiv.org/abs/1907.07562.

[2] A. Kaposi, S. Huber, and C. Sattler, "Gluing for type theory," in *4th International Conference on Formal Structures for Computation and Deduction (FSCD 2019)*, ser. Leibniz International Proceedings in Informatics (LIPIcs), H. Geuvers, Ed., vol. 131. Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2019, pp. 25:1–25:19. [Online]. Available: http://drops.dagstuhl.de/opus/volltexte/2019/10532

[3] T. Altenkirch and A. Kaposi, "Type theory in type theory using quotient inductive types," in *Symposium on Principles of Programming Languages (POPL'16), SIGPLAN Not.*, vol. 51, no. 1. New York, NY, USA: ACM, Jan. 2016, pp. 18–29. [Online]. Available: http://doi.acm.org/10.1145/2914770.2837638

[4] T. Univalent Foundations Program, *Homotopy Type Theory: Univalent Foundations of Mathematics*. Institute for Advanced Study: http://homotopytypetheory.org/book/, 2013.

[5] E. Palmgren, "From type theory to setoids and back," *ArXiv*, 2019, available online at https://arxiv.org/abs/1909.01414.

[6] J.-P. Bernardy, T. Coquand, and G. Moulin, "A presheaf model of parametric type theory," *Electronic Notes in Theoretical Computer Science*, vol. 319, pp. 67 – 82, 2015, the 31st Conference on the Mathematical Foundations of Programming Semantics (MFPS XXXI). [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1571066115000730

[7] M. Sozeau and N. Tabareau, "Internalization of the groupoid interpretation of type theory," in *TYPES 2014*, 2014.

[8] M. Hofmann and T. Streicher, "The groupoid interpretation of type theory," in *In Venice Festschrift*. Oxford University Press, 1996, pp. 83–111. [Online]. Available: www.mathematik.tu-darmstadt.de/~streicher/venedig.ps.gz

[9] A. Abel, J. Öhman, and A. Vezzosi, "Decidability of conversion for type theory in type theory," in *Symposium on Principles of Programming Languages (POPL'18), Proceedings of the ACM on programming languages*, vol. 2, no. POPL. ACM, 2017, p. 23. [Online]. Available: https://doi.org/10.1145/3158111

[10] T. Streicher, "Investigations into intensional type theory," 1993, habilitationsschrift, Ludwig-Maximilians-Universität München.

[11] G. Brunerie, M. de Boer, P. L. Lumsdaine, and A. Mörtberg, "A formalization of the initiality conjecture in agda," 2019, talk given by Brunerie at the HoTT 2019 conference, slides available at https://guillaumebrunerie.github.io/pdf/initiality.pdf.

[12] P. L. Lumsdaine and A. Mörtberg, "Formalising the initiality conjecture in Coq," 2018, talk given by Lumsdaine at the Göteborg-Stockholm Joint Type Theory Seminar, slides available at http://peterlefanulumsdaine.com/research/Lumsdaine-2018-Goteborg-Initiality.pdf.

[13] G. Brunerie and P. L. Lumsdaine, "Formalising the initiality conjecture in coq and agda," 2018, talk at the Stockholm-Göteborg Type Theory Serminar.

[14] ——, "Initiality for Martin-Löf type theory," 2020, talk at the Homotopy Type Theory Electronic Seminar Talks (HOTTEST).

[15] M. de Boer, "A proof and formalization of the initiality conjecture of dependent type theory," Stockholm, Sweden, 2020, licentiate thesis, available online at https://su.diva-portal.org/smash/record.jsf?pid=diva2%3A1431287.

[16] J. Chapman, "Type theory should eat itself," *Electronic Notes in Theoretical Computer Science*, vol. 228, pp. 21–36, 2009.

[17] N. A. Danielsson, "A formalisation of a dependently typed language as an inductive-recursive family," in *Types for Proofs and Programs*, T. Altenkirch and C. McBride, Eds. Berlin, Heidelberg: Springer, 2007, pp. 93–109.

[18] M. H. Escardó and C. 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.

[19] A. Kaposi, "Type theory in a type theory with quotient inductive types," Ph.D. dissertation, School of Computer Science, University of Nottingham, Nottingham, UK, 2016, available online at http://eprints.nottingham.ac.uk/41385/1/th.pdf.

[20] P. L. Lumsdaine, "Formalising the categorical semantics of type theory, in type theory," 2015, talk at DMV, Hamburg, on joint work with Hakon Gylterud and Erik Palmgren.

[21] U. Buchholtz, "Formalizing type theory in type theory using nominal techniques," 2017, talk at HoTT/UF, Oxford.

[22] P. Dybjer, "Internal type theory," in *Types for Proofs and Programs (TYPES)*, ser. Lecture Notes in Computer Science, S. Berardi and M. Coppo, Eds., vol. 1158. Springer-Verlag, 1995, pp. 120–134.

[23] S. Awodey, "Natural models of homotopy type theory," *Mathematical Structures in Computer Science*, vol. 28, no. 2, pp. 241–286, 2018.

[24] B. Ahrens, P. L. Lumsdaine, and V. Voevodsky, "Categorical structures for type theory in univalent foundations," in *26th EACSL Annual Conference on Computer Science Logic (CSL 2017)*, ser. Leibniz International Proceedings in Informatics (LIPIcs), V. Goranko and M. Dam, Eds., vol. 82. Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2017, pp. 8:1–8:16. [Online]. Available: http://drops.dagstuhl.de/opus/volltexte/2017/7696

[25] ——, "Categorical structures for type theory in univalent foundations," *Logical Methods in Computer Science*, vol. Volume 14, Issue 3, Sep. 2018. [Online]. Available: https://lmcs.episciences.org/4814

[26] J. Cartmell, "Generalised algebraic theories and contextual categories," Ph.D. dissertation, Oxford, 1978.

[27] ——, "Generalised algebraic theories and contextual categories," *Annals of pure and applied logic*, vol. 32, pp. 209–243, 1986.

[28] T. Altenkirch, P. Capriotti, G. Dijkstra, N. Kraus, and F. N. Forsberg, "Quotient inductive-inductive types," in *Foundations of Software Science and Computation Structures (FoSSaCS 2018)*, C. Baier and U. Dal Lago, Eds. Springer International Publishing, 2018, pp. 293–310.

[29] A. Kaposi and A. Kovács, "Signatures and induction principles for higher inductive-inductive types," *Logical Methods in Computer Science*, vol. Volume 16, Issue 1, Feb. 2020. [Online]. Available: https://lmcs.episciences.org/6100

[30] T. Coquand, S. Huber, and C. Sattler, "Homotopy Canonicity for Cubical Type Theory," in *4th International Conference on Formal Structures for Computation and Deduction (FSCD 2019)*, ser. Leibniz International Proceedings in Informatics (LIPIcs), H. Geuvers, Ed., vol. 131. Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2019, pp. 11:1–11:23. [Online]. Available: http://drops.dagstuhl.de/opus/volltexte/2019/10518

[31] M. 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.

[32] P. Claiambault and P. Dybjer, "The biequivalence of locally cartesian closed categories and martin-löf type theories," *Mathematical Structures in Computer Science*, vol. 24, no. 6, p. e240606, 2014.

[33] A. Kaposi and T. Altenkirch, "Normalisation by evaluation for type theory, in type theory," *Logical Methods in Computer Science*, vol. 13, 2017.

[34] T. Altenkirch, "Towards the syntax and semantics of higher dimensional type theory," 2018, talk abstract for HoTT/UF, Oxford.

[35] B. Ahrens, K. Kapulkin, and M. Shulman, "Univalent categories and the Rezk completion," *Mathematical Structures in Computer Science (MSCS)*, pp. 1–30, Jan 2015. [Online]. Available: http://journals.cambridge.org/article_S0960129514000486

[36] J. E. Bergner, "A survey of (infinity, 1)-categories," in *Towards higher categories*. Springer, 2010, pp. 69–83.

[37] N. Kraus and C. Sattler, "Higher homotopies in a hierarchy of univalent universes," *ACM Transactions on Computational Logic (TOCL)*, vol. 16, no. 2, pp. 18:1–18:12, April 2015.

[38] V. Voevodsky, "A simple type system with two identity types," 2013, unpublished note.

[39] D. Annenkov, P. Capriotti, N. Kraus, and C. Sattler, "Two-level type theory and applications," *ArXiv*, 2019, available online at https://arxiv.org/abs/1705.03307.

[40] P. Capriotti, "Models of type theory with strict equality," Ph.D. dissertation, School of Computer Science, University of Nottingham, Nottingham, UK, 2016, available online at https://arxiv.org/abs/1702.04912.

[41] U. 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.

[42] P. Capriotti and N. Kraus, "Univalent higher categories via complete semi-Segal types," in *Symposium on Principles of Programming Languages (POPL'18), Proceedings of the ACM on Programming Languages*, vol. 2, no. POPL. New York, NY, USA: ACM, Dec. 2017, pp. 44:1–44:29. [Online]. Available: http://doi.acm.org/10.1145/3158132

[43] S. Awodey and M. A. Warren, "Homotopy theoretic models of identity types," *Mathematical Proceedings of the Cambridge Philosophical Society*, vol. 146, pp. 45–55, 2009. [Online]. Available: http://journals.cambridge.org/article_S0305004108001783

[44] C. Kapulkin and P. L. Lumsdaine, "The simplicial model of univalent foundations (after voevodsky)," *ArXiv e-prints*, November 2012, to appear in the Journal of the European Mathematical Society.

[45] B. Barras, T. Coquand, and S. Huber, "A generalization of the takeuti-gandy interpretation," *Mathematical Structures in Computer Science*, vol. 25, no. 5, pp. 1071–1099, 2015.

[46] S. Boulier, "Extending type theory with syntactic models," Ph.D. dissertation, École des Mines de Nantes, Nantes, France, 2018.

[47] H. K. Nguyen and T. Uemura, "∞-type theories," 2020, abstract presented at the online workshop HoTT/UF'20.

[48] P. Capriotti and C. Sattler, "Higher categories of algebras for higher inductive definitions," 2020, abstract for the conference TYPES'20.

[49] A. Kaposi and A. Kovács, "A syntax for higher inductive-inductive types," in *3rd International Conference on Formal Structures for Computation and Deduction (FSCD 2018)*, ser. Leibniz International Proceedings in Informatics (LIPIcs), vol. 108. Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2018, pp. 20:1–20:18. [Online]. Available: http://drops.dagstuhl.de/opus/volltexte/2018/9190

[50] B. Ahrens, P. Capriotti, and R. Spadotti, "Non-wellfounded trees in homotopy type theory," in *Typed Lambda Calculi and Applications (TLCA 2015)*, ser. Leibniz International Proceedings in Informatics (LIPIcs), vol. 38. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2015, pp. 17–30. [Online]. Available: http://drops.dagstuhl.de/opus/volltexte/2015/5152

[51] V. Voevodsky, P. L. Lumsdaine *et al.*, "Semi-simplicial types," Institute for Advanced Study, 2013, discussion preserved on the nLab, https://ncatlab.org/homotopytypetheory/show/semi-simplicial+types.

[52] N. Kraus, "On the role of semisimplicial types," 2018, abstract, presented at TYPES'18.

[53] T. Altenkirch, P. Capriotti, and N. Kraus, "Extending homotopy type theory with strict equality," in *25th EACSL Annual Conference on Computer Science Logic (CSL 2016)*, ser. Leibniz International Proceedings in Informatics (LIPIcs), vol. 62. Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2016, pp. 21:1–21:17. [Online]. Available: http://drops.dagstuhl.de/opus/volltexte/2016/6561

[54] M. Makkai, "First order logic with dependent sorts, with applications to category theory," 1995.

[55] C. Reedy, "Homotopy theory of model categories," 1974, unpublished, date estimated, available at http://www-math.mit.edu/~psh/.

[56] C. Rezk, "A model for the homotopy theory of homotopy theory," *Transactions of the American Mathematical Society*, vol. 353, no. 3, pp. 973–1007, 2001.

[57] C. P. Rourke and B. J. Sanderson, "Δ-sets I: Homotopy theory," *The Quarterly Journal of Mathematics*, vol. 22, no. 3, pp. 321–338, 1971.

[58] J. Lurie, "Higher algebra," Sep 2014, available at https://www.math.ias.edu/~lurie/.

[59] Y. Harpaz, "Quasi-unital ∞–categories," *Algebraic & Geometric Topology*, vol. 15, no. 4, pp. 2303–2381, 2015.

[60] N. Kraus and C. Sattler, "Space-valued diagrams, type-theoretically (extended abstract)," *ArXiv e-prints*, 2017, available online at https://arxiv.org/abs/1704.04543.

[61] J. Kock, "Weak identity arrows in higher categories," *International Mathematics Research Papers*, vol. 2006, 2006.

[62] E. Zeeman, "On the dunce hat," *Topology*, vol. 2, no. 4, pp. 341–358, 1963.

[63] L. Lai, "$(\infty, 1)$-categories in infinitary HoTT," 2018, unpublished note.

[64] J. Lurie, *Higher Topos Theory*, ser. Annals of Mathematics Studies. Princeton: Princeton University Press, 2009, also avaialabe online at http://arxiv.org/abs/math/0608040.

[65] N. Kraus and T. Altenkirch, "Free higher groups in homotopy type theory," in *Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science*, ser. LICS '18. New York, NY, USA: ACM, 2018, pp. 599–608. [Online]. Available: http://doi.acm.org/10.1145/3209108.3209183

[66] N. Kraus and J. von Raumer, "Path spaces of higher inductive types in homotopy type theory," in *34th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS'19)*. IEEE, 2019, pp. 1–13. [Online]. Available: https://doi.org/10.1109/LICS.2019.8785661

[67] ——, "Coherence via well-foundedness: Taming set-quotients in homotopy type theory," in *Symposium on Logic in Computer Science (LICS 2020)*. New York, NY, USA: Association for Computing Machinery, 2020, pp. 662–675. [Online]. Available: https://doi.org/10.1145/3373718.3394800

[68] D. R. Licata and R. Harper, "2-dimensional directed type theory," *Electronic Notes in Theoretical Computer Science*, vol. 276, pp. 263 – 289, 2011, twenty-seventh Conference on the Mathematical Foundations of Programming Semantics (MFPS XXVII). [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1571066111001174

[69] E. Riehl and M. Shulman, "A type theory for synthetic ∞-categories," *Higher Structures*, vol. 1, no. 1, 2017. [Online]. Available: https://journals.mq.edu.au/index.php/higher_structures/article/view/36

[70] A. Nuyts, "Towards a directed homotopy type theory based on 4 kinds of variance," Master's thesis, KU Leuven, Belgium, 2015. [Online]. Available: https://anuyts.github.io/

[71] P. R. North, "Towards a directed homotopy type theory," *Electronic Notes in Theoretical Computer Science*, vol. 347, pp. 223 – 239, 2019, proceedings of the Thirty-Fifth Conference on the Mathematical Foundations of Programming Semantics (MFPS'19). [Online]. Available: https://doi.org/10.1016/j.entcs.2019.09.012

[72] M. Z. Weaver and D. R. Licata, "A constructive model of directed univalence in bicubical sets," in *Proceedings of the 35th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS'20)*. New York, NY, USA: Association for Computing Machinery, 2020, p. 915–928. [Online]. Available: https://doi.org/10.1145/3373718.3394794