Semisimplicial Types in Internal Categories with Families

Joshua Chen and Nicolai Kraus

University of Nottingham, United Kingdom

Abstract

An open question in homotopy type theory, known as the problem of *constructing semisimplicial types*, is whether one can define a function $\mathsf{SST}\colon \mathbb{N} \to \mathsf{Type}_1$ such that $\mathsf{SST}(n)$ is the type of all configurations of triangles and tetrahedra of dimension up to n. We show in Agda that semisimplicial types can be constructed in any set-based internal category with families that contains Σ, Π, and a universe. This means that, given a CwF $(\mathsf{Con}, \mathsf{Ty}, \ldots)$ in type theory, we construct a function $\mathsf{SST}_\mathsf{c}\colon \mathbb{N} \to \mathsf{Con}$.

This project is work in progress, with code available at `github.com/jaycech3n/CwF`.

Semisimplicial Types. *Constructing semisimplicial types* [Uni13, Her15] is an open problem in homotopy type theory and, more generally, in dependent type theory without uniqueness of identity proofs (UIP). It was first discussed between Voevodsky, Lumsdaine and others during the Univalent Foundations special year at the IAS Princeton in 2012–13.

A semisimplicial type of dimension 2 is a tuple (A_0, A_1, A_2), where $A_0\colon \mathsf{Type}$ is a type of points, $A_1\colon A_0 \to A_0 \to \mathsf{Type}$ is a family of lines (for any two points), and $A_2\colon (x\,y\,z\colon A_0) \to A_1\,x\,y \to A_1\,y\,z \to A_1\,x\,z \to \mathsf{Type}$ is a family of triangle fillers (for any three points and three lines forming a triangle). Similarly, a *semisimplicial type of dimension* n should be a tuple $(A_0, \ldots, A_n)$ which represents families of simplices of dimension at most n. The open problem asks: can one can define a function $\mathsf{SST}\colon \mathbb{N} \to \mathsf{Type}_1$ such that $\mathsf{SST}(n)$ is equivalent to the (record/Σ-) type of such tuples $(A_0, \ldots, A_n)$?

Construction in Internal CwF's. By an *internal CwF*, we mean a type $\mathsf{Con}\colon \mathsf{Type}$ together with families $\mathsf{Sub}\colon \mathsf{Con} \to \mathsf{Con} \to \mathsf{Type}$, $\mathsf{Ty}\colon \mathsf{Con} \to \mathsf{Type}$, $\mathsf{Tm}\colon (\Gamma\colon \mathsf{Con}) \to \mathsf{Ty}\,\Gamma \to \mathsf{Type}$, and all the components and equalities which are needed to define a category with families [Dyb95]. We say that such a CwF is *set-based* if Con, Ty, Sub, Tm are families of sets in the sense of homotopy type theory, i.e. types satisfying UIP.

The goal of this project is to define, for any set-based internal CwF with Π and Σ-types and a universe U, a function $\mathsf{SST}_\mathsf{c}\colon \mathbb{N} \to \mathsf{Con}$ in Agda such that $\mathsf{SST}_\mathsf{c}\,n$ is the context $(A_0\colon U, A_1\colon A_0 \to A_0 \to U, \ldots, A_n\colon \ldots)$.

Motivation: Connecting Open Problems. A second open problem, originally asked by Shulman [Shu14], is whether homotopy type theory can internalize ("eat" [Cha09]) itself. More concretely, the problem is to formalize the syntax of HoTT inside HoTT as a set-based CwF, and to give interpretation functions which send the syntax to actual types and their elements in the "obvious" way—for example, a context in the CwF should be interpreted as the nested Σ-type of all its components. For a detailed discussion, see the introduction of [Kra21].

Our current project shows how a solution of this question would give rise to a solution to the problem of constructing semisimplicial types, as claimed by Shulman [Shu14]: composing SST_c with the interpretation $\mathsf{Con} \to \mathsf{Type}_1$, we would get the desired function $\mathbb{N} \to \mathsf{Type}_1$.

Related Work. We are aware of two different scripts which, when given a number n as input, produce valid Agda code for $\mathsf{SST}(n)$—one script using Haskell [Kra14] and one using Python [Bru]. Our function SST_c can be seen as a dependently typed version of such a script, with Agda replaced by an internal CwF and Haskell/Python replaced by Agda. Since Haskell and Python simply produce strings while SST_c is required to type-check, the latter is significantly more difficult to define and requires several new ideas.

Our work has analogies with the construction of semisimplicial types in Voevodsky's *homotopy type system* [Voe13] or *2LTT (2-level type theory)* [ACK16, ACKS19]. Here, Agda plays the role of the outer theory[1] and the internal CwF the role of the inner ("fibrant") one. However, our internal CwF is too minimalistic (e.g. does not contain finite types) to mimic the direct construction of [ACK16]. Moreover, 2LTT allows one to first formulate a type in the outer theory and prove its fibrancy afterwards, a strategy which is not possible in our setting.

It is known that semisimplicial *sets* can be constructed in homotopy type theory, i.e. we can define a function $\mathsf{SST}_0 \colon \mathbb{N} \to \mathsf{Type}_1$ which only considers *sets* of points, *sets* of lines, and so on. Although our CwF's are based on sets, this is unrelated; since our CwF's are not assumed to have an identity type, the notion of truncatedness does not exist for internal types. The fact that Con and $\mathsf{Ty}\,\Gamma$ are sets corresponds to the fact that *judgmental* equality is proof-irrelevant.

Formalization of the Construction. Using the HoTT-Agda library [SH12] we formalize set-based CwF's as records with fields Con, Sub, Ty, Tm, the usual operations thereon, and equations given by their usual presentation as a generalized algebraic theory (see e.g. Fig. 1 of [Kra21]). Since we are motivated by the goal of internalizing constructions in generic homotopy type theory, where many CwF's (such as the formalized syntax proposed by Altenkirch and Kaposi [AK16]) do not satisfy additional definitional equalities, we avoid any use of rewriting pragmas. Our CwF's are further equipped with internal type formers $\hat\Pi$ and $\hat\Sigma$, as well as a family U of base types (polymorphic over contexts Γ) together with decoding function $\mathsf{el} \colon \mathsf{Tm}\,\Gamma\,U \to \mathsf{Ty}\,\Gamma$. We then define $\mathsf{SST}_\mathsf{c}\,0 \coloneqq U$ and $\mathsf{SST}_\mathsf{c}\,(n+1) \coloneqq \big(\mathsf{SST}_\mathsf{c}\,n\,,\ (\mathsf{M}\,n \mathbin{\hat\to} U)\big)$ by mutual induction with the "matching object" $\mathsf{M} \colon (n \colon \mathbb{N}) \to \mathsf{Ty}\,(\mathsf{SST}_\mathsf{c}\,n)$, where $\hat\to$ is the function type in the internal CwF. The main difficulty lies in defining the type $(\mathsf{M}\,n)$ of $\partial\Delta^{n+1}$-shaped tuples indexed over $\mathsf{SST}_\mathsf{c}\,n$.

A high-level description of our approach to this is as follows. We define $\mathsf{M}\,n \coloneqq \mathsf{Sk}\big(n+1, n, \binom{n+2}{n+1}\big)$, where

$$\mathsf{Sk} \colon (b\,h\,t \colon \mathbb{N}) \to \mathsf{Ty}\,(\mathsf{SST}_\mathsf{c}\,h) \quad \text{for } 0 \le h < b,\ 1 \le t \le \tbinom{b+1}{h+1}$$

encodes, as a nested internal $\hat\Sigma$-type $\mathsf{Sk}\,b\,h\,t$, the subfunctor of the representable functor $\Delta_+[b]$[2] which omits all face maps $[i] \to [b]$ for $i > h$, as well as those face maps $[h] \to [b]$ above the t-th (ordered via a bijection $\varphi \colon \mathsf{Fin}\binom{b+1}{h+1} \cong \Delta_+([h],[b])$). The intuition is that $\mathsf{Sk}\,b\,h\,t$ presents the partial h-dimensional boundary of the b-simplex given by "shape" (b, h, t). The point of this is to allow us to define, by induction on h and t,

$$\mathsf{Sk}\,b\,h\,t \coloneqq \hat\Sigma[\sigma \colon \mathsf{Sk}\,b\,h'\,t']\,\big(A_h(\mathsf{inter}\,\sigma\,\varphi(t))\big),$$

where (h', t') is the lexicographic predecessor of (h, t) and $\mathsf{inter}\,\sigma\,f$ picks out the subtuple of σ corresponding to the face f. This intersection function inter is, again, to be constructed by induction on the indices b, h, t. This is work in progress.

[1] However, we do not assume UIP in Agda, as this would make the connection with *type theory eating itself* impossible. The strictness that seems to be needed to construct semisimplicial types is satisfied in our case because the internal CwF is set-based.

[2] $\Delta_+ \coloneqq \Delta$ without degeneracies, i.e. the category of finite non-empty sets and strictly increasing functions.

References

[ACK16] 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.

[ACKS19] 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.

[AK16] Thorsten Altenkirch and Ambrus Kaposi. Type Theory in Type Theory Using Quotient Inductive Types. In *Proceedings of the 43rd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, POPL '16, page 18–29, New York, NY, USA, 2016. Association for Computing Machinery.

[Bru] Guillaume Brunerie. A Python script generating Agda code for semi-simplicial types, truncated at any given level. Available at https://guillaumebrunerie.github.io/other/semisimplicial.py. Accessed 23 Apr 2021.

[Cha09] James Chapman. Type Theory Should Eat Itself. *Electronic Notes in Theoretical Computer Science*, 228:21–36, 2009. Proceedings of the International Workshop on Logical Frameworks and Metalanguages: Theory and Practice (LFMTP 2008).

[Dyb95] 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.

[Her15] Hugo Herbelin. A dependently-typed construction of semi-simplicial types. *Mathematical Structures in Computer Science*, 25(5):1116–1131, 2015.

[Kra14] Nicolai Kraus. A Haskell script to generate the type of n-truncated semi-simplicial types, 2014. Available at https://nicolaikraus.github.io/docs/generateSemiSimp.hs. Accessed 23 Apr 2021.

[Kra21] Nicolai Kraus. Internal ∞-categorical models of dependent type theory: Towards 2LTT eating HoTT, 2021. Available online at https://arxiv.org/abs/2009.01883, to appear in the proceedings of LICS '21.

[SH12] Andrew Swan and the HoTT and UF community. Homotopy type theory, Since 2012. Fork of the original Agda library. Available online at https://github.com/awswan/HoTT-Agda/tree/agda-2.6.1-compatible.

[Shu14] Michael Shulman. Homotopy type theory should eat itself (but so far, it's too big to swallow), 2014. Blog post, https://homotopytypetheory.org/2014/03/03/hott-should-eat-itself.

[Uni13] The Univalent Foundations Program. Semi-simplicial types, 2013. Wiki page of the Univalent Foundations special year at the Institute for Advanced Study, Princeton. Available at https://ncatlab.org/ufias2012/published/Semi-simplicial+types.

[Voe13] Vladimir Voevodsky. A simple type system with two identity types, 2013. Unpublished note available online at https://www.math.ias.edu/vladimir/Lectures.