Computational Cubical Type Theory

Kuen-Bang Hou (Favonia)
favonia@math.ias.edu
Joint work with Carlo Angiuli, Evan Cavallo, Robert Harper and Jonathan Sterling

Homotopy type theory (HoTT) extends Martin-Löf type theory with axioms for univalence and higher inductive types, enabling a significant amount of homotopy theory to be developed synthetically [10]. When rendered as axioms, these extensions disrupt the computational content of type theory, as witnessed by the failure of canonicity: in HoTT, closed terms of natural number type do not necessarily simplify to numerals.

The failure of canonicity has affected the practicality of mechanization. A well-known example is the fourth homotopy group of the 3-sphere, $\pi_4(\mathbb{S}^3)$, which has been shown by Guillaume Brunerie to be $\mathbb{Z}/n\mathbb{Z}$ for some closed term $n : \mathbb{N}$, but more sophisticated arguments were required to prove n is two [4]. If canonicity held, a computer could have simplified n to two automatically.

To date, two univalent type theories enjoy the canonicity property: the cubical type theory of Cohen et al. [6, 8], and our work on computational cubical type theory [2, 3, 5]. Both cubical type theories present higher-dimensional structure using a cubical generalization of typehood and membership judgments, represented syntactically through dependence on dimension variables.

Our contribution comes in two main axes: the application of the computation-first methodology to cubical type theory, and our use of Cartesian cubical structure. In this talk, I will discuss the basics of computational type theory, the extensions to support features of cubical type theory, the differences from other cubical type theories, and the proof assistants.

Computational type theory A computational type theory is not defined by a collection of inference rules, but rather by a programming language and a collection of partial equivalence relations over values of that language. In this style, the computational content of proofs is inevitable; the difficulty lies, instead, in proving that each type former validates the expected formation, introduction, and elimination rules. Computational type theory is therefore a *bottom-up* approach to canonicity, in the sense that computation is prior to typing.

This computation-first approach, pioneered by [7], makes clear how one can potentially integrate homotopy type theory with other programming languages in computer science. We show the extended theory can serve as a model of many variants of homotopy type theory, including two-level type theories with both exact equalities and paths. We also have multiple proof assistants (such as [9]) under active development.

Despite its achievements in modeling a wide range of type theories and its application in making proof assistants, this computation-first approach to type theory has remained largely unknown or been confused with "extensional type theory" or "equality reflection", along with misconceptions on decidability of type-checking. My talk will show unique technical advantages of this less known approach and hopefully inspire new developments in homotopy type theory.

Cartesian cubes Another important contribution is that our type theory employs a *Cartesian* notion of cube: dimension expressions are generated by face, diagonal, and degeneracy (weakening) maps, yielding the free finite-product category generated by two distinct endpoint inclusions $1 \rightrightarrows \mathbb{I}$. In contrast, Cohen et al. [6] use a *De Morgan* notion of cubes that additionally has connections and reversals. Cartesian cubes are appealing because they correspond precisely to a structural notion of dimension variable, subject to exchange, weakening, and contraction.

We equip types with two Kan operations: coercion and homogeneous Kan composition [1–3, 5]. Let A be a type parametrized by a dimension variable x and r, r' be two dimension expressions. Coercion sends elements of type $A\langle r/x\rangle$ to those of type $A\langle r'/x\rangle$, witnessing the continuity of such a parametrized type from $x = r$ to $x = r'$. Homogeneous Kan composition specifies that every open box can be filled. For instance, given the $y = 0$, $x = 0$, and $x = 1$ sides of a x, y-square that agree where they overlap, there is a $y = 1$ side (the *composite*) that agrees with the specified sides and a square that agrees with all four faces. In order to define univalent universes supporting these Kan operations, we crucially allow open boxes consisting not only of faces ($x = 0, 1$) but also diagonals ($x = z$).

Cohen et al. [6] use only one combined Kan operation with forward direction (0 to 1), but with connections and reversals are able to derive other compositions. In comparison, we do not have connections or reversals, but more primitive compositions (forward, backward, filling and more) to complete the construction. It is interesting to see how these technical differences might affect the efficiency of computing programs such as the n in the $\pi_4(\mathbb{S}^3) \simeq \mathbb{Z}/n\mathbb{Z}$.

References

[1] Carlo Angiuli, Guillaume Brunerie, Thierry Coquand, Kuen-Bang Hou (Favonia), Robert Harper and Daniel R. Licata. "Cartesian Cubical Type Theory". Preprint. Dec. 2017. URL: https://github.com/dlicata335/cart-cube.

[2] Carlo Angiuli, Robert Harper and Todd Wilson. "Computational Higher-Dimensional Type Theory". In: *Proceedings of the 44th ACM SIGPLAN Symposium on Principles of Programming Languages*. POPL 2017. Paris, France: ACM, 2017, pp. 680–693. ISBN: 978-1-4503-4660-3. DOI: 10.1145/3009837.3009861.

[3] Carlo Angiuli, Kuen-Bang Hou (Favonia) and Robert Harper. "Computational Higher Type Theory III: Univalent Universes and Exact Equality". Dec. 2017. URL: https://arxiv.org/abs/1712.01800.

[4] Guillaume Brunerie. "On the homotopy groups of spheres in homotopy type theory". PhD thesis. Université Nice Sophia Antipolis, 2016. arXiv: 1606.05916.

[5] Evan Cavallo and Robert Harper. "Computational Higher Type Theory IV: Inductive Types". Jan. 2018. URL: https://arxiv.org/abs/1801.01568.

[6] Cyril Cohen, Thierry Coquand, Simon Huber and Anders Mörtberg. "Cubical Type Theory: A Constructive Interpretation of the Univalence Axiom". In: *21st International Conference on Types for Proofs and Programs (TYPES 2015)*. Vol. 69. Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2018, 5:1–5:34. DOI: 10.4230/LIPIcs.TYPES.2015.5.

[7] Robert L. Constable, et al. *Implementing Mathematics with the Nuprl Proof Development Environment*. Englewood Cliffs, NJ: Prentice-Hall, 1985. ISBN: 978-1-4680-5910-6. URL: http://www.nuprl.org/book/.

[8] Simon Huber. "Cubical Interpretations of Type Theory". PhD thesis. University of Gothenburg, Nov. 2016.

[9] The RedPRL Development Team. *RedPRL – the People's Refinement Logic*. 2018. URL: http://www.redprl.org/.

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