Cumulative universes hierarchies in theory and practice

Raphaël Sterbac[1] and Jonathan Sterling[2]

[1] ENS Paris-Saclay — Université Paris-Saclay, Saclay, France
raphael.sterbac@ens-paris-saclay.fr
[2] Computer Laboratory, University of Cambridge, Cambridge, United Kingdom.
js2878@cl.cam.ac.uk,

In type theory, universes allow to treat types as proper terms and use them as such, while avoiding common paradoxes. Their presentations vary, from implicit systems to more explicit versions with a type of codes U and a decoding family El. Many of the notable discrepancies between different accounts of universe hierarchies come down to the question of *cumulativity*, which asks:

> Given $i \leq j$ and term $\Gamma \vdash M : \mathsf{U}_i$, is there some term $\Gamma \vdash M' : \mathsf{U}_j$ such that $\Gamma \vdash \mathsf{El}_j(M') \equiv \mathsf{El}_i(M)$ *type*?

In systems like Lean and Agda, the hierarchy is explicitly *non*-cumulative. There are technical reasons why non-cumulative hierarchies are prefered, including difficult implementation problems such as the contention between higher-order unification and subtyping coercions. From the semantics side, non-cumulative hierarchies are compatible with a broad range of models, including all Grothendieck ∞-toposes [10], whereas this does not (yet) support cumulativity.

In practice, people often use cumulative universes, especially in Homotopy Type Theory and Univalent foundations Program [9]. We aim at clarifying the state of affairs and to bridge the gap between the theory and implementation of universe hierarchies.

Design of cumulative hierarchies. In a cumulative hierarchy, a new lifting constructor is added that allows codes from one universe to be lifted to a higher one. Then, we need to require this lifting to be "functorial" in the preorder of universe levels, and "natural" in the sense that it should commute with all built-in type constructors [1]. Part of our work substantiate this in a technical sense, working internal to a conservative higher-order extension of the logical framework of second-order algebraic theories [5]. When presenting this system with inference rules, it ends up being quite verbose (here we don't even show all the rules):

$$\frac{i,j : \mathsf{Lvl} \qquad \_ : i \leq j \qquad M : \mathsf{El}(\mathsf{U}_i)}{\mathsf{el}_j\big(\uparrow_i^j M\big) = \mathsf{el}_i(M) : \mathsf{Tp}} \qquad \frac{i : \mathsf{Lvl} \qquad M : \mathsf{El}(\mathsf{U}_i)}{\uparrow_i^i M = M : \mathsf{El}(\mathsf{U}_i)}$$

$$\frac{i,j,k : \mathsf{Lvl} \qquad \_ : i \leq j \qquad \_ : j \leq k \qquad M : \mathsf{El}(\mathsf{U}_i)}{\uparrow_j^k \uparrow_i^j M = \uparrow_i^k M : \mathsf{El}(\mathsf{U}_k)} \qquad \frac{i,j : \mathsf{Lvl} \qquad \_ : i^+ \leq j}{\mathsf{el}_j\big(\mathsf{u}_i^j\big) = \mathsf{U}_i : \mathsf{Tp}}$$

$$\frac{M : \mathsf{El}(\mathsf{U}_i) \qquad N : \mathsf{El}(\mathsf{el}_i(M)) \to \mathsf{El}(\mathsf{U}_i)}{\mathsf{el}_i(\pi_i(M,N)) = \Pi(\mathsf{el}_i(M), \lambda x.\, \mathsf{el}_i(N(x))) : \mathsf{Tp}} \qquad \frac{i,j,k : \mathsf{Lvl} \qquad \_ : i^+ \leq j \qquad \_ : j \leq k}{\uparrow_j^k \mathsf{u}_i^j = \mathsf{u}_i^k : \mathsf{El}(\mathsf{U}_k)}$$

$$\frac{i,j : \mathsf{Lvl} \qquad \_ : i \leq j \qquad M : \mathsf{El}(\mathsf{U}_i) \qquad N : \mathsf{El}(\mathsf{el}_i(M)) \to \mathsf{El}(\mathsf{U}_i)}{\uparrow_i^j (\pi_i(M,N)) = \pi_j\big(\uparrow_i^j M, \lambda x. \uparrow_i^j N(x)\big) : \mathsf{El}(\mathsf{U}_j)}$$

However, we observe that in this setting, we can prove from a normalisation by glueing argument that the decoding maps are monomorphism $\mathsf{el}_i : \mathsf{U}_i \rightarrowtail \mathsf{Tp}$. In light of this, we might revisit the design of the cumulative hierarchy to take this injectivity property as a primitive rule such that the functoriality and naturality rules for the lifting operation become redundant! We can also go further by introducing a *judgemental notion of smallness* that axiomatises the image of the decoding family.

A judgemental notion of smallness. Our proposal is to take the injectivity of decoding seriously, and simply replace U_i with its ***image*** in the class of all types. In type theory, this amounts to defining a judgemental notion of *smallness* $\Gamma \vdash A\ \mathit{small}_i$ which should be closed under the type connectives:

$$\textsc{dfun-small}\ \frac{\Gamma \vdash A\ \mathit{small}_i \qquad \Gamma, x : A \vdash B[x]\ \mathit{small}_i}{\Gamma \vdash (x : A) \to B[x]\ \mathit{small}_i} \qquad \textsc{univ-small}\ \frac{\Gamma \vdash i < j}{\Gamma \vdash \mathsf{U}_i\ \mathit{small}_j} \qquad \textsc{small-mono}\ \frac{\Gamma \vdash i \leq j \qquad \Gamma \vdash A\ \mathit{small}_i}{\Gamma \vdash A\ \mathit{small}_j}$$

$$\textsc{univ-intro}\ \frac{\Gamma \vdash A\ \mathit{small}_i}{\Gamma \vdash \Downarrow_i A : \mathsf{U}_i} \qquad \textsc{univ-elim}\ \frac{\Gamma \vdash M : \mathsf{U}_i}{\Gamma \vdash \Uparrow_i M\ \mathit{type}, \Uparrow_i M\ \mathit{small}} \qquad \textsc{univ-beta}\ \frac{\Gamma \vdash A\ \mathit{small}_i}{\Gamma \vdash \Uparrow_i \Downarrow_i A \equiv A\ \mathit{type}} \qquad \textsc{univ-ext}\ \frac{\Gamma \vdash \Uparrow_i M \equiv \Uparrow_i N\ \mathit{type}}{\Gamma \vdash M \equiv N : \mathsf{U}_i}$$

Smallness is here formulated as a "synthetic" judgement in the sense of Martin-Löf [8]. We could (and later shall) equivalently formulate smallness analytically as $p :: A\ \mathit{small}_i$ with subject p ranging over smallness witnesses, with an additional rule asserting $p \equiv q :: A\ \mathit{small}_i$ for all p, q. This shows that our formulation can still be presented in an *algebraic* way. In fact, our rules for universes are parametric in *any* notion of smallness.

Syntactic equivalence of the smallness calculus. We show that the simplified calculus with judgemental smallness is *equivalent* to the more complicated natural hierarchy. This has consequences for both theory and practice:

1. **Theory**: We can interpret the very convenient type theory with judgemental smallness into semantic models of cumulative hierarchies *without* requiring the model to satisfy injectivity laws, as these can first be eliminated our syntactic equivalence.

2. **Practice**: The smallness calculus offers many simplifications for practical implementation, because many equational rules are replaced by a very small collection of easily programmed rules. Our proposal is to elaborate to this algebraic core calculus, as would be needed in order to justify the adequacy of the formalism as a tool for proving things in standard mathematics. This has been effectively implemented in a practical system [11].

Additionally, the smallness calculus admits a particularly simple account of inductive datatypes, including large ones. Our methodology presents a judgemental presentation of cumulative data type descriptions, following the work of Dagand [4].

There are still open implementation problems, related to unification and scheduling with the coercion mechanism. It alls stems to the following fact : solving $\alpha \equiv A$ by taking $\alpha = A$ is the least general solution for our coercion system. Since the problem is *postponement*, we believe that extending the algorithm of Kovács [7] to the coercions induced by cumulativity should improve the behaviour of the system, but this has to be tested in a practical implementation.

Related work. Our proposal is similar to that of Coquand [2], except that we retain a top-level notion of "large" type, and we retain the judgemental/syntactic distinction between types and terms. We note that several others have given a directly structural account of "universes à la Coquand" without introducing a top level of types; see for example the work of Gratzer et al. [6]. Our presentation is also quite similar to the one in Coquand et al. [3].

References

[1] M. Bezem, T. Coquand, P. Dybjer, and M. Escardó. Type Theory with Explicit Universe Polymorphism. In D. Kesner and P.-M. Pédrot, editors, *28th International Conference on Types for Proofs and Programs (TYPES 2022)*, volume 269 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 13:1–13:16, Dagstuhl, Germany, 2023. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. ISBN 978-3-95977-285-3. doi: 10.4230/LIPIcs.TYPES.2022.13. URL https://drops.dagstuhl.de/entities/document/10.4230/LIPIcs.TYPES.2022.13.

[2] T. Coquand. Presheaf model of type theory. Unpublished manuscript, 2012. URL http://www.cse.chalmers.se/~coquand/presheaf.pdf.

[3] T. Coquand, B. Mannaa, and F. Ruch. Stack semantics of type theory. In *Logic in Computer Science (LICS)*, 2017. doi: 10.1109/LICS.2017.8005130.

[4] P.-E. Dagand. *Reusability and Dependent Types*. PhD thesis, University of Strathclyde, 2013. URL https://www.irif.fr/~dagand/stuffs/thesis-2011-phd/thesis.pdf.

[5] D. Gratzer and J. Sterling. Syntactic categories for dependent type theory: sketching and adequacy. Unpublished manuscript, 2020.

[6] D. Gratzer, G. A. Kavvos, A. Nuyts, and L. Birkedal. Multimodal dependent type theory. *Logical Methods in Computer Science*, Volume 17, Issue 3:11, Jul 2021. ISSN 1860-5974. doi: 10.46298/lmcs-17(3:11)2021. URL https://lmcs.episciences.org/7571.

[7] A. Kovács. Observing definitional equality. Talk at WITS 2026.

[8] P. Martin-Löf. Analytic and synthetic judgements in type theory. In P. Parrini, editor, *Kant and Contemporary Epistemology*, pages 87–99. Springer Netherlands, Dordrecht, 1994. ISBN 978-94-011-0834-8.

[9] T. U. F. Program. *Homotopy Type Theory: Univalent Foundations of Mathematics*. Institute for Advanced Study, 2013.

[10] M. Shulman. All $(\infty, 1)$-toposes have strict univalent universes. Unpublished manuscript, Apr. 2019.

[11] R. Sterbac. Implementation of the smallness calculus, 2026. URL https://github.com/raphael-sterbac/elaboration-universes.