The Case for Impredicative Universe Polymorphism

Stefan Monnier[1]

Université de Montréal - DIRO, Montréal, Canada
monnier@iro.umontreal.ca

Abstract

Predicativity tells us that universe polymorphic definitions should live in the universe of level ω. We argue for an alternative rule we call IUP (impredicative universe polymorphism) which allows such definitions to live in a much lower universe. The conditions under which this rule leads to inconsistencies are not yet known, so in the mean time we present different use cases to provide some intuition about its possibilities and its limits.

1 Introduction

The predicative tower of universes is nowadays ubiquitous in proof assistants, and more often than not it comes equipped with some form of universe polymorphism [2]. This can take various forms, from a "simple" sort of implicit meta-level macro-expansion, through *displacement algebras* [5], to something like Agda where universe levels can be manipulated explicitly and universe polymorphic definitions can be manipulated as first-class values [10], or even systems where universe levels are themselves first-class values [8].

In all these works, universe polymorphism is predicative, meaning that universe polymorphic definitions are placed in a universe of level strictly above the largest level to which it can be instantiated. Concretely, this typically means that such definitions are placed in the universe of level ω.

We argue that, under some conditions, we could place universe polymorphic definitions in a lower universe, making those definitions impredicative. More specifically, for a type τ of universe level ℓ, which can depend on a universe level variable l, the accepted predicative rule places the type $\forall l.\tau$ in the universe of level $\mathsf{sup}_l\ \ell$, i.e. ω, whereas we argue that, under some conditions, we could place it in the universe $\mathsf{inf}_l\ \ell$, which in our case is also $\ell[0/l]$.

In terms of practical uses, this form of universe polymorphism would be convenient to manipulate universe-polymorphic terms without resorting to universe levels beyond ω, it could be used as an alternative form of impredicativity which does not require a special impredicative universe, and it appears to be one of the ingredients necessary for type-preserving closure conversion of languages with a hierarchy of universes.

We do not know under which conditions we can use such a rule without breaking consistency of the system, and it seems difficult to relate this form of impredicativity we call IUP [9, Sec 4] to other known forms of impredicativity such the Prop universe of the Calculus of Constructions [4] or the propositional resizing axiom of HoTT [11, Sec 3.5], so until the underlying theory is better understood, and as a way to motivate such investigation, we present a few specific use cases where we have reasons to think they should be accepted or not.

2 Case Studies

We explore a few use cases, chosen for their ability to provide some intuition about the possibilities and the limits of IUP.

2.1 Strong sums

One clear limit to the consistency of IUP comes if we apply it to strong sums, as in:

$$\frac{\Gamma, l : \mathsf{Level} \vdash \tau \ : \ \mathsf{Type}_\ell}{\Gamma \vdash \Sigma l.\tau \ : \ \mathsf{Type}_{\ell[0/l]}}$$

Since we can then resize any function down to universe Type_1 by stuffing it in an object of type $\Sigma l_1.\Sigma t_1 : \mathsf{Type}_{l_1}.\Sigma l_2.\Sigma t_2 : \mathsf{Type}_{l_2}.t_1 \to t_2$ out of which we can easily recover the original function, regardless of its original universe level. This also suggests that IUP is incompatible with first-class universe levels.

2.2 Church encoding

Functions that correspond to Church encodings of inductive types are examples where the IUP rule seems to be sound. Consider a type of the form:

$$\forall l.\forall t : \mathsf{Type}_l.t \to (\tau \to t \to t) \to t$$

Assuming that τ do not refer to l nor t, this type corresponds to a Church-style encoding of a list of elements of type τ. If τ is in universe level ℓ, then the corresponding inductive type would be placed in universe level ℓ, but Agda would place it in ω. The IUP rule would instead put the above type in universe $1 \sqcup \ell$. Compared to the original Church encoding, this adds support for strong elimination, i.e. the possibility to eliminate to any universe rather than only to the bottom impredicative universe (i.e. Prop or Set), which is allowed for inductive types and thus suggests the IUP rule should be sound in this case.

More generally, we can consider a type of the following form:

$$\forall l.\forall t : \mathsf{Type}_l.(\vec{\tau_1} \to t) \to ... \to (\vec{\tau_n} \to t) \to t$$

If we assume that the τ_{ij} types do not refer to l, this type corresponds to a Church-style encoding of an inductive type. If each τ_{ij} is in universe level ℓ_{ij}, then the corresponding inductive type would be placed in universe level $\bigsqcup_{i,j} \ell_{ij}$, while our IUP rule would similarly put the above type in universe $1 \sqcup \bigsqcup_{i,j} \ell_{ij}$.
Note that compared to actual inductive types, Church-style encodings using our IUP rule still lack dependent elimination, which is a problem that has been tackled by Awodey et.al. [1] and Firsov and Stump [6]. Of mention also would be the work by Jenkins et.al. [7] which tries to support strong elimination but without a rule like our IUP rule.

2.3 Closures

In type-preserving compilers for functional programming languages, functions end up reified as tuples that are then wrapped in an existential package so as not to expose the set of captured variables in its type. When captured variables can come from an arbitrary universe level, we similarly need to hide that level: For a source function of type $\tau_1 \to \tau_2$, the closure needs to

have a type of the form $\exists l.\exists t : \mathsf{Type}_l.t \times ((\tau_1 \times t) \to \tau_2)$ where t is the type of the captured environment. If those $\exists$ are weak sums, one cannot do anything with such closures that one cannot already do with its corresponding function of type $\tau_1 \to \tau_2$, so it supports the idea that such existentials could live in the same universe level $l_1 \sqcup l_2$, and thus suggests that our IUP rule is justified in this case to puts this existential type in universe level $1 \sqcup l_1 \sqcup l_2$.
See Bowman and Ahmed [3] for more challenges with type preserving closure conversion.

2.4 Well-ordering

Girard's paradox starts with an impredicative definition of an ordering and then exhibits an ordering of such orderings. The ordering is a tuple of a type along with some properties:

$$\mathsf{type}\ \mathit{Ordering} : \mathsf{Type} = \mathit{mk\text{-}ord}\ (\mathit{set} : \mathsf{Type})\ (\mathit{less\text{-}than} : \mathit{set} \to \mathit{set} \to \mathsf{Type})\ ...$$

In a predicative setting this does not work because *Ordering* ends up in a higher universe level than *set*. If we want to try and reproduce the paradox using IUP, we need to abstract over the universe level of *set*:

$$\mathsf{type}\ \mathit{Ordering} : \mathsf{Type}_1 = \mathit{mk\text{-}ord}\ (l : \mathsf{Level})\ (\mathit{set} : \mathsf{Type}_l)\ (\mathit{less\text{-}than} : \mathit{set} \to \mathit{set} \to \mathsf{Type}_l)\ ...$$

But since IUP is restricted to apply to weak sums only, the above definition is unusable, because the *set* cannot be extracted from that tuple any more, making it difficult to compare two such orderings to make an ordering of orderings.

Acknowledgments

This work was supported by the Natural Sciences and Engineering Research Council of Canada (NSERC) grant N^o RGPIN-2018-06225. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author and do not necessarily reflect the views of the NSERC.

References

[1] Steve Awodey, Jonas Frey, and Sam Speight. Impredicative encodings of (higher) inductive types. In *Annual Symposium on Logic in Computer Science*, page 17, 2018. doi:10.1145/3209108.3209130.

[2] Marc Bezem, Thierry Coquand, Peter Dybjer, and Martín Escardó. Type theory with explicit universe polymorphism. In *Types for Proofs and Programs*, Leibniz International Proceedings in Informatics (LIPIcs), 2022. doi:10.4230/LIPIcs.TYPES.2022.13.

[3] William J. Bowman and Amal Ahmed. Typed closure conversion for the calculus of constructions. In *Programming Languages Design and Implementation*, page 797–811, 2018. doi:10.1145/3192366.3192372.

[4] Thierry Coquand and Gérard P. Huet. The calculus of constructions. Technical Report RR-0530, INRIA, 1986.

[5] Kuen-Bang Hou (Favonia), Carlo Angiuli, and Reed Mullanix. An order-theoretic analysis of universe polymorphism. In *Principles of Programming Languages*, pages 1659–1685. ACM Press, 2023. doi:10.1145/3571250.

[6] Denis Firsov and Aaron Stump. Generic derivation of induction for impredicative encodings in Cedille. In *Certified Programs and Proofs*, pages 215–227, 2018. doi:10.1145/3167087.

[7] Christa Jenkins, Andrew Marmaduke, and Aaron Stump. Simulating large eliminations in Cedille. In *Types for Proofs and Programs*, Leibniz International Proceedings in Informatics (LIPIcs), page 22, 2021. doi:10.4230/LIPIcs.TYPES.2021.9.

[8] András Kovács. Generalized universe hierarchies and first-class universe levels. In *Computer Science Logic*, pages 28:1–28:17, 2022. doi:10.4230/LIPIcs.CSL.2022.28.

[9] Stefan Monnier and Nathaniel Bos. Is impredicativity implicitly implicit? In *Types for Proofs and Programs*, Leibniz International Proceedings in Informatics (LIPIcs), pages 9:1–9:19, 2019. doi:10.4230/LIPIcs.TYPES.2019.9.

[10] The Agda team. The Agda manual. URL: https://agda.readthedocs.io/en/v2.6.2.1/.

[11] The Univalent Foundations Program. *Homotopy Type Theory: Univalent Foundations of Mathematics*. Institute for Advanced Study, 2013. URL: https://arxiv.org/abs/1308.0729.