There is Life in the Universes Beyond ω

ANONYMOUS AUTHOR(S)

The first draft of Martin-Löf's type theory proposed the assumption Type:Type. Subsequently, universe levels have been introduced to avoid the resulting inconsistencies by assuming Type_i:Type_{i+1}. Proof assistants based on type theory support such universe levels to varying degree, but they impose restrictions that can make coding awkward.

Specifically, we consider the ramifications of Agda's approach to handling levels using a denotational semantics of a stratified version of System F as a motivating example. We propose a simple fix that extends Agda's capabilities for handling universe levels parametrically up to ε_0.

Additional Key Words and Phrases: Dependent types, universes, ordinal numbers

ACM Reference Format:
Anonymous Author(s). 2018. There is Life in the Universes Beyond ω. In *Proceedings of Make sure to enter the correct conference title from your rights confirmation email (Conference acronym 'XX).* ACM, New York, NY, USA, 4 pages. https://doi.org/XXXXXXX.XXXXXXX

1 Introduction

The origin of universe levels.

How do universe levels work in Agda?

What becomes awkward with Agda's approach?

How do we propose to fix it?

Contributions.

2 Preliminaries

Agda, Ordinals, IR-Universes

2.1 Universes in Agda

TODO: typeset in Agda style

Agda contains significant support for an infinite hierarchy of universes [10]. It provides an abstract datatype LEVEL of universe levels along with constants ZERO, SUC, and MAX that denote the base-level of the hierarchy, the successor, and the maximum of two levels. The level-parametric type $Set\ i$ stands for the universe at level i and it obeys the typing SET i : SET (SUC i) Base types inhabit the universe $Set\ zero$. Type formation handles universe levels in the same way as finitely stratified System F [7]. That is,

- if $A_1 : Set\ i_1$ and $A_2 : Set\ i_2$, then $A_1 \to A_2 : Set(i_1 \sqcup i_2)$;
- if $\alpha : Set\ i$ is a type variable and $A : Set\ j$, then $\forall\alpha.A : Set(suc\ i \sqcup j)$.

To avoid inconsistencies, Agda does not allow pattern matching on the type LEVEL. However, quantification over LEVEL is allowed and results in a type at level ω (if the level-typed variable is used in a significant way). Unfortunately, levels ω and higher are **not** handled in a parametric way in Agda. Rather there are predefined types Set_ω, $Set_{\omega 1}$, $Set_{\omega 2}$, and so on, for the levels ω, $\omega + 1$, $\omega + 2$. This design has some limitations, as most notions in the standard library (prominently the equality type and its supporting infrastructure, but also parameteric types like sums, pairs, lists, and so on) are defined in a level-parametric way, and thus they cannot be used with SET$_\omega$ and higher universe levels.

For convenience, Agda's unification algorithm has special treatment for the LEVEL datatype and its operators [10]:

- Idempotence: $a \sqcup a \equiv a$.
- Associativity: $(a \sqcup b) \sqcup c$ is the same as $a \sqcup (b \sqcup c)$.
- Commutativity: $a \sqcup b$ is the same as $b \sqcup a$.
- Distributivity of SUC over MAX: $suc(a \sqcup b)$ is the same as $suc\ a \sqcup suc\ b$.
- Neutrality of ZERO: $a \sqcup zero$ is the same as a.
- Subsumption: $a \sqcup suc\ a$ is the same as $suc\ a$.

In other words, the level structure is a join-semilattice with an inflationary endomorphism [1].

TODO: Cumulativity

2.2 Ordinals

Ordinal numbers are an important concept in mathematics and computer science. They are closely related to well-ordered sets as any well-ordered set is order-isomorphic to an ordinal. The significance to computer science is that such well-orders can be used for termination proofs.

The best known construction of ordinals is a set-theoretic one due to von Neumann. It starts with the smallest ordinal 0 represented by the empty set $\emptyset$. To construct the successor of an ordinal α, we define $\alpha + 1 := \alpha \cup \{\alpha\}$. This way, we construct $1, 2, 3, \ldots$. Then, we can scoop them all up into the smallest limit ordinal $\omega = \{0, 1, 2, 3, \ldots\}$, which contains 0 and all finite applications of the successor to it. We continue with $\omega + 1 = \omega \cup \{\omega\}$ and carry on until we build the next limit ordinal $\omega \cdot 2$, then $\omega \cdot 3$, and so on. Constructing the limit at this level yields ω^ω and continuing further leads to ω^{ω^ω}, $\omega^{\omega^{\omega^\omega}}$, and so on. The limit of this sequence is ε_0 which is the smallest ordinal that fulfills the equation $\varepsilon_0 = \omega^{\varepsilon_0}$.

In the context of this paper, we plan to use ordinals as universe levels where the well-ordering avoids collapsing levels. The construction of ordinals does not stop at ε_0, but we wish to carve out a particular set of ordinals that fits well in an implementation context. Concretely, we consider ordinals less than ε_0 as they can be represented by binary trees. To see this, recall that every ordinal α can be written in Cantor normal form

$$\alpha = \omega^{\beta_1} + \omega^{\beta_2} + \cdots + \omega^{\beta_n}$$

for some $n \geq 0$ and ordinals β_i such that $\beta_1 \geq \beta_2 \geq \cdots \geq \beta_n$. If $\alpha < \varepsilon_0$, then it can be shown that each exponent satisfies $\beta_i < \alpha$. If we, again, write $\beta_i = \omega^{\gamma_1} + \cdots + \omega^{\gamma_m}$ in Cantor normal form, then clearly $\gamma_j < \beta_i < \alpha$. As the ordering on ordinals is a well-order, we know this decreasing sequence must terminate and we obtain a finite representation for each ordinal less than ε_0.

Forsberg et al. [4] developed Agda formalizations for three equivalent representations for precisely this set of ordinals. They define addition and multiplication of ordinals, prove the principle of transfinite induction, and use that to prove that the represented subset of ordinals is well-ordered.

We build on one of their representations, called MutualOrd, define some additional operations, and prove some properties which are needed in the context of universe levels.[1] In particular, we prove the laws of a join-semilattice with an inflationary endomorphism.

TODO: Insert

- Definition of MutualOrd
- Explanation
- Example(s)
- any additional properties that we had to prove

2.3 Encoding Universes

While Agda has built-in support for universes, it is possible to "compress" very general universe hierarchies into *Set*. Kovács [6] constructs a very general model of cumulative universe hierarchies, which situated entirely in Set_0 and Set_1. The construction requires induction-recursion [3], so that the model can be constructed in Agda along with its consistency proof.

Kovacs's construction is implemented in Agda along the lines of McBride [8]'s universe construction. Here we just recall the most important definitions of his construction of TTDL (type theory with dependent levels).

A *level structure* provides a set Lvl of levels, a strict linear order <, evidence that this order is propositional, and a composition operator that embodies transitivity of the order.

```
Lvl    : Set
_<_    : Lvl → Lvl → Set
<-prop : ∀ {i j}{p q : i < j} → p ≡ q
_∘_    : ∀ {i j k} → j < k → i < j → i < k
```

The inductive type U^{ir} of universe codes is defined mutually recursively with its interpretation El^{ir} in Set.

```
data U^ir {i}(l : ∀ (j : Lvl) → j < i → Set) : Set

U'   : ∀ {j} → j < i → U^ir l
ℕ'   : U^ir l
⊤'   : U^ir l
Π'   : (a : U^ir l) → (El^ir a → U^ir l) → U^ir l
Lvl' : U^ir l
_<'_ : Lvl → Lvl → U^ir l
```

```
El^ir : ∀ {i l} → U^ir {i} l → Set

El^ir {_}{l}(U' p) = l _ p
El^ir ℕ'          = ℕ
El^ir ⊤'          = ⊤
El^ir (Π' a b)    = ∀ x → El^ir (b x)
El^ir Lvl'        = Lvl
El^ir (i <' j)    = i < j
```

3 Running Example

As a running example for demonstrating various encodings, we consider different extensions of finitely stratified System F [7]. More precisely, we start from an intrinsically typed encoding of types and expressions, then we construct denotational semantics for different encodings and discuss their respective merits.

[1]Technically, the mechanization of Forsberg et al. [4] relies on cubical Agda [?]. We back-ported the definitions for MutualOrd to standard Agda, as cubical is not needed for this representation.

We choose this system as it is significant, presents non-trivial challenges, and it has been studied in the literature. Our encoding of syntax is inspired by Chapman et al. [2], who develop the syntactic metatheory of System-Fω (without stratification). It has been picked up by Saffrich et al. [9], who give denotational and operational semantics for finitely stratified System-F and develop a logical relation for it. Hubers and Morris [5] use a similar syntax representation for a finitely stratified version of System-Fω extended with qualified types. They also develop a denotational semantics for their calculus. All these papers come with Agda formalizations.

4 Constructions

5 Related Work

How do other proof assistants (Coq, Lean) handle universes? Cumulativity, Impact of impredicativity

6 Conclusions

References

[1] Marc Bezem and Thierry Coquand. 2022. Loop-checking and the uniform word problem for join-semilattices with an inflationary endomorphism. *Theor. Comput. Sci.* 913 (2022), 1–7. doi:10.1016/J.TCS.2022.01.017

[2] James Chapman, Roman Kireev, Chad Nester, and Philip Wadler. 2019. System F in Agda, for Fun and Profit. In *Mathematics of Program Construction - 13th International Conference, MPC 2019 (Lecture Notes in Computer Science, Vol. 11825)*, Graham Hutton (Ed.). Springer, Porto, Portugal, 255–297. doi:10.1007/978-3-030-33636-3_10

[3] Peter Dybjer and Anton Setzer. 1999. A Finite Axiomatization of Inductive-Recursive Definitions. In *Typed Lambda Calculi and Applications, 4th International Conference, TLCA'99, L'Aquila, Italy, April 7-9, 1999, Proceedings (Lecture Notes in Computer Science, Vol. 1581)*, Jean-Yves Girard (Ed.). Springer, 129–146. doi:10.1007/3-540-48959-2_11

[4] Fredrik Nordvall Forsberg, Chuangjie Xu, and Neil Ghani. 2020. Three equivalent ordinal notation systems in cubical Agda. In *Proceedings of the 9th ACM SIGPLAN International Conference on Certified Programs and Proofs, CPP 2020, New Orleans, LA, USA, January 20-21, 2020*, Jasmin Blanchette and Catalin Hritcu (Eds.). ACM, 172–185. doi:10.1145/3372885.3373835

[5] Alex Hubers and J. Garrett Morris. 2023. Generic Programming with Extensible Data Types: Or, Making Ad Hoc Extensible Data Types Less Ad Hoc. *Proc. ACM Program. Lang.* 7, ICFP (2023), 356–384. doi:10.1145/3607843

[6] András Kovács. 2022. Generalized Universe Hierarchies and First-Class Universe Levels. In *30th EACSL Annual Conference on Computer Science Logic, CSL 2022, February 14-19, 2022, Göttingen, Germany (Virtual Conference) (LIPIcs, Vol. 216)*, Florin Manea and Alex Simpson (Eds.). Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 28:1–28:17. doi:10.4230/LIPICS.CSL.2022.28

[7] Daniel Leivant. 1991. Finitely Stratified Polymorphism. *Inf. Comput.* 93, 1 (1991), 93–113. doi:10.1016/0890-5401(91)90053-5

[8] Conor McBride. 2015. Datatypes of Datatypes. https://www.cs.ox.ac.uk/projects/utgp/school/conor.pdf.

[9] Hannes Saffrich, Peter Thiemann, and Marius Weidner. 2024. Intrinsically Typed Syntax, a Logical Relation, and the Scourge of the Transfer Lemma. In *Proceedings of the 9th ACM SIGPLAN International Workshop on Type-Driven Development, TyDe 2024, Milan, Italy, 6 September 2024*, Sandra Alves and Jesper Cockx (Eds.). ACM, Milan, Italy, 2–15. doi:10.1145/3678000.3678201

[10] The Agda Team. 2025. Agda Language Reference: Universe Levels. https://agda.readthedocs.io/en/stable/language/universe-levels.html.