Universes In Type Theory

András Kovács[a]

January 29, 2021, PLC Department Workshop

Eötvös Loránd University, Budapest

[a]The author was supported by the European Union, co-financed by the European Social Fund (EFOP-3.6.3-VEKOP-16-2017-00002).

Russell's paradox, or size issues

There is no set S in a consistent set theory such that $x \in S$ iff $x \notin x$.

Alternate formulations:

- Barber's paradox
- No set of all sets

Related:

- Real numbers are uncountable
- Gödel's first incompleteness theorem
- Undecidability of halting

(All instances of Lawvere's fixed point theorem)

Solution with universes

Solving Russell's paradox: set comprehension can only define a **subset** of a set.

But sometimes we still want to quantify over **all sets** of Zermelo-Fraenkel set theory.

We can assume the existence of a very large set such that all of ZF can fit there, which is called a **universe**. Then, just quantify over elements of the universe. The resulting theory is stronger than ZF.

If we want quantify over sets which **don't** fit (or universes themselves), we can just assume even more universes.

Universes in type theory

Agda:

```
Bool          : Set₀
Set₀          : Set₁
Set₁          : Set₂
(Bool → Set₀) : Set₁
(Set₀ → Bool) : Set₁
```

Assuming `Set₀ : Set₀` implies contradiction (by Russell-like argument).

(But type theory with `Set₀ : Set₀` is still a perfectly good Turing-complete programming language)

Universes in type theory

The basic system can be tedious:

```
id₀ : (A : Set₀) → A → A
id₁ : (A : Set₁) → A → A
...
```

In Coq/Agda/Lean, various extra features are used. Example: universe polymorphism in Agda:

```
id : (l : Level)(A : Set l) → A → A
id l A x = x
```

Design choices and variations

How many universes? Agda/Coq: countably many.

Are universes totally ordered? Agda/Coq: yes.

What kind of level polymorphism?. Coq: bounded polymorphism. Agda: no bounds allowed. Bounded example:

```
myId : (l : Level) → l < 3 → (A : Set l) → A → A
myId l p A x = x
```

What kind of operations are available on levels? Agda is more liberal than Coq. Example:

```
ℕtoLevel : ℕ → Level
```

Are universes cumulative Agda: no. Coq: yes. Cumulativity: whenever `A : Set i`, we also have `A : Set (i + 1)`.

Research goals

We want to know that each point in the design space makes sense.

Making sense:

- Logical consistency.
- Is the type theory a proper programming language? Programs should compute to values and not get randomly stuck.
- Is type checking decidable? (I don't plan to cover this in research).

Approach: use generic framework to cover as many features/variations as possible. Prove that everything in the framework is sensible.

Results

Framework covers:

1. Universe levels may come from any well-ordered set.
2. Polymorphism with bounds is allowed.
3. Universes are cumulative.
4. Levels can be manipulated arbitrarily as program data (WIP).

Things we need to assume in order to prove the framework sensible:

1. **Two** universes.
2. Types of certain infinitely branching trees (W-types).

Potential further applications (in future work):

- Information flow type systems ("secure" and "public" levels).
- Staged compilation ("runtime" and "compile time" levels).

This talk is a side project intended for FSCD 2021 submission (in February).

Other 2020 publications:

- LICS 2020, with Ambrus: “Large and Infinitary Quotient Inductive-Inductive Types”
- ICFP 2020: “Elaboration with First-Class Implicit Function Types”
- TYPES 2019 post-proceedings, with Ambrus and Ambroise Lafont: “For Finitary Induction-Induction, Induction is Enough”

Thank you!