Cumulative universes in theory and practice

Raphaël Sterbac, ENS Paris-Saclay
Joint work with Jon Sterling

HoTT/UF 2026, Aarhus, Denmark (02/06/2026)

Outline

1. Introduction
2. Natural universes and The Smallness calculus
3. Equivalence and Normalisation
4. Easy Cumulative Inductive types
5. Conclusion

Introduction

Implementing type theory

- You want to do synthetic mathematics

Implementing type theory

- You want to do synthetic mathematics
- ... on a computer, using a proof assistant

Implementing type theory

- You want to do synthetic mathematics
- ... on a computer, using a proof assistant
- For different usages, a different tool : Cubical Agda, Rzk...

Implementing type theory

- You want to do synthetic mathematics
- ... on a computer, using a proof assistant
- For different usages, a different tool : Cubical Agda, Rzk...
- We need more and more implementations of type theory, that are often relying on complex higher structures

Painful step : Universes

- If you are implementing a type theory, you will inevitably have to implement a universe hierarchy

Painful step : Universes

- If you are implementing a type theory, you will inevitably have to implement a universe hierarchy
  - $\rightarrow$ Universes turn types into terms such that types can be used by functions or embedded in inductives

Painful step : Universes

- If you are implementing a type theory, you will inevitably have to implement a universe hierarchy
  - $\rightarrow$ Universes turn types into terms such that types can be used by functions or embedded in inductives
  - $\rightarrow$ For instance, you need universes to even *state* univalence.

Painful step : Universes

- If you are implementing a type theory, you will inevitably have to implement a universe hierarchy
  - $\rightarrow$ Universes turn types into terms such that types can be used by functions or embedded in inductives
  - $\rightarrow$ For instance, you need universes to even *state* univalence.
  - $\rightarrow$ In HoTT, they are often assumed to be cumulative

Painful step : Universes

- If you are implementing a type theory, you will inevitably have to implement a universe hierarchy
  - → Universes turn types into terms such that types can be used by functions or embedded in inductives
  - → For instance, you need universes to even *state* univalence.
  - → In HoTT, they are often assumed to be cumulative
- But universes are often painful to implement, and lead to many difficult design choices

Russell and Tarski

- For implementation, you often want to use implicit (Russell) universes

Russell and Tarski

- For implementation, you often want to use implicit (Russell) universes
- But for mathematics, the useful presentation is explicit (Tarski) universes

Russell and Tarski

- For implementation, you often want to use implicit (Russell) universes
- But for mathematics, the useful presentation is explicit (Tarski) universes
  - $\rightarrow$ They have interpretation in all Grothendieck toposes ! (Gratzer, Shulman, Sterling 2024)

Russell and Tarski

- For implementation, you often want to use implicit (Russell) universes
- But for mathematics, the useful presentation is explicit (Tarski) universes
  - $\rightarrow$ They have interpretation in all Grothendieck toposes ! (Gratzer, Shulman, Sterling 2024)
  - $\rightarrow$ Even in all ∞-toposes without cumulativity (Shulman 2019)

Russell and Tarski

- For implementation, you often want to use implicit (Russell) universes
- But for mathematics, the useful presentation is explicit (Tarski) universes
  - $\rightarrow$ They have interpretation in all Grothendieck toposes ! (Gratzer, Shulman, Sterling 2024)
  - $\rightarrow$ Even in all ∞-toposes without cumulativity (Shulman 2019)
- Ultimately, those two presentations should be equivalent, and thus have the same models (See my talk at TYPES'26 !)

Russell and Tarski

- For implementation, you often want to use implicit (Russell) universes
- But for mathematics, the useful presentation is explicit (Tarski) universes
  - $\rightarrow$ They have interpretation in all Grothendieck toposes ! (Gratzer, Shulman, Sterling 2024)
  - $\rightarrow$ Even in all ∞-toposes without cumulativity (Shulman 2019)
- Ultimately, those two presentations should be equivalent, and thus have the same models (See my talk at TYPES'26 !)
- Maybe this is not really about *explicit* and *implicit* syntax...

Codes and injectivity

- The intrinsic difference is the injectivity of the decoding map:

$$\mathsf{El}_i(a) = \mathsf{El}_i(b) \implies a = b : \mathsf{U}_i$$

Codes and injectivity

- The intrinsic difference is the injectivity of the decoding map:

$$\mathsf{El}_i(a) = \mathsf{El}_i(b) \implies a = b : \mathsf{U}_i$$

- Injectivity is a bit weird from a semantics perspective...

Codes and injectivity

- The intrinsic difference is the injectivity of the decoding map:

$$\mathsf{El}_i(a) = \mathsf{El}_i(b) \implies a = b : \mathsf{U}_i$$

- Injectivity is a bit weird from a semantics perspective...
  - $\rightarrow$ This looks a bit like a "definitional univalence"
- Yet, injectivity plays an essential role with the realignment property
  - $\rightarrow$ *Chose* coherent codes if the lifts $\uparrow_i^j : \pi_i \rightarrowtail \pi_j$ are monos

A new presentation

- Our proposal : take injectivity seriously, but still have an *algebraic* presentation

A new presentation

- Our proposal : take injectivity seriously, but still have an *algebraic* presentation
- Replace U_i with its image in the sort of types, leading to a *judgmental* notion of smallness

A new presentation

- Our proposal : take injectivity seriously, but still have an *algebraic* presentation
- Replace U_i with its image in the sort of types, leading to a *judgmental* notion of smallness

$\rightarrow$ Structural account of "universes à la Coquand"

Contributions

Our contributions are the following:

Contributions

Our contributions are the following:

- We provide a simpler presentation for algebraic universes
  - $\rightarrow$ Semantically compatible with a wide range of models

Contributions

Our contributions are the following:

- We provide a simpler presentation for algebraic universes
  - $\rightarrow$ Semantically compatible with a wide range of models
- Equivalence with the usual *natural* universe hierarchies
  - $\rightarrow$ Normalisation for a theory with **internal level judgements**

Contributions

Our contributions are the following:

- We provide a simpler presentation for algebraic universes
  - $\rightarrow$ Semantically compatible with a wide range of models
- Equivalence with the usual *natural* universe hierarchies
  - $\rightarrow$ Normalisation for a theory with **internal level judgements**
- Simpler account of cumulative inductive types

Contributions

Our contributions are the following:

- We provide a simpler presentation for algebraic universes
  - $\rightarrow$ Semantically compatible with a wide range of models
- Equivalence with the usual *natural* universe hierarchies
  - $\rightarrow$ Normalisation for a theory with **internal level judgements**
- Simpler account of cumulative inductive types
- This has been effectively implemented !
  (https://github.com/raphael-sterbac/elaboration-universes)

Natural universes and The Smallness calculus

Natural universe hierarchies

To have a well-behaved *explicit* universe hierarchy, we need a lot of equations:

$$\mathsf{El}_m(\mathsf{U}_l^m) = \mathsf{U}_l$$

$$\mathsf{El}_l(\Pi^l(a, b)) = \Pi(\mathsf{El}_l(a), \mathsf{El}_l(b))$$

$$\mathsf{El}_m(\uparrow_l^m a) = \mathsf{El}_l(a)$$

(Functoriality)

$$\uparrow_m^n \uparrow_l^m a = \uparrow_l^n a$$

$$\uparrow_m^n \mathsf{U}_l^m = \mathsf{U}_l^n$$

$$\uparrow_l^m \Pi^l(a, b) = \Pi^m(\uparrow_l^m a, \uparrow_l^m b)$$

(Naturality)

Natural universe hierarchies

To have a well-behaved *explicit* universe hierarchy, we need a lot of equations:

$$\mathsf{El}_m(\mathsf{U}_l^m) = \mathsf{U}_l \qquad \uparrow_m^n \uparrow_l^m a = \uparrow_l^n a$$

$$\mathsf{El}_l(\Pi^l(a,b)) = \Pi(\mathsf{El}_l(a), \mathsf{El}_l(b)) \qquad \uparrow_m^n \mathsf{U}_l^m = \mathsf{U}_l^n$$

$$\mathsf{El}_m(\uparrow_l^m a) = \mathsf{El}_l(a) \qquad \uparrow_l^m \Pi^l(a,b) = \Pi^m(\uparrow_l^m a, \uparrow_l^m b)$$

(Functoriality) *(Naturality)*

$\rightarrow$ We can make this precise in the language of SOGATs, but it gets quite complicated...

A judgemental notion of smallness

- All those equations make injectivity of the decoding admissible

A judgemental notion of smallness

- All those equations make injectivity of the decoding admissible
- Why not take it as a rule instead of all of those equations?

$$\frac{\Gamma \vdash M : \mathsf{U}_i}{\Gamma \vdash \Uparrow_i M \ \mathit{type}} \qquad \frac{\Gamma \vdash \Uparrow_i M = \Uparrow_i N \ \mathit{type}}{\Gamma \vdash M = N : \mathsf{U}_i}$$

A judgemental notion of smallness

- All those equations make injectivity of the decoding admissible
- Why not take it as a rule instead of all of those equations?

$$\frac{\Gamma \vdash M : \mathsf{U}_i}{\Gamma \vdash \Uparrow_i M \ \mathit{type}} \qquad \frac{\Gamma \vdash \Uparrow_i M = \Uparrow_i N \ \mathit{type}}{\Gamma \vdash M = N : \mathsf{U}_i}$$

- This is a η/extensionality law for the universe !

A judgemental notion of smallness

- All those equations make injectivity of the decoding admissible
- Why not take it as a rule instead of all of those equations?

$$\frac{\Gamma \vdash M : \mathsf{U}_i}{\Gamma \vdash \Uparrow_i M \ \mathit{type}} \qquad \frac{\Gamma \vdash \Uparrow_i M = \Uparrow_i N \ \mathit{type}}{\Gamma \vdash M = N : \mathsf{U}_i}$$

- This is a η/extensionality law for the universe !
- What does it mean for a type A to be small ?

A judgemental notion of smallness

- All those equations make injectivity of the decoding admissible
- Why not take it as a rule instead of all of those equations?

$$\frac{\Gamma \vdash M : \mathsf{U}_i}{\Gamma \vdash \Uparrow_i M \ \mathit{type}} \qquad \frac{\Gamma \vdash \Uparrow_i M = \Uparrow_i N \ \mathit{type}}{\Gamma \vdash M = N : \mathsf{U}_i}$$

- This is a η/extensionality law for the universe !
- What does it mean for a type A to be small ?

$$\exists\, a : \mathsf{U}_i,\ A = \Uparrow_i a$$

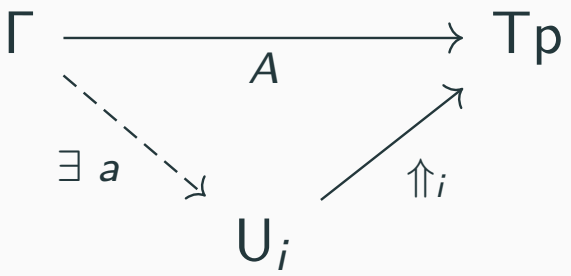
A judgemental notion of smallness

- All those equations make injectivity of the decoding admissible
- Why not take it as a rule instead of all of those equations?

$$\frac{\Gamma \vdash M : \mathsf{U}_i}{\Gamma \vdash \Uparrow_i M \ \mathit{type}} \qquad \frac{\Gamma \vdash \Uparrow_i M = \Uparrow_i N \ \mathit{type}}{\Gamma \vdash M = N : \mathsf{U}_i}$$

- This is a η/extensionality law for the universe !
- What does it mean for a type A to be small ?

$$\exists\, a : \mathsf{U}_i,\ A = \Uparrow_i a$$



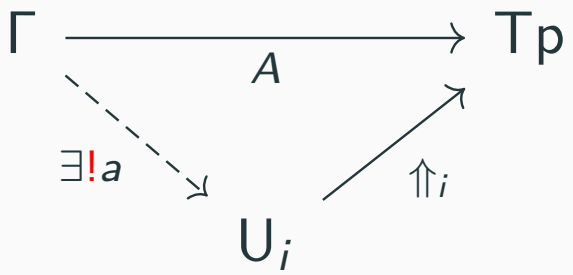
A judgemental notion of smallness

- All those equations make injectivity of the decoding admissible
- Why not take it as a rule instead of all of those equations?

$$\frac{\Gamma \vdash M : \mathsf{U}_i}{\Gamma \vdash \Uparrow_i M \ \mathit{type}} \qquad \frac{\Gamma \vdash \Uparrow_i M = \Uparrow_i N \ \mathit{type}}{\Gamma \vdash M = N : \mathsf{U}_i}$$

- This is a η/extensionality law for the universe !
- What does it mean for a type A to be small ?

$$\exists! a : \mathsf{U}_i,\ A = \Uparrow_i a$$



→ This is now a *proposition*, by injectivity of $\Uparrow_i$!

A judgemental notion of smallness

- We can axiomatise a smallness judgement:

A judgemental notion of smallness

- We can axiomatise a smallness judgement:

$$\frac{\Gamma \vdash M : \mathsf{U}_i}{\Gamma \vdash \Uparrow_i M \ \mathit{small}_i} \qquad \frac{\Gamma \vdash i \leq j \qquad \Gamma \vdash A \ \mathit{small}_i}{\Gamma \vdash A \ \mathit{small}_j}$$

A judgemental notion of smallness

- We can axiomatise a smallness judgement:

$$\frac{\Gamma \vdash M : \mathsf{U}_i}{\Gamma \vdash \Uparrow_i M \ \mathit{small}_i} \qquad \frac{\Gamma \vdash i \leq j \qquad \Gamma \vdash A \ \mathit{small}_i}{\Gamma \vdash A \ \mathit{small}_j}$$

- To reflect types in terms, we introduce a coding operation:

A judgemental notion of smallness

- We can axiomatise a smallness judgement:

$$\frac{\Gamma \vdash M : \mathsf{U}_i}{\Gamma \vdash \Uparrow_i M \ \mathit{small}_i} \qquad \frac{\Gamma \vdash i \leq j \qquad \Gamma \vdash A \ \mathit{small}_i}{\Gamma \vdash A \ \mathit{small}_j}$$

- To reflect types in terms, we introduce a coding operation:

$$\frac{\Gamma \vdash A \ \mathit{small}_i}{\Gamma \vdash \Downarrow_i A : \mathsf{U}_i} \qquad \frac{\Gamma \vdash A \ \mathit{small}_i}{\Gamma \vdash \Uparrow_i \Downarrow_i A = A \ \mathit{type}}$$

Closure under smallness

Smallness of type formers is now easily stated:

$$\frac{\Gamma \vdash A\ \mathit{small}_i \qquad \Gamma, x : A \vdash B[x]\ \mathit{small}_i}{\Gamma \vdash \Pi(A, B)\ \mathit{small}_i}$$

$$\frac{\Gamma \vdash A\ \mathit{small}_i \qquad \Gamma, x : A \vdash B[x]\ \mathit{small}_i}{\Gamma \vdash \Sigma(A, B)\ \mathit{small}_i} \qquad \frac{\Gamma \vdash i < j}{\Gamma \vdash \mathsf{U}_i\ \mathit{small}_j}$$

Closure under smallness

Smallness of type formers is now easily stated:

$$\frac{\Gamma \vdash A\ \mathit{small}_i \qquad \Gamma, x : A \vdash B[x]\ \mathit{small}_i}{\Gamma \vdash \Pi(A, B)\ \mathit{small}_i}$$

$$\frac{\Gamma \vdash A\ \mathit{small}_i \qquad \Gamma, x : A \vdash B[x]\ \mathit{small}_i}{\Gamma \vdash \Sigma(A, B)\ \mathit{small}_i} \qquad \frac{\Gamma \vdash i < j}{\Gamma \vdash \mathsf{U}_i\ \mathit{small}_j}$$

$\Rightarrow$ We can *define* lifts $\uparrow_i^j a := \Downarrow_j \Uparrow_i a$

Closure under smallness

Smallness of type formers is now easily stated:

$$\frac{\Gamma \vdash A\ \mathit{small}_i \qquad \Gamma, x : A \vdash B[x]\ \mathit{small}_i}{\Gamma \vdash \Pi(A, B)\ \mathit{small}_i}$$

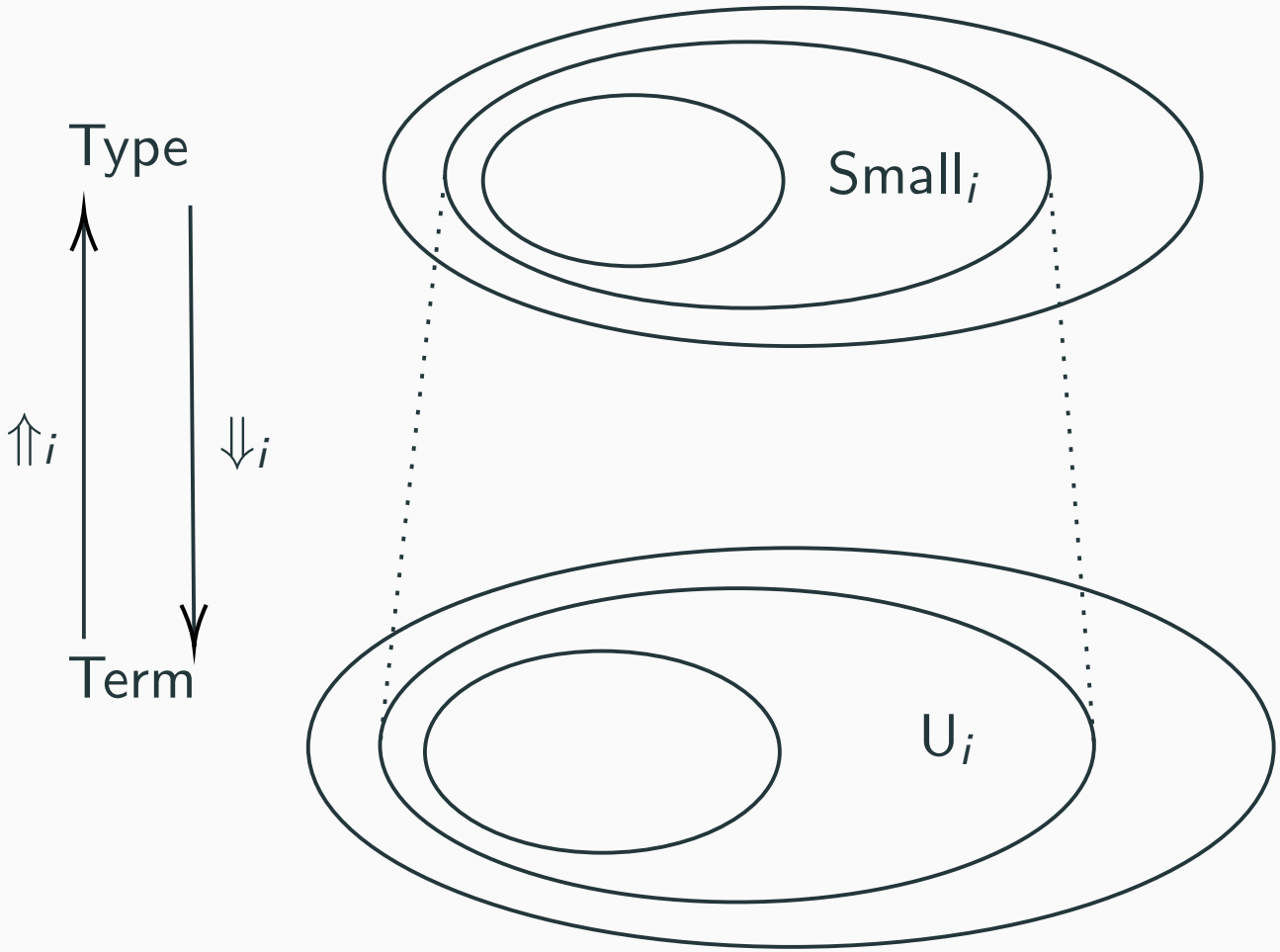
$$\frac{\Gamma \vdash A\ \mathit{small}_i \qquad \Gamma, x : A \vdash B[x]\ \mathit{small}_i}{\Gamma \vdash \Sigma(A, B)\ \mathit{small}_i} \qquad \frac{\Gamma \vdash i < j}{\Gamma \vdash \mathsf{U}_i\ \mathit{small}_j}$$

$\Rightarrow$ We can *define* lifts $\uparrow_i^j a := \Downarrow_j \Uparrow_i a$

$\Rightarrow$ We can *define* codes $\Pi^I(a, b) := \Downarrow_i \Pi(\Uparrow_i a, \Uparrow_i b)$ such that:

$$\Uparrow_I \Pi^I(a, b) = \Pi(\Uparrow_I a, \Uparrow_I b)$$
$$\uparrow_I^m \Pi^I(a, b) = \Pi^m(\uparrow_I^m a, \uparrow_I^m b)$$

In short : reflecting types in terms, locally



Working down from the top

- By keeping a top level type judgement, we are able to do *large* things by default, as in standard mathematical practice:

```
theory Category where
    ob: Type
    hom: ob -> ob -> Type
    // operations and axioms...
```

  → No need for universe polymorphism if we don't need small categories !

- Our theory is still compatible with most forms of universe polymorphism
  → We use the one of Bezem, Coquand, Dybjer and Escardo 2024

Equivalence and Normalisation

Equivalence result

We show that the smallness calculus is equivalent to the natural hierarchy with *injectivity*.

Equivalence result

We show that the smallness calculus is equivalent to the natural hierarchy with *injectivity*.

$$\mathbb{M}_F.\mathsf{Small}_i(A) \coloneqq \sum_{(M:\mathsf{U}_i)} \mathsf{el}_i(M) = A$$
$$\mathbb{M}_F.\Downarrow_i(A, (M, \_)) \coloneqq M$$
$$\mathbb{M}_F.\Uparrow_i(M) \coloneqq \mathsf{el}_i(M)$$

Equivalence result

We show that the smallness calculus is equivalent to the natural hierarchy with *injectivity*.

$$\mathbb{M}_F.\mathsf{Small}_i(A) := \sum_{(M:\mathsf{U}_i)} \mathsf{el}_i(M) = A$$
$$\mathbb{M}_F.\Downarrow_i(A, (M, \_)) := M$$
$$\mathbb{M}_F.\Uparrow_i(M) := \mathsf{el}_i(M)$$

$$\mathbb{M}_G.\mathsf{el}_i(M) := \Uparrow_i M$$
$$\mathbb{M}_G.\uparrow_i^j M := \Downarrow_j \Uparrow_i M$$
$$\mathbb{M}_G.\mathsf{u}_i := \Downarrow_i \mathsf{U}_i$$
$$\mathbb{M}_G.\pi_i(M, N) := \Downarrow_i(\Pi(\Uparrow_i M, \lambda x.\, \Uparrow_i N(x)))$$

This constructs F, G morphisms of CwRs for which we can construct a unit and counit.

Normalisation

- Now, we prove admissibility of injectivity for the natural hierarchy, as a corollary of normalisation.

Normalisation

- Now, we prove admissibility of injectivity for the natural hierarchy, as a corollary of normalisation.
- We have internal level judgements: different notion of model

Normalisation

- Now, we prove admissibility of injectivity for the natural hierarchy, as a corollary of normalisation.
- We have internal level judgements: different notion of model
  $\rightarrow$ We cannot directly use standard gluing techniques...

Normalisation

- Now, we prove admissibility of injectivity for the natural hierarchy, as a corollary of normalisation.
- We have internal level judgements: different notion of model
  $\rightarrow$ We cannot directly use standard gluing techniques...
- We use the new *Sconing* technique of Bocqet, Kaposi and Sattler (2023), that applies to any SOGAT !

Normalisation: Structure for universes

We define the computability family for a code $a : \mathsf{U}_i$ as the set of i-small normalisation structures, i.e. of 5-tuples:

$$
\begin{aligned}
a_p &: \mathsf{El}(1, \mathsf{el}_i(a)) \to \mathcal{U}_i && \text{(computability family)} \\
a_N &: [\mathsf{U}_i \ni_{\mathsf{nf}} a] && \text{(normal form)} \\
a_T &: [\mathsf{Tp} \ni_{\mathsf{nf}} \mathsf{el}_i(a)] && \text{(normal form for } \mathsf{el}_i\text{)} \\
a_l &: (j \geq i : \mathsf{Lvl}) \to [\mathsf{U}_j \ni_{\mathsf{nf}} \uparrow_i^j(a)] && \text{(normal form for } \uparrow_i^j\text{)} \\
a_u &: (k : \mathsf{El}(1, \mathsf{el}_i(a))) \to [\mathsf{el}_i(a) \ni_{\mathsf{ne}} k] \to a_p(k) && \text{(unquoting function)} \\
a_q &: (u : \mathsf{El}(1, \mathsf{el}_i(a))) \to a_p \to [\mathsf{el}_i(a) \ni_{\mathsf{nf}} u] && \text{(quoting function)}
\end{aligned}
$$

Now we can prove admissibility of injectivity by induction on the normal forms:

$$\mathsf{El}_i(a) = \mathsf{El}_i(b) \implies a = b : \mathsf{U}_i$$

Now we can prove admissibility of injectivity by induction on the normal forms:

$$\mathsf{El}_i(a) = \mathsf{El}_i(b) \implies a = b : \mathsf{U}_i$$

To summarize, we have the following:

$$\textbf{NatHier} \underset{\text{Admissibility.}}{\Longleftrightarrow} \textbf{NatHierInj} \underset{\text{CwR Equiv.}}{\Longleftrightarrow} \textbf{SmallCalc}$$

Easy Cumulative Inductive types

Cumulative Inductive types

- Most approaches to inductive types use Russell universes and set models for consistency (Timany and Sozeau 2018)

Cumulative Inductive types

- Most approaches to inductive types use Russell universes and set models for consistency (Timany and Sozeau 2018)
- As argued by Timany and Jacobs (2015), working with explicit lifts could solve some issues, but this becomes complicated...

Cumulative Inductive types

- Most approaches to inductive types use Russell universes and set models for consistency (Timany and Sozeau 2018)
- As argued by Timany and Jacobs (2015), working with explicit lifts could solve some issues, but this becomes complicated...

$\rightarrow$ The fuss-free hierarchy simplifies this !

Internal datatype descriptions

Following Dagand's *Cosmology of datatypes*, we describe an internal judgemental presentation of datatypes

Internal datatype descriptions

Following Dagand's *Cosmology of datatypes*, we describe an internal judgemental presentation of datatypes

$$\frac{}{\mathsf{Desc} : \mathbf{Sort}} \qquad \frac{}{\mathsf{var}_{\mathsf{D}} : \mathsf{Desc}} \qquad \frac{}{\mathsf{unit}_{\mathsf{D}} : \mathsf{Desc}} \qquad \frac{D : \mathsf{Desc} \qquad E : \mathsf{Desc}}{D \times_{\mathsf{D}} E : \mathsf{Desc}}$$

$$\frac{A : \mathsf{Tp} \qquad D : A \to \mathsf{Desc}}{\bigsqcup_{\mathsf{D}}(A, D) : \mathsf{Desc}} \qquad \frac{A : \mathsf{Tp} \qquad D : A \to \mathsf{Desc}}{\bigsqcap_{\mathsf{D}}(A, D) : \mathsf{Desc}}$$

Internal datatype descriptions

Following Dagand's *Cosmology of datatypes*, we describe an internal judgemental presentation of datatypes

$$\frac{}{\mathsf{Desc} : \mathbf{Sort}} \qquad \frac{}{\mathsf{var}_{\mathsf{D}} : \mathsf{Desc}} \qquad \frac{}{\mathsf{unit}_{\mathsf{D}} : \mathsf{Desc}} \qquad \frac{D : \mathsf{Desc} \qquad E : \mathsf{Desc}}{D \times_{\mathsf{D}} E : \mathsf{Desc}}$$

$$\frac{A : \mathsf{Tp} \qquad D : A \to \mathsf{Desc}}{\bigsqcup_{\mathsf{D}}(A, D) : \mathsf{Desc}} \qquad \frac{A : \mathsf{Tp} \qquad D : A \to \mathsf{Desc}}{\bigsqcap_{\mathsf{D}}(A, D) : \mathsf{Desc}}$$

- *Extension* operation $[\![D]\!] : \mathsf{Tp} \to \mathsf{Tp}$ recovering the signature functor of a datatype description

Internal datatype descriptions

Following Dagand's *Cosmology of datatypes*, we describe an internal judgemental presentation of datatypes

$$\frac{}{\mathsf{Desc} : \mathbf{Sort}} \qquad \frac{}{\mathsf{var}_\mathsf{D} : \mathsf{Desc}} \qquad \frac{}{\mathsf{unit}_\mathsf{D} : \mathsf{Desc}} \qquad \frac{D : \mathsf{Desc} \qquad E : \mathsf{Desc}}{D \times_\mathsf{D} E : \mathsf{Desc}}$$

$$\frac{A : \mathsf{Tp} \qquad D : A \to \mathsf{Desc}}{\bigsqcup_\mathsf{D}(A, D) : \mathsf{Desc}} \qquad \frac{A : \mathsf{Tp} \qquad D : A \to \mathsf{Desc}}{\bigsqcap_\mathsf{D}(A, D) : \mathsf{Desc}}$$

- *Extension* operation $[\![D]\!] : \mathsf{Tp} \to \mathsf{Tp}$ recovering the signature functor of a datatype description
- A fixpoint operation $\mathsf{Data}(D)$ that gives the datatype corresponding to a description.

Internal smallness of datatypes

Then we can *define* a smallness predicate for datatypes:

Internal smallness of datatypes

Then we can *define* a smallness predicate for datatypes:

$$\mathsf{Small}^{i}_{\mathsf{D}}(D) \triangleq (x : \mathsf{El}(\mathsf{U}_i)) \to \mathsf{Small}_i\big([\![D]\!](\mathsf{el}_i(x))\big)$$

Internal smallness of datatypes

Then we can *define* a smallness predicate for datatypes:

$$\mathsf{Small}^{i}_{\mathsf{D}}(D) \triangleq (x : \mathsf{El}(\mathsf{U}_i)) \to \mathsf{Small}_i\big([\![D]\!](\mathsf{el}_i(x))\big)$$

And we can simply add the following rule

$$\frac{i : \mathsf{Lvl} \qquad D : \mathsf{Desc} \qquad \_ : \mathsf{Small}^{i}_{\mathsf{D}}(D)}{\_ : \mathsf{Small}_i(\mathsf{Data}(D))}$$

Internal smallness of datatypes

Then we can *define* a smallness predicate for datatypes:

$$\mathsf{Small}^i_{\mathsf{D}}(D) \triangleq (x : \mathsf{El}(\mathsf{U}_i)) \to \mathsf{Small}_i\big([\![D]\!](\mathsf{el}_i(x))\big)$$

And we can simply add the following rule

$$\frac{i : \mathsf{Lvl} \qquad D : \mathsf{Desc} \qquad \_ : \mathsf{Small}^i_{\mathsf{D}}(D)}{\_ : \mathsf{Small}_i(\mathsf{Data}(D))}$$

$\Rightarrow$ We can define *large* inductives, and work with small ones easily !

An example: Lists

We can define the description of lists by:

$$\mathsf{List}_{\mathsf{D}}(A) = \bigsqcup_{\mathsf{D}}(\{\mathsf{nil}, \mathsf{cons}\}, \{\mathsf{nil} \mapsto \mathsf{unit}_{\mathsf{D}}, \mathsf{cons} \mapsto \bigsqcup_{\mathsf{D}}(A, \lambda x.\, \mathsf{var}_{\mathsf{D}} \times_{\mathsf{D}} \mathsf{unit}_{\mathsf{D}})\}$$

An example: Lists

We can define the description of lists by:

$$\mathsf{List}_{\mathsf{D}}(A) = \bigsqcup_{\mathsf{D}}(\{\mathsf{nil}, \mathsf{cons}\}, \{\mathsf{nil} \mapsto \mathsf{unit}_{\mathsf{D}}, \mathsf{cons} \mapsto \bigsqcup_{\mathsf{D}}(A, \lambda x.\, \mathsf{var}_{\mathsf{D}} \times_{\mathsf{D}} \mathsf{unit}_{\mathsf{D}})\}$$

Then, we recover the well-known signature functor:

$$[\![\mathsf{List}_{\mathsf{D}}(A)]\!] : X \mapsto 1 + A \times X$$

Instanciated on a decoding, we get:

$$[\![\mathsf{List}_{\mathsf{D}}(A)]\!](\mathsf{el}_i(x)) = 1 + A \times \mathsf{el}_i(x)$$

An example: Lists

We can define the description of lists by:

$$\mathsf{List}_{\mathsf{D}}(A) = \bigsqcup_{\mathsf{D}}(\{\mathsf{nil}, \mathsf{cons}\}, \{\mathsf{nil} \mapsto \mathsf{unit}_{\mathsf{D}}, \mathsf{cons} \mapsto \bigsqcup_{\mathsf{D}}(A, \lambda x.\, \mathsf{var}_{\mathsf{D}} \times_{\mathsf{D}} \mathsf{unit}_{\mathsf{D}})\}$$

Then, we recover the well-known signature functor:

$$[\![\mathsf{List}_{\mathsf{D}}(A)]\!] : X \mapsto 1 + A \times X$$

Instanciated on a decoding, we get:

$$[\![\mathsf{List}_{\mathsf{D}}(A)]\!](\mathsf{el}_i(x)) = 1 + A \times \mathsf{el}_i(x)$$

Which satisfies the smallness predicate only if A is small !

An example: Lists

We can define the description of lists by:

$$\mathsf{List}_{\mathsf{D}}(A) = \bigsqcup_{\mathsf{D}}(\{\mathsf{nil}, \mathsf{cons}\}, \{\mathsf{nil} \mapsto \mathsf{unit}_{\mathsf{D}}, \mathsf{cons} \mapsto \bigsqcup_{\mathsf{D}}(A, \lambda x.\, \mathsf{var}_{\mathsf{D}} \times_{\mathsf{D}} \mathsf{unit}_{\mathsf{D}})\}$$

Then, we recover the well-known signature functor:

$$[\![\mathsf{List}_{\mathsf{D}}(A)]\!] : X \mapsto 1 + A \times X$$

Instanciated on a decoding, we get:

$$[\![\mathsf{List}_{\mathsf{D}}(A)]\!](\mathsf{el}_i(x)) = 1 + A \times \mathsf{el}_i(x)$$

Which satisfies the smallness predicate only if A is small !

Thus we recover :

$$\frac{\_ : \mathsf{Small}_i(A)}{\_ : \mathsf{Small}_i(\mathsf{List}(A))}$$

Conclusion

Summary

Summary

- New presentation for cumulative algebraic universes

Summary

- New presentation for cumulative algebraic universes
  - $\rightarrow$ Smallness as a proposition

Summary

- New presentation for cumulative algebraic universes
  - $\rightarrow$ Smallness as a proposition
  - $\rightarrow$ Simple implementation

Summary

- New presentation for cumulative algebraic universes
  - $\rightarrow$ Smallness as a proposition
  - $\rightarrow$ Simple implementation
  - $\rightarrow$ Direct interpretation in models

Summary

- New presentation for cumulative algebraic universes
  - $\rightarrow$ Smallness as a proposition
  - $\rightarrow$ Simple implementation
  - $\rightarrow$ Direct interpretation in models
- Equivalence with natural universe hierarchies

Summary

- New presentation for cumulative algebraic universes
  - $\rightarrow$ Smallness as a proposition
  - $\rightarrow$ Simple implementation
  - $\rightarrow$ Direct interpretation in models
- Equivalence with natural universe hierarchies
- Simpler account of cumulative inductive types

Summary

- New presentation for cumulative algebraic universes
  - $\rightarrow$ Smallness as a proposition
  - $\rightarrow$ Simple implementation
  - $\rightarrow$ Direct interpretation in models
- Equivalence with natural universe hierarchies
- Simpler account of cumulative inductive types

Future work

- Shulman showed in 2019 that all ∞-toposes had strict univalent universes, but left cumulativity open

Summary

- New presentation for cumulative algebraic universes
  - $\rightarrow$ Smallness as a proposition
  - $\rightarrow$ Simple implementation
  - $\rightarrow$ Direct interpretation in models
- Equivalence with natural universe hierarchies
- Simpler account of cumulative inductive types

Future work

- Shulman showed in 2019 that all ∞-toposes had strict univalent universes, but left cumulativity open
  - $\rightarrow$ It should be possible to use the techniques of Gratzer, Shulman and Sterling (2022) to extend this result !

Thanks for your attention !