Algebraic Universes and Variances for All

TYPES 2025, Glasgow, Scotland — June 10th 2025

Matthieu Sozeau, Inria & LS2N, University of Nantes, France

Marc Bezem, University of Bergen, Finland

ROCQ

Inria

Outline

1. Universes in Type Theory: Typing and Elaboration
2. Algebraic Universes and Variances in a Polymorphic, Cumulative Type Theory

Type Theory with Universes

$\mathcal{U} : \mathcal{U}$ is inconsistent (System U/U$^-$, Girard's paradox, Hurkens)

Use a hierarchy $\mathcal{U}_i (i \in \mathbb{N})$ to stratify universes.

$$\vdash \mathcal{U}_i : \mathcal{U}_{i+1}$$

Type Theory with Universes

$\mathcal{U} : \mathcal{U}$ is inconsistent (System U/U$^-$, Girard's paradox, Hurkens)

Use a hierarchy $\mathcal{U}_i (i \in \mathbb{N})$ to stratify universes.

$$\vdash \mathcal{U}_i : \mathcal{U}_{i+1}$$

$$\frac{\vdash A : \mathcal{U}_i \quad x : A \vdash B : \mathcal{U}_j}{\vdash \Pi x : A.B : \mathcal{U}_{i \sqcup j}}$$

Usually with a bottom universe $\mathcal{U}_0$ (≡ (predicative) `Set` in Rocq)

Type Theory with Polymorphic Universes

Universe polymorphism = universal quantification on universe variables

$$
\begin{aligned}
u, v ::= {} & l && \text{(universe level variable)} \\
\mid {} & 0 && \text{(bottom universe)} \\
\mid {} & u + 1 && \text{(successor universe)} \\
\mid {} & \max(u, v) && \text{(supremum, } \sqcup \text{)}
\end{aligned}
$$

0 is neutral for max, successor distributes on max, max is idempotent, associative, commutative and subsumptive: $\max(u, u + 1) = u + 1$

During elaboration: level **metavariables** $?_l$ subject to constraints

Non-unitary theory, no m.g.u.s in general, stuck constraints in practice.

Cumulativity + Polymorphism

Cumulative hierarchy $\mathcal{U}_i (i \in \mathbb{N})$:

$$\vdash \mathcal{U}_i \leq \mathcal{U}_{i+1} \qquad \vdash \Pi x : A.\mathcal{U}_i \leq \Pi x : A.\mathcal{U}_j \quad \text{iff } i \leq j$$

With polymorphism $\Rightarrow$ bounded quantification on universe variables (Harper and Pollack, TCS'91, Sozeau and Tabareau, ITP'14)

```
Definition le@{i j | i ≤ j} (A : Type@{i}) : Type@{j} := A.
```

Consistency check:

$$u_0 \leq u_1 \wedge \ldots \wedge u_i \leq u_{i+1} \vdash u \leq v$$

Herbelin's Algebraic Universes

Previous type inference algorithm of Rocq restricted to:

$$u_0 \leq \boldsymbol{l_0} \wedge \ldots \wedge u_i \leq \boldsymbol{l_i} \vdash u \leq \boldsymbol{l}$$

I.e. $\max()$, $+$ expressions could only appear in inferred types of terms, not in terms themselves $\Rightarrow$ need a "freshening" operation.

Herbelin's Algebraic Universes

Previous type inference algorithm of Rocq restricted to:

$$u_0 \leq \boldsymbol{l_0} \wedge \ldots \wedge u_i \leq \boldsymbol{l_i} \vdash u \leq \boldsymbol{l}$$

I.e. $\max()$, $+$ expressions could only appear in inferred types of terms, not in terms themselves $\Rightarrow$ need a "freshening" operation.

Example elaboration of `@id ?X Type@{0}`. We have `Type@{0} : Type@{0+1}`, but:

`?X := Type@{0+1}` forbidden as `0 + 1` is an algebraic expression.

`?X := Type@{?i}, 0 < ?i` ok as `?i` is a level variable.

- Pro: efficient entailment checking is possible (Bender et al.)
- Cons: restricted algebra, error-prone API, proliferation of fresh levels

Algebraic Universes and Variances in a Polymorphic, Cumulative Type Theory

Bezem and Coquand's algorithm

"Loop-checking and the uniform word problem for join-semilattices with an inflationary endomorphism", Bezem and Coquand, TCS, 2022

Entailment and loop-checking are decidable in PTIME for the $(\max, +, \leq)$ algebra.

Bezem and Coquand's algorithm

"Loop-checking and the uniform word problem for join-semilattices with an inflationary endomorphism", Bezem and Coquand, TCS, 2022

Entailment and loop-checking are decidable in PTIME for the $(\max, +, \leq)$ algebra.

Based on a forward search algorithm on Horn clauses, e.g.:

$$[\![w \leq \max(u+1, v) \wedge s < w]\!] = \{u+1, v \to w; w \to s+1\}$$

Bezem and Coquand's algorithm

"Loop-checking and the uniform word problem for join-semilattices with an inflationary endomorphism", Bezem and Coquand, TCS, 2022

Entailment and loop-checking are decidable in PTIME for the $(\max, +, \leq)$ algebra.

Based on a forward search algorithm on Horn clauses, e.g.:

$$[\![w \leq \max(u+1, v) \wedge s < w]\!] = \{u+1, v \to w; w \to s+1\}$$

Minimal model of the variables and clauses in $\mathbb{N} + \infty$:

$$u \mapsto 0, v \mapsto 0, w \mapsto 1, s \mapsto 2 \quad \text{for the clauses above}$$

Intuitively, the lowest universe (s) has highest value and we maintain a distance to it, ∞ represents a loop.

Using Bezem and Coquand's algorithm in Rocq

- Model benign loops $l \leq l' + k \wedge l' + k \leq l$ as substitutions $l := l' + k$

Using Bezem and Coquand's algorithm in Rocq

- Model benign loops $l \leq l' + k \wedge l' + k \leq l$ as substitutions $l := l' + k$
- Termination, correctness and (WIP) completeness proofs in Rocq/MetaRocq

Using Bezem and Coquand's algorithm in Rocq

- Model benign loops $l \leq l' + k \wedge l' + k \leq l$ as substitutions $l := l' + k$
- Termination, correctness and (WIP) completeness proofs in Rocq/MetaRocq
- Generalizes the system to handle all constraints, simplifying the APIs

Minimization

Elaboration with level metavariables and cumulativity ⇒ large number of unnecessary constraints.

Example: `id@{?i} nat 0 : nat` ⇒ `0 ≤ ?i` rather than `?i = 0`

W.l.o.g. `?i := 0`, same principal type and convertibility properties:

$$\texttt{id@\{?i\}}\ \texttt{nat}\ \texttt{0} \to^* \texttt{0} \ {}^*\!\leftarrow \texttt{id@\{0\}}\ \texttt{nat}\ \texttt{0}$$

Formally, record **where** and **how** bound universes appear in a definition to perform sound simplifications.

Universe Variances

Record for each definition the occurrences of bound universe variables in their body and type and their associated variance (compositionally).

A variance is either: irrelevant ($*$), covariant ($+$), contravariant ($-$) or invariant ($=$).

Universe Variances

Record for each definition the occurrences of bound universe variables in their body and type and their associated variance (compositionally).

A variance is either: irrelevant ($*$), covariant ($+$), contravariant ($-$) or invariant ($=$).

Cumulative Inductive Types already use this notion.

```
Inductive eq@{i} (A : Type@{i}) (a : A) : A → Type@{i} :=
| eq_refl : eq A a a.
```

For `i`:

- contravariant in the first binder
- covariant in the type
- irrelevant in the constructor arguments

Universe Variances

```
Inductive eq@{i} : (A : Type@{i}), A → A → Type@{i} := ..
```

Timany and Sozeau have shown that this justifies treating the i universe as **irrelevant** when eq is applied to its parameter.

`eq@{i} nat 0 1` ≡ `eq@{j} nat 0 1 : Type@{max(i,j)}` for all `i`, `j`

⇒ Justification behind so-called `template-polymorphic` inductives in Rocq

Idea: Lift this analysis to all definitions.

Universe Variances for Definitions

```
Definition relation@{i} (A : Type@{i}) := A → A → Prop.

Class Proper@{i} (A : Type@{i}) (R : relation A) := ∀ m, R m m.

Definition respectful@{i j} (A : Type@{i}) (B : Type@{j})
  (RA : relation A) (RB : relation B) : relation@{max(i,j)} (A → B) :=
  fun f g ⇒ ∀ x y, RA x y → RB (f x) (g y).
```

All universe binders here are irrelevant, when defs are fully applied.

$$\texttt{relation@\{i\}}\ A \equiv A \to A \to \texttt{Prop} \equiv \texttt{relation@\{j\}}\ A \equiv \texttt{relation@\{max(i,j)\}}\ A$$

- Sound first-order unification: enforce constraints only on invariant or covariant universes before unifying actual arguments.
- Enables simplifications: instances for the `i` binder can be lowered w.l.o.g.

Universe Variances for Definitions

Polymorphic functions on containers, e.g.:

```
map@{*i *j} (A : Type@{i}) (B : Type@{j}) : (A → B) → list@{i} A → list@{j} B
```

HoTT path algebra: groupoid laws use the equality's universe level irrelevantly, e.g.:

```
concat1p@{*i} (A : Type@{i}) (x y : A) (p : x = y) : 1 @ p = p.
```

User Interface

Mixed inference mode

Declaring only quantified universes:

```
Lemma foo@{i j} : forall X : Type@{i}, predicate@{i j} X. Proof. …. Qed.
```

Solves all metavariables in terms of `i` and `j` or errors (rare).

Checking mode

```
Lemma foo@{i j| } : Type@{j}.
Proof. Fail exact Type@{i}. (* i < j not implied by the constraints. *)
```

- Porting the HoTT library: $\sim$ 500LoC diff, adding explicit universe bindings.
- Use checking for fine-grained cumulativity constraints in some cases.

Summary

- Bezem and Coquand's algorithm for full universe inference and checking with algebraic universes and cumulativity
- Variances for sound and efficient conversion, unification and minimization
- Supports both explicit or implicit universes and constraints in Rocq

Summary

- Bezem and Coquand's algorithm for full universe inference and checking with algebraic universes and cumulativity
- Variances for sound and efficient conversion, unification and minimization
- Supports both explicit or implicit universes and constraints in Rocq

WIP:

- Complete the formal proof of the loop-checking algorithm.
- Porting of Rocq developments: done for HoTT, next is math-comp

Rocq PR #20754

Unification with $(\max, +)$

The theory is **not unitary**: no m.g.u.'s in general for the $(\max, +)$ algebra:

Thiago Felicissimo PhD, "Generic Bidirectional Typing in a Logical Framework for Dependent Type Theories". (Theorem 17.1)

Example: $?_x + 1 = ?_y \sqcup ?_z$ solvable but has no m.g.u.

Stuck constraints in practice (Agda, Lean).