Algebraic Universes and Variances For All

Matthieu Sozeau[1] and Marc Bezem[2]

[1] LS2N & Inria de l'Université de Rennes,
Nantes, France
matthieu.sozeau@inria.fr
[2] University of Bergen
Bergen, Norway
Marc.Bezem@uib.no

Building on a refinement of the novel loop-checking algorithm of Bezem and Coquand [2], we present a new design and implementation of cumulative universe polymorphism for dependent type theories, materialized in a branch [6] of the Rocq prover. Our system handles universe level polymorphism for the full universe algebra $(\ell, 0, +1, \max)$. The constraint solving algorithm based on Horn clauses is theoretically polynomial. However, to stay competitive with the previous algorithm based on the work of Bender et al. [1], we make the algorithm incremental and integrate a union-find datastructure to capture semantic universe level equalities by efficiently represented equivalence classes; this makes the algorithm more user-friendly in practice, reflecting inferred equalities instantaneously in annotations.

On the type theory side, the original system of Sozeau and Tabareau [9] is simply generalized: universe-polymorphic binders can take universe expressions as arguments, all constraints can be enforced and checked, and we remove a long-standing distinction between inferred types and checkable types [4]. This puts the system on par with Agda and Lean's (non-cumulative) universe systems.

On the elaboration side, the previous system [9] introduced a heuristic in unification to avoid unnecessary unfoldings and an (infamous) minimization procedure to try to reduce the number of unnecessarily quantified universes and constraints, which naturally grows exponentially fast in presence of implicit cumulativity [3]. We generalize the notion of universe variance introduced for cumulative inductive types [10] to all definitions and use it to refine the unification heuristics and provide a more principled universe level metavariable solving process: it should now guarantee a principal typing property.

The talk will present all these aspects along with examples and our preliminary performance benchmarks and compatibility status.

A new loop-checking algorithm for universes

In dependent type theories with predicative universe levels, universe checking involves verifying equalities (inequalities $u \leq v$ are interpreted as $v = \max(u, v)$) between universe level expressions:

Universe levels u, v	$:=$	ℓ	(universe level variable)
	$\mid$	0	(bottom universe)
	$\mid$	$u + 1$	(successor)
	$\mid$	$\max(u, v)$	(supremum)

Here 0 should be neutral for max, successor distributes over max and max is idempotent, subsumptive ($\max(u, u + 1) = u + 1$), associative and commutative. This already generates a non-trivial quotient where for example $\max(0 + 1, u + 1) = u + 1$. To make matters a little more interesting one might add level *metavariables* (noted $?_\ell$) to solve during elaboration

and unification. Then, unification can get stuck: $\max(?_l, ?_k) = \max(?_u, ?_v)$ has no general solution. In Agda and Lean, this requires adding user annotations, instantiating metavariables sufficiently to disambiguate. This situation is quite problematic when one considers automated tactics as used in Rocq or Lean, where it might be difficult to make the tactics annotate generated terms. We would rather like to keep the universe checking entirely automated, so Rocq rather handles a context of universe constraints that are incrementally checked for consistency. Previous versions of Rocq did not run into the problem of getting stuck as the system carefully avoided generating $\max(\_, \_) = \max(\_, \_)$ constraints [4], at the cost of a less expressive theory and unpleasant API: inferred types needed to be "refreshed" before being put into terms, using fresh universe variables as proxies for max expressions. Currently in Rocq, we actually consider a cumulative hierarchy of universes, so equational problems become *entailments* of shape $u_0 \leq \ell_0 \dots u_i \leq \ell_i \vdash u \leq \ell$. Note the restriction to levels on the right hand side of inequality constraints which is enforced by the type inference algorithm [4]. We lift this restriction to levels on the right using the new algorithm of [2], which is able to handle all constraints. *I.e.,* the whole algebra is supported and the entailments to check can now have the general shape: $u_0 \leq u_1 \dots u_i \leq u_i + 1 \vdash u \leq v$. This also greatly simplifies the API for term manipulation. In addition, it cannot "get stuck": it is correct and complete for the (in)equational theory.

The new algorithm [2] is based on a representation of constraints as Horn clauses and performs forward reasoning to check for consistency. Informally, it maintains an assignment of the variables in $\mathbb{N} \cup \{+\infty\}$ that forms a minimal model of the Horn clauses; the value $+\infty$ signals a loop. The algorithm can be used both to check that a new constraint is consistent or produce a loop (used during elaboration), and to decide whether a constraint is deducible from existing constraints (used during kernel type-checking). We extended the algorithm so that each time a constraint $\ell \leq \ell'$ is introduced, we check if $\ell' \leq \ell$ also holds, in which case we choose a canonical variable among the two and substitute the other one by it in the constraints. Then checks for equality between level variables can be done in constant time, which is crucial for performance.

We tested the new implementation against the previous one on the standard library and some large Rocq projects (e.g. math-comp, the HoTT library and Iris), and found a 15% overhead on average, which is expected against a highly optimized implementation for a simpler problem. Note that on average universe checking accounts for 25% of Rocq's compilation times, according to an empirical study by Pierre-Marie Pédrot. However, most libraries do not use polymorphism yet but rely on a very large global graph of constraints, which is less suitable to the new algorithm. On the other hand, experiments on universe polymorphic code shows drastic improvements on performance, with much less constraints being generated and more reasonable inferred quantifications on universes.

Variances

We found the largest slowdowns of the new algorithm in Rocq's `Reals` library, in proof scripts using the `setoid_rewrite` tactics [8]. Those proofs generated a large amount of *global* universe constraints. To solve this, we switched the `setoid_rewrite` tactic to rather use universe polymorphic definitions, avoiding interaction with the global graph. The main polymorphic definitions of interest are:

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

The tactic performs proof search on instances of `Proper@{i} R m`, indexed by the syntax of the relation `R` and the morphism `m`. If done naïvely, however, unification will spend a lot of time unifying universe levels before doing useful work, for example in a unification constraint `relation@{i} A` $\simeq$ `relation@{j} B` will always first try to unify i and j before looking at the arguments `A` and `B`. However one can see that unfolding `relation` would result in a unification that would rather start comparing `A` and `B` as `i` and `j` do not appear in the unfoldings. For `respectful A B RA RB`, the same is true for `i` and `j`.

Hence, we reify information on the occurrences of universe variables as a variance annotation on universe level binders, reusing the variance analysis used for cumulative inductive types. A variance can be one of: irrelevant ($*$), covariant ($+$), contravariant ($-$) or equivariant ($=$). We automatically infer an over-approximation of the variance of each variable at specific positions: in the nth binder type, the return type or the body of the definition. If a variable does not appear in a specific position, it is irrelevant. Typically, parameter universes appear contravariantly in some binder of the definition (e.g. in the 1st binder `A : Type@{i}` for `relation`) but irrelevantly everywhere else. So, if the definition is applied to at least one argument, for example `i` in `relation@{i} A` in `Proper`'s second binder, then we can consider this occurrence irrelevant as well: unfolding `relation@{i} A` makes `i` disappear. Applying this analysis compositionally results in most universe binders being considered as irrelevant in `Proper R m` instances. We can use this automatically inferred information during unification to avoid doing any unification on universes until the real indexes of the proof search (the syntactic shape of the relation and morphism) are unified, resulting in type unifications that will set the universes straight. This improves unification performance significantly. The new `setoid_rewrite` tactic is even more performant than the previous tactic in the `Reals` library.

In addition to improving unification performance, this variance information can also be used to drive the so-called "minimization" algorithm we we now call simplification that solves level metavariables: those appearing in contravariant positions only can be maximized without loss of generality while those appearing in covariant positions only can be minimized without loss of generality. The user can also annotate definitions to provide the exact set of universe variables he expects a definition to quantify over, and simplification will then proceed to instantiate metavariables in terms of the explicitly quantified variables, providing much finer control to the user than the previous implementation. Preliminary experiments using this to develop a new sort-and-universe polymorphic prelude [7, 5] are encouraging.

Acknowledgments We are grateful to Thierry Coquand, Martín Escardó, Pierre-Marie Pédrot & Nicolas Tabareau for helpful discussions on this work.

References

[1] Michael A. Bender, Jeremy T. Fineman, Seth Gilbert, and Robert E. Tarjan. A new approach to incremental cycle detection and related problems. *ACM Trans. Algorithms*, 12(2):14:1–14:22, December 2015. ISSN 1549-6325. doi: 10.1145/2756553. URL http://doi.acm.org/10.1145/2756553.

[2] Marc Bezem and Thierry Coquand. Loop-checking and the uniform word problem for join-semilattices with an inflationary endomorphism. *Theoretical Computer Science*, 913: 1–7, 2022. ISSN 0304-3975. doi: https://doi.org/10.1016/j.tcs.2022.01.017. URL https://www.sciencedirect.com/science/article/pii/S0304397522000317.

[3] Robert Harper and Robert Pollack. Type checking with universes. *Theor. Comput. Sci.*, 89(1):107–136, 1991.

[4] Hugo Herbelin. Type Inference with Algebraic Universes in the Calculus of Inductive Constructions. 2005. URL http://pauillac.inria.fr/~herbelin/publis/univalgcci.pdf. Manuscript.

[5] Josselin Poiret, Matthieu Sozeau, and Nicolas Tabareau. Sort and Universe Polymorphic Core Library for Rocq, March 2025. URL https://github.com/jpoiret/coq/tree/new-prelude.

[6] Matthieu Sozeau. Algebraic Universes and Variances for Rocq, March 2025. URL https://github.com/mattam82/coq/tree/universes-clauses.

[7] Josselin Poiret, Gaëtan Gilbert, Kenji Maillard, Pierre-Marie Pédrot, Matthieu Sozeau, Nicolas Tabareau, and Éric Tanter. All Your Base Are Belong to $\mathcal{U}^s$: Sort Polymorphism for Proof Assistants. *Proc. ACM Program. Lang.*, 9(POPL):2253–2281, 2025. doi: 10.1145/3704912. URL https://doi.org/10.1145/3704912.

[8] Matthieu Sozeau. A New Look at Generalized Rewriting in Type Theory. *Journal of Formalized Reasoning*, 2(1):41–62, December 2009.

[9] Matthieu Sozeau and Nicolas Tabareau. Universe Polymorphism in Coq. In Gerwin Klein and Ruben Gamboa, editors, *ITP 2014*, volume 8558 of *Lecture Notes in Computer Science*, pages 499–514. Springer, 2014. ISBN 978-3-319-08969-0. doi: 10.1007/978-3-319-08970-6_32. URL http://dx.doi.org/10.1007/978-3-319-08970-6.

[10] Amin Timany and Matthieu Sozeau. Cumulative Inductive Types In Coq. In Hélène Kirchner, editor, *FSCD*, volume 108 of *LIPIcs*, pages 29:1–29:16, July 2018. ISBN 978-3-95977-077-4. doi: 10.4230/LIPIcs.FSCD.2018.29. URL https://doi.org/10.4230/LIPIcs.FSCD.2018.29.