Universe Polymorphism in Coq

Matthieu Sozeau & Nicolas Tabareau, Inria Paris & Rennes

ITP 2014
July 16th 2014
Vienna, Austria

What are universes?

Universes are the types of *types*, e.g:

- $\mathsf{nat}, \mathsf{bool} : \mathtt{Type}_0$
- $\mathtt{Type}_0 : \mathtt{Type}_1$
- $\mathsf{list} : \mathtt{Type}_0 \to \mathtt{Type}_0$
- $\forall \alpha : \mathtt{Type}_0, \mathsf{list}\ \alpha : \mathtt{Type}_1$
- $\forall n : \mathsf{nat}, \{n = 0\} + \{n \neq 0\} : \mathtt{Type}_0$

How are they organised?

A *hierarchy* of predicative universes $\texttt{Type}_0 < \texttt{Type}_1 < \ldots$

- Avoids the $\texttt{Type} : \texttt{Type}$ paradox (system U^-)
- Replicates RUSSELL's paradox of $\{x \mid x \notin x\}$, the set of all sets etc....
- Think of $\texttt{Type}_0$ as sets, $\texttt{Type}_1$ as classes etc...

Coq's theory

sort of t = type of the type of t, necessarily a $\texttt{Type}_i$.

$$\textsc{Type-intro}\quad \frac{\vdash \Gamma \quad (i \in \mathbb{N})}{\Gamma \vdash \texttt{Type}_i : \texttt{Type}_{i+1}}$$

$$\textsc{Type-prod}\quad \frac{\Gamma \vdash A : \texttt{Type}_i \qquad \Gamma, x : A \vdash B : \texttt{Type}_j}{\Gamma \vdash \Pi x : A.B : \texttt{Type}_{\mathsf{max}(i,j)}}$$

Typical ambiguity

Working with explicit universe indices is cumbersome, annotations pervade definitions and proofs.

$\Rightarrow$ Allow *typical ambiguity* (first used by Russell in Principia).

Idea: write `Type` to mean any type that "fits" (keeps the system consistent).

- On paper: let the reader infer levels for universes and check consistency.
- On computer: let the computer infer levels and check consistency in the background.

Floating universes

Formally, translate from anonymous Types to explicit $\mathtt{Type}_i$s.
But in general many i's can work!

$$\mathtt{Definition}\ \mathrm{id}\ (A : \mathtt{Type})\ (a : A) := a.$$

$$\rightsquigarrow\ \vdash \mathrm{id} : \Pi(A : \mathtt{Type}_0),\ A \to A : \mathtt{Type}_1$$

or

$$\rightsquigarrow\ \vdash \mathrm{id} : \Pi(A : \mathtt{Type}_1),\ A \to A : \mathtt{Type}_2$$

or . . . ?

$$\Rightarrow \text{ universe variables}$$

Floating universes and constraints

Consistency is now ensure by giving an assignment of natural numbers to universe variables, satisfying *constraints*. New judgment $\vdash_{float}$

$$\textsc{Type-intro}\quad \frac{\vdash_{float} \Gamma \quad (i, j \in \mathbb{L})}{\Gamma \vdash_{float} \mathtt{Type}_i : \mathtt{Type}_j \rightsquigarrow i < j}$$

$$\textsc{Type-prod}\quad \frac{\Gamma \vdash_{float} A : \mathtt{Type}_i \qquad \Gamma, x : A \vdash B : \mathtt{Type}_j}{\Gamma \vdash_{float} \Pi x : A.B : \mathtt{Type}_k \rightsquigarrow \mathsf{max}(i, j) \leq k}$$

Without polymorphism

Floating levels give a false sense of polymorphism:

$$\texttt{Definition}\ \mathrm{id}\ (A : \texttt{Type})\ (a : A) := a$$

$$\rightsquigarrow\ \vdash \mathrm{id} : \Pi(A : \texttt{Type}_l),\ A \to A : \texttt{Type}_{l+1}$$

$\Rightarrow l$ is *not* quantified at the definition level here, it is *global*:

$$\not\vdash \mathrm{id}\ (\Pi(A : \texttt{Type}_l),\ A \to A)\ \mathrm{id} : \tau$$

Because $l + 1 \not\leq l$.

With polymorphism

Real, bounded polymorphism:

$$\texttt{Polymorphic Definition}\ \mathrm{id}\ (A : \texttt{Type})\ (a : A) := a$$

$$\underline{\mathrm{id}}_l : \Pi(A : \texttt{Type}_l),\ A \to A$$

$\Rightarrow l$ is quantified at the definition level now and we can *instantiate* it at each application:

$$l < k \vdash_{poly} \underline{\mathrm{id}}_k\ (\Pi(A : \texttt{Type}_l),\ A \to A)\ \underline{\mathrm{id}}_l : (\Pi(A : \texttt{Type}_l),\ A \to A)$$

1 Introduction

2 Elaborating Universes
- Universe polymorphic definitions
- Unification
- Minimization
- Dealing with Prop
- Implementation & benchmarks

Constraint checking

Constraints are generated once at refinement time outside the kernel. The kernel just checks that the constraints are consistent and sufficient to typecheck the terms.

$$\text{universe context} \quad \Psi \quad ::= \quad \overrightarrow{i} \models \Theta$$

Constraint checking

Constraints are generated once at refinement time outside the kernel. The kernel just checks that the constraints are consistent and sufficient to typecheck the terms.

$$\text{universe context} \quad \Psi \quad ::= \quad \overrightarrow{i} \vDash \Theta$$

Elaboration in bidirectionl fashion:

- Inference: $\Gamma; \Psi \vdash t \Uparrow \rightsquigarrow \Psi' \vdash t' : T$
- Checking: $\Gamma; \Psi \vdash t \Downarrow T \rightsquigarrow \Psi' \vdash t' : T$

Constraint checking

Constraints are generated once at refinement time outside the kernel. The kernel just checks that the constraints are consistent and sufficient to typecheck the terms.

$$\text{universe context} \quad \Psi \quad ::= \quad \overrightarrow{i} \vDash \Theta$$

Elaboration in bidirectionl fashion:

- Inference: $\Gamma; \Psi \vdash t \Uparrow \rightsquigarrow \Psi' \vdash t' : T$
- Checking: $\Gamma; \Psi \vdash t \Downarrow T \rightsquigarrow \Psi' \vdash t' : T$

$$\textsc{Check-Type}\ \frac{\theta \vdash \texttt{Type}_{i+1} \leq T \rightsquigarrow \theta'}{\Gamma; us \vDash \theta \vdash \texttt{Type} \Downarrow T \rightsquigarrow us, i \vDash \theta' \vdash \texttt{Type}_i : T}$$

Introducing universe polymorphic definitions

Suppose a top-level `Definition` id $: T := t$.

Introducing universe polymorphic definitions

Suppose a top-level `Definition` id $: T := t$.

1. $\Gamma; \vdash T \Uparrow \rightsquigarrow \Psi \vdash T' : s$

Introducing universe polymorphic definitions

Suppose a top-level `Definition` id $: T := t$.

1. $\Gamma; \vdash T \Uparrow \rightsquigarrow \Psi \vdash T' : s$
2. $\Gamma; \Psi \vdash t \Downarrow T' \rightsquigarrow i \vDash \theta \vdash t : T'$

Introducing universe polymorphic definitions

Suppose a top-level `Definition` $\mathrm{id} : T := t$.

1. $\Gamma; \vdash T \Uparrow \rightsquigarrow \Psi \vdash T' : s$
2. $\Gamma; \Psi \vdash t \Downarrow T' \rightsquigarrow i \vDash \theta \vdash t : T'$
3. Add $\mathrm{id} : \forall\ i \vDash \theta, T' := t$ to the environment.

Guiding principle:
Constants are transparent, indistinguishable from their bodies.

Using universe polymorphic definitions

$$\textsc{Infer-Cst}\ \frac{(\mathrm{id} : \forall\ i \vDash \theta, T) \in \Sigma \qquad \overrightarrow{l} \notin \overrightarrow{u}}{\Gamma; \overrightarrow{u} \vDash \Theta \vdash \mathrm{id} \Uparrow \rightsquigarrow \psi \vdash \mathrm{id}_{\overrightarrow{l}} : T[\overrightarrow{l} / \overrightarrow{i}]}$$

where $\psi = \overrightarrow{u}, \overrightarrow{l} \vDash \Theta \cup \theta[\overrightarrow{l} / \overrightarrow{i}]$

$\Rightarrow$ Constants now carry their universe substitution/instance.

$\Rightarrow$ Inductives and constructors treated the same way.

Conversion

$$\textsc{Cumul-Sort}\ \frac{\psi \vDash i\ R\ j}{\texttt{Type}_i =^R_\psi \texttt{Type}_j}$$

$$\textsc{Cumul-Prod}\ \frac{U =^=_\psi U' \qquad T =^R_\psi T'}{\Pi x : U.T =^R_\psi \Pi x : U'.T'}$$

Conversion

$$\textsc{Cumul-Sort}\ \frac{\psi \vDash i\ R\ j}{\texttt{Type}_i =^R_\psi \texttt{Type}_j} \qquad \textsc{Cumul-Prod}\ \frac{U =^=_\psi U' \qquad T =^R_\psi T'}{\Pi x : U.T =^R_\psi \Pi x : U'.T'}$$

$$\textsc{Conv-FO}\ \frac{\overrightarrow{as} =^=_\psi \overrightarrow{bs} \qquad \psi \models \overrightarrow{u} = \overrightarrow{v}}{\underline{c}_{\overrightarrow{u}}\ \overrightarrow{as} =^R_\psi \underline{c}_{\overrightarrow{v}}\ \overrightarrow{bs}}$$

Uses backtracking

Unification

Unification of id_i and id_j:

$\texttt{Definition}\ U2 := \texttt{Type}_i.$

$\texttt{Definition}\ U1 : U2 := \texttt{Type}_j \rightsquigarrow j < i$

$\texttt{Definition}\ U0 : U1 := \texttt{Type}_k \rightsquigarrow k < j$

$\texttt{Definition}\ U02 : U2 := U0 \rightsquigarrow k < i$

$$\mathrm{id}_j\ U02\ \sim\ \mathrm{id}_i\ U0 \rightsquigarrow i = j$$

But:

$$\mathrm{id}_j\ U02 \rightarrow^* (U0 \rightarrow U0) \leftarrow^* \mathrm{id}_i\ U0$$

Least-commitment principle

Use two kinds of universe level variables during elaboration:

- Polymorphic constants get elaborated with fresh flexible argument levels.
- Typical ambiguity (e.g. `Type`) creates rigid variables.
- User-given levels are rigid

Unification with universes

$t \equiv^{R}_{\psi} u \rightsquigarrow \psi'$: unification of t and u under ψ.

$$\textsc{Elab-R-FO}\ \frac{\overrightarrow{as} \equiv^{=}_{\psi} \overrightarrow{bs} \rightsquigarrow \psi' \qquad \psi' \models \overrightarrow{u} \equiv \overrightarrow{v} \rightsquigarrow \psi''}{\underline{c}_{\overrightarrow{u}}\ \overrightarrow{as} \equiv^{R}_{\psi} \underline{c}_{\overrightarrow{v}}\ \overrightarrow{bs} \rightsquigarrow \psi'}$$

Unification with universes

$t \equiv^{R}_{\psi} u \rightsquigarrow \psi'$: unification of t and u under ψ.

$$\frac{\overrightarrow{as} \equiv^{=}_{\psi} \overrightarrow{bs} \rightsquigarrow \psi' \qquad \psi' \models \overrightarrow{u} \equiv \overrightarrow{v} \rightsquigarrow \psi''}{\underline{c}_{\overrightarrow{u}}\ \overrightarrow{as} \equiv^{R}_{\psi} \underline{c}_{\overrightarrow{v}}\ \overrightarrow{bs} \rightsquigarrow \psi'}\ \textsc{Elab-R-FO}$$

$\psi \models i \equiv j \rightsquigarrow \psi'$: unification of universe instances.

$$\frac{\psi \models i = j}{\psi \models i \equiv j \rightsquigarrow \psi}\ \textsc{Elab-Univ-Eq}$$

$$\frac{i_{\mathsf{f}} \vee j_{\mathsf{f}} \in \overrightarrow{u_s} \qquad \psi \wedge i = j \models}{(\overrightarrow{u_s} \models \psi) \models i \equiv j \rightsquigarrow \psi \wedge i = j}\ \textsc{Elab-Univ-Flexible}$$

Design choice

Universe instances are levels: Suppose

$$\mathrm{id} : \forall i \vDash, \Pi A : \mathtt{Type}_i, A \to A$$

$$\Gamma = A : \mathtt{Type}_i, P : \mathrm{fibration}_{i,j} A \vdash \Sigma_{ij}\ A\ P : \mathtt{Type}_{\mathsf{max}(i,j)}$$

Levels only, adding constraint if an algebraic would appear:

$$\Gamma; \overrightarrow{u} \vDash \theta \vdash \mathrm{id}\ (\Sigma\ A\ P) \Uparrow \overrightarrow{u}, k \vDash \theta \cup \mathsf{max}(i,j) \leq k \vdash \mathrm{id}_k(\Sigma_{ij}\ A\ P) \ldots$$

Design choice

Universe instances are levels: Suppose

$$\mathrm{id} : \forall i \vDash, \Pi A : \mathtt{Type}_i, A \to A$$

$$\Gamma = A : \mathtt{Type}_i, P : \mathrm{fibration}_{i,j} A \vdash \Sigma_{ij}\ A\ P : \mathtt{Type}_{\mathsf{max}(i,j)}$$

Levels only, adding constraint if an algebraic would appear:

$$\Gamma; \overrightarrow{u} \vDash \theta \vdash \mathrm{id}\ (\Sigma\ A\ P) \Uparrow \overrightarrow{u}, k \vDash \theta \cup \mathsf{max}(i,j) \leq k \vdash \mathrm{id}_k(\Sigma_{ij}\ A\ P) \ldots$$

Minimization

That's *a lot* of fresh universe variables!!

Typical example:

$$\Gamma; \vdash \mathrm{id}\ \mathsf{true} \Uparrow \rightsquigarrow i_{\mathsf{f}} \vDash \mathtt{Set} \leq i \vdash @\mathrm{id}_i\ \mathsf{bool}\ \mathsf{true} : \mathsf{bool}$$

Minimization

That's *a lot* of fresh universe variables!!

Typical example:

$$\Gamma; \vdash \mathrm{id}\ \mathsf{true} \Uparrow \rightsquigarrow i_{\mathsf{f}} \vDash \mathtt{Set} \leq i \vdash @\mathrm{id}_i\ \mathsf{bool}\ \mathsf{true} : \mathsf{bool}$$

We'd want: $@\mathrm{id}_{\mathtt{Set}}\ \mathsf{bool}\ \mathsf{true} : \mathsf{bool}$, no new universe, no additional constraint, just as general.

Minimization

That's *a lot* of fresh universe variables!!

Typical example:

$$\Gamma; \vdash \mathrm{id}\ \mathsf{true} \Uparrow \rightsquigarrow i_{\mathsf{f}} \vDash \mathtt{Set} \leq i \vdash @\mathrm{id}_i\ \mathsf{bool}\ \mathsf{true} : \mathsf{bool}$$

We'd want: $@\mathrm{id}_{\mathtt{Set}}\ \mathsf{bool}\ \mathsf{true} : \mathsf{bool}$, no new universe, no additional constraint, just as general.

$\Rightarrow$ Minimization: compute a minimal set of universe variables.

See Cardelli's greedy algorithm for $F^{\leq}$ inference, local type inference (Pierce & Turner).

- Only applies to flexible variables.

Minimization, results

Correctness proof: easy, preservation of local solutions.

Of course this is *not* endangering the consistency of Coq!

Theorem (*Conservativity*)

Unfolding universe polymorphic definitions gives correct typings in the original system. Might just not be the most general ones if minimization did anything. For inductives, each instantiation is a new copy.

Dealing with Prop

Let $\underline{\text{false}}_i : \texttt{Type}_{i+1} \triangleq (\Pi A : \texttt{Type}_i, A : \texttt{Type}_{\mathsf{max}(i+1,i)})$.

But $\underline{\text{false}}_{\texttt{Prop}} \rightarrow^* \Pi A : \texttt{Prop}, A$, of type Prop by impredicativity (and $\texttt{Type}_{\texttt{Prop}+1}$ still).

Dealing with Prop

Let $\underline{\text{false}}_i : \texttt{Type}_{i+1} \triangleq (\Pi A : \texttt{Type}_i, A : \texttt{Type}_{\mathsf{max}(i+1,i)})$.

But $\underline{\text{false}}_{\texttt{Prop}} \rightarrow^* \Pi A : \texttt{Prop}, A$, of type Prop by impredicativity (and $\texttt{Type}_{\texttt{Prop}+1}$ still).

Fact: our universe polymorphism cannot handle precisely the implicit $\texttt{Prop} \leq \texttt{Type}$ rule (models and syntax of proof-irrelevance as well).

Ideal Solution: Use an explicit coercion.

Dealing with Prop II

Partial Solutions:

- Allow to disable Prop completely, with `-no-prop`, for HoTTists.
- Disallow instantiating a parameter level i with Prop, similar to the current Coq solution. But get less precise types.
- Currently, don't mind the problem, it's ok in practice and you don't want to lose the benefit of:

$$\underline{\text{subset}}_i\ (A : \texttt{Type}_i)\ (P : A \to \texttt{Prop}) : \texttt{Type}_i \triangleq \Sigma_{i,\texttt{Prop}}\ A\ P$$

Summary

- Reduced TCB: checking vs inference, less polymorphism-specific code.
- Avoid diffuse use of global gensym $\Rightarrow$ more functional.
- User-level control on generated universes and form of constraints (simplification, declaration...).
- Mixing polymorphic and monomorphic definitions.
- Elaboration/tactics must become universe aware.

Implementation

- Key technique: hash-consing for fast comparison of universe levels, instances and algebraic universes.
- Minimization is fast.
- Still using some naive structures. Optimize last!
- Getting stable now, to be shipped with Coq 8.5.

Benchmarks

- Runs the Homotopy Type Theory Coq library with full universe polymophism. No noticeable slowdown. Most definitions polymorphic on 6 universes at most.
- A universe polymorphic formalization of weak groupoids + an interpretation of CC in groupoids, takes 5 min to compile with polymorphism and fast projections, impossible without those two (inconsistency, exponential slowdown).

Related work

- Harper and Pollack (TCS'91). Handling of definitions and typical ambiguity in type synthesis.
- J. Courant: Explicit Universes for CC (TPHOLs'02). User-level declarations of $u \leq i$ in contexts, no other change.
- Matita (Coen et al.): checked universes, polymorphism at library level.
- Pierce and Turner (JFP): Local type inference (based on Cardelli's greedy inference algorithm).

What changes

Now:

- Universe polymorphic developments: reuse definitions and lemmas at different levels.
- Polymorphism for universes appearing *inside* structures: old discrepancy between parameters and fields.
- Rewriting with computational relations.

In the future:

- *Declaration* of universes and constraints.
- Resizing rules?

The End

Minimization II

At the end of elaboration: $\overrightarrow{i} \vDash \Theta \vdash t : T$.
Find a mimimal set of universes variables $\overrightarrow{i'} \subset \overrightarrow{i}$, universes $\overrightarrow{u}$, a substitution $\sigma : \overrightarrow{i} \to \overrightarrow{u}$ and constraints Θ' s.t. $\overrightarrow{i'} \vDash \Theta' \cup \Theta\sigma$ and $\overrightarrow{i'} \vDash \Theta\sigma \Rightarrow \Theta'$.

- First normalize the constraints w.r.t. loops ($l \leq r \wedge r \leq l$) and equalities.

Minimization II

At the end of elaboration: $\overrightarrow{i} \vDash \Theta \vdash t : T$.
Find a mimimal set of universes variables $\overrightarrow{i'} \subset \overrightarrow{i}$, universes $\overrightarrow{u}$, a substitution $\sigma : \overrightarrow{i} \to \overrightarrow{u}$ and constraints Θ' s.t. $\overrightarrow{i'} \vDash \Theta' \cup \Theta\sigma$ and $\overrightarrow{i'} \vDash \Theta\sigma \Rightarrow \Theta'$.

- First normalize the constraints w.r.t. loops ($l \leq r \wedge r \leq l$) and equalities.
- Canonicalize Θ w.r.t equalities (except globals)
- Mark i's that are fresh universe variables from universe instances as candidates for unification + restriction for universes "on the left".

Minimization II

At the end of elaboration: $\overrightarrow{i} \vDash \Theta \vdash t : T$.
Find a mimimal set of universes variables $\overrightarrow{i'} \subset \overrightarrow{i}$, universes $\overrightarrow{u}$, a substitution $\sigma : \overrightarrow{i} \to \overrightarrow{u}$ and constraints Θ' s.t. $\overrightarrow{i'} \vDash \Theta' \cup \Theta\sigma$ and $\overrightarrow{i'} \vDash \Theta\sigma \Rightarrow \Theta'$.

- First normalize the constraints w.r.t. loops ($l \leq r \wedge r \leq l$) and equalities.
- Canonicalize Θ w.r.t equalities (except globals)
- Mark i's that are fresh universe variables from universe instances as candidates for unification + restriction for universes "on the left".

Minimization III

We now have Θ with only inequality constraints and a set f of flexible universe variables.

- Let $i \in f$, compute its g.l.b: $\max(\overrightarrow{j}), j \ \mathcal{O} \ i \in \Theta$. If i has no lower constraints it must be kept.
- Generate upper constraints $\{glb \ \mathcal{O} \ j \mid i \ \mathcal{O} \ j \in \Theta\}$
- Set $i := glb$ except if glb algebraic and i has upper constraints. We can share such glbs though.