Universe Polymorphism and Inference in Coq

Matthieu Sozeau
IAS, Princeton

Coq working group
December 5th 2012
IAS, Princeton, USA

Project Team πr^2
INRIA Rocquencourt &
PPS, Paris 7 University

Universes in Coq

1. The current setup
   - Definitions
   - Issues

2. The new setup
   - Universe Polymorphic Definitions
   - The good, the bad and the ugly
   - Minimizing the ugly
   - Benchmarking
   - Demo

3. The past & the future

Kernel Type Inference

Implicit universes with cumulativity, à la Russell.
In the kernel, build up a set of universe constraints Θ.

$$\frac{\Gamma; \Theta \vdash T : \mathtt{Type}_i \rightsquigarrow \Theta_1 \qquad \Gamma, x : T; \Theta_1 \vdash U : \mathtt{Type}_j \rightsquigarrow \Theta_2}{\Gamma; \Theta \vdash \Pi x : T.U : \mathtt{Type}_{\mathsf{max}(i,j)} \rightsquigarrow \Theta_2} \ \textsc{Prod}$$

$$\frac{\begin{array}{c}\Gamma; \Theta \vdash t : U \rightsquigarrow \Theta_1 \\ \Gamma; \Theta_1 \vdash V : s \rightsquigarrow \Theta_2 \qquad \Theta_2 \vdash U \leq_{\alpha\beta\delta\iota} V \rightsquigarrow \Theta_3\end{array}}{\Gamma; \Theta \vdash t : V \rightsquigarrow \Theta_3} \ \textsc{Conv}$$

Kernel Conversion

$$\frac{}{\Theta \vdash \texttt{Type}_i \leq_{\alpha\beta\delta\iota} \texttt{Type}_j \rightsquigarrow \Theta \cup i \leq j} \ \textsc{Cumul-Sort}$$

$$\frac{\Theta \vdash U =_{\alpha\beta\delta\iota} U' \rightsquigarrow \Theta_1 \qquad \Theta_1 \vdash T \leq_{\alpha\beta\delta\iota} T' \rightsquigarrow \Theta_2}{\Theta \vdash \Pi x : U.T \leq_{\alpha\beta\delta\iota} \Pi x : U'.T' \rightsquigarrow \Theta_2} \ \textsc{Cumul-Prod}$$

Some definitions

Algebraic universes and constraints:

levels	i, j, le, lt	$\in$	$\mathbb{N}$
universes	u, v	$::=$	$i \mid \mathsf{max}(\vec{le}, \vec{lt})$
successor	$i + 1$	$::=$	$\mathsf{max}([], i)$
order	$\mathcal{O}$	$::=$	$= \mid < \mid \leq$
atomic constraint	c	$::=$	$i \ \mathcal{O} \ j$
constraints	Θ	$::=$	$\epsilon \mid c \cup \Theta$

Some definitions

Algebraic universes and constraints:

levels	i, j, le, lt	$\in$	$\mathbb{N}$
universes	u, v	$::=$	$i \mid \mathsf{max}(\vec{le}, \vec{lt})$
successor	$i + 1$	$::=$	$\mathsf{max}([], i)$
order	$\mathcal{O}$	$::=$	$= \mid < \mid \leq$
atomic constraint	c	$::=$	$i \ \mathcal{O} \ j$
constraints	Θ	$::=$	$\epsilon \mid c \cup \Theta$

The kernel only handles constraints of the form $u \ \mathcal{O} \ j$ by translation to atomic constraints:

$$\mathsf{max}(i \ j, \emptyset) \leq k \Leftrightarrow i \leq k \cup j \leq k$$

Issues

- Constraints are regenerated at each type checking
- Forces the generation of universe variables. Any term going out of the kernel must get refreshed universe variables because $\max(i, j)$ shouldn't be fed back to it.
- To implement universe polymorphism, must hack directly inside the kernel. Done for inductive types for now.

The new setup

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

Constraints are generated once at refinement time:
Inference: $\Gamma; \Psi \vdash t \Uparrow \rightsquigarrow \Psi' \vdash t' : T$
Checking: $\Gamma; \Psi \vdash t \Downarrow T \rightsquigarrow \Psi' \vdash t' : T$

The new setup

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

Constraints are generated once at refinement time:
Inference: $\Gamma; \Psi \vdash t \Uparrow \rightsquigarrow \Psi' \vdash t' : T$
Checking: $\Gamma; \Psi \vdash t \Downarrow T \rightsquigarrow \Psi' \vdash t' : T$

$$\frac{\theta \vdash \mathtt{Type}_{i+1} \leq_{\alpha\beta\delta\iota} T \rightsquigarrow \theta'}{\Gamma; us \vDash \theta \vdash \mathtt{Type} \Downarrow T \rightsquigarrow us, i \vDash \theta' \vdash \mathtt{Type}_i : T} \textsc{ Check-Type}$$

$$\frac{(\mathrm{id} : T) \in \Sigma}{\Gamma; \Psi \vdash \mathrm{id} \Uparrow \rightsquigarrow \Psi \vdash \mathrm{id} : T} \textsc{ Infer-Cst}$$

The new setup

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

Constraints are generated once at refinement time:
Inference: $\Gamma; \Psi \vdash t \Uparrow \rightsquigarrow \Psi' \vdash t' : T$
Checking: $\Gamma; \Psi \vdash t \Downarrow T \rightsquigarrow \Psi' \vdash t' : T$

$$\frac{\theta \vdash \mathtt{Type}_{i+1} \leq_{\alpha\beta\delta\iota} T \rightsquigarrow \theta'}{\Gamma; us \vDash \theta \vdash \mathtt{Type} \Downarrow T \rightsquigarrow us, i \vDash \theta' \vdash \mathtt{Type}_i : T} \; \textsc{Check-Type}$$

$$\frac{(\mathrm{id} : T) \in \Sigma}{\Gamma; \Psi \vdash \mathrm{id} \Uparrow \rightsquigarrow \Psi \vdash \mathrm{id} : T} \; \textsc{Infer-Cst}$$

- The kernel just checks constraints: $\Gamma; \Psi \vdash t : T$
- All universes and constraints that appear in the derivation (including conversions) must be in Ψ.

Universe Polymorphic definitions

Now we can introduce universe polymorphism.
Suppose a top-level definition $\mathrm{id} := t : T$.

Universe Polymorphic definitions

Now we can introduce universe polymorphism.
Suppose a top-level definition $\mathrm{id} := t : T$.

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

Universe Polymorphic definitions

Now we can introduce universe polymorphism.
Suppose a top-level definition $\mathrm{id} := t : T$.

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

Universe Polymorphic definitions

Now we can introduce universe polymorphism.
Suppose a top-level definition $\mathrm{id} := t : T$.

1. We infer T: $\Gamma; \vdash T \Uparrow \rightsquigarrow \Psi \vdash T' : s$
2. We check t: $\Gamma; \Psi \vdash t \Uparrow \rightsquigarrow i \vDash \theta \vdash t : T$
3. We can consider t, T as abstracted over the universes i and add a polymorphic definition $\mathrm{id} : \forall\ i \vDash \theta, T$ to the environment.

Universe Polymorphic definitions

To use id, we change elaboration of constants to:

$$\frac{(\mathrm{id} : \forall\ i \vDash \theta, T) \in \Sigma \qquad \vec{i'} : \vec{i} \notin \vec{u}}{\Gamma; \vec{u} \vDash \Theta \vdash \mathrm{id} \Uparrow \rightsquigarrow \vec{u}, \vec{i'} \vDash \Theta \cup \theta[\vec{i'}/\vec{i}] \vdash \mathrm{id}_{\vec{i'}} : T[\vec{i'}/\vec{i}]} \text{ INFER-CST}$$

$\Rightarrow$ Constants now carry their universe substitution/instance.
Same as Harper and Pollack (TCS'91).

Advantages to elaboration

- Reduced trusted code base: checking vs infernce. No gensym in the kernel! Reduced polymorphism-specific code.

Advantages to elaboration

- Reduced trusted code base: checking vs infernce. No gensym in the kernel! Reduced polymorphism-specific code.
- User-level control on generated universes and form of constraints (simplification, reduction...).

Advantages to elaboration

- Reduced trusted code base: checking vs infernce. No gensym in the kernel! Reduced polymorphism-specific code.
- User-level control on generated universes and form of constraints (simplification, reduction...).
- Mixing polymorphic and monomorphic definitions.

Implementation choices

Disadvantage (for me): tactics must be made universe-aware.

- Unification of id_i and id_j: Syntactic equality of i and j or add constraints? Adding equality constraints for now.

Implementation choices

Disadvantage (for me): tactics must be made universe-aware.

- Unification of id_i and id_j: Syntactic equality of i and j or add constraints? Adding equality constraints for now.
- Should universe instances be levels or algebraic universes?
  Suppose $\mathrm{id} : \forall i \vDash, \Pi A : \mathtt{Type}_i, A \to A$.
  $\Gamma = A : \mathtt{Type}_i, P : \mathrm{fibration}_{i,j} A \vdash \Sigma\ A\ P : \mathtt{Type}_{\mathsf{max}(i,j)}$

Implementation choices

Disadvantage (for me): tactics must be made universe-aware.

- Unification of id_i and id_j: Syntactic equality of i and j or add constraints? Adding equality constraints for now.
- Should universe instances be levels or algebraic universes? Suppose $\mathrm{id} : \forall i \vDash, \Pi A : \mathtt{Type}_i, A \to A$.
  $\Gamma = A : \mathtt{Type}_i, P : \mathrm{fibration}_{i,j} A \vdash \Sigma\ A\ P : \mathtt{Type}_{\mathsf{max}(i,j)}$
  Should we elaborate

  $$\Gamma \vdash \mathrm{id}_{\mathsf{max}(i,j)}(\Sigma\ A\ P) : \Sigma\ A\ P \to \Sigma\ A\ P$$

  $\Rightarrow$ Currently levels only, adding constraint if an algebraic would appear:

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

Minimization I

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

Typical example:

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

Minimization I

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

Typical example:

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

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

Minimization I

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

Typical example:

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

We'd want: $@\mathrm{id}_{\mathtt{Set}}$ bool true : bool, no new universe, no additional constraint.
Requires no upper constraints on i ($i\ \mathcal{O}\ j$).

$\Rightarrow$ Minimization

Minimization II

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

- Mark i's that are fresh universe variables as candidates for unification + restriction for universes "on the left".

Minimization II

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

- Mark i's that are fresh universe variables as candidates for unification + restriction for universes “on the left”.
- Canonicalize Θ w.r.t equalities (except globals)

Minimization III

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

- Let $i \in f$, compute its l.u.b.: $\max(\vec{j}), j \; \mathcal{O} \; i \in \Theta$. If i has no lower constraints it must be kept.
- Generate upper constraints $\{lub \; \mathcal{O} \; j \mid i \; \mathcal{O} \; j \in \Theta\}$
- Substitute lub/i except if lub algebraic and i appears "on the left"

Minimization IV

Correctness proof: TODO but rather hopeful.

This is *not* endangering the consistency of Coq!

Minimization IV

Correctness proof: TODO but rather hopeful.

This is *not* endangering the consistency of Coq!
You may ask why.

Minimization IV

Correctness proof: TODO but rather hopeful.

This is *not* endangering the consistency of COQ!
You may ask why.

1. The kernel takes $\Gamma; \vec{i} \vDash \Theta \vdash t : T$ in.
2. It retypechecks and generates $\Gamma \vdash t : T \rightsquigarrow \Theta'$.
3. It does *not* check $\Theta \Rightarrow \Theta'$: it ignores Θ.

Minimization IV

Correctness proof: TODO but rather hopeful.

This is *not* endangering the consistency of COQ!
You may ask why.

1. The kernel takes $\Gamma; \vec{i} \vDash \Theta \vdash t : T$ in.
2. It retypechecks and generates $\Gamma \vdash t : T \rightsquigarrow \Theta'$.
3. It does *not* check $\Theta \Rightarrow \Theta'$: it ignores Θ.

$\Rightarrow$ Minimization only minimizes the number of universe variables for now, not the constraints.

Minimization IV

Correctness proof: TODO but rather hopeful.

This is *not* endangering the consistency of Coq!
You may ask why.

1. The kernel takes $\Gamma; \vec{i} \vDash \Theta \vdash t : T$ in.
2. It retypechecks and generates $\Gamma \vdash t : T \rightsquigarrow \Theta'$.
3. It does *not* check $\Theta \Rightarrow \Theta'$: it ignores Θ.

$\Rightarrow$ Minimization only minimizes the number of universe variables for now, not the constraints.

- Safe: kernel is just as safe as before. Will tell if minimization produced an inconsistency by e.g. equating some universes.
- Inefficient: could get smaller constraints than the inferred ones.

Benchmarks

Implementation still in progress but:

- On the stdlib: with pair, Σ, projections and list declared polymorphic. No noticeable change.
- On HoTT/HoTT with full universe polymophism: WIP (universe inconsistency in `UnivalenceImpliesFunext.v`). No noticeable change. Most definitions polymorphic on 6 universes at most.

Demo

Related work

- Harper and Pollack (TCS'91). Handling of definitions and typical ambiguity in type synthesis. Just slightly different from elaboration.
- J. Courant: Explicit Universes for CC (TPHOLs'02). User-level declarations of $i \leq u$ in contexts, no other change.
- Matita (Coen et al.): checked universes, polymorphism at library level.

Nice things that become possible

- Universe polymorphic structures: category of categories.
- Polymorphism for universes appearing *inside* structures: old discrepancy between parameters and fields.

Nice things that become possible

- Universe polymorphic structures: category of categories.
- Polymorphism for universes appearing *inside* structures: old discrepancy between parameters and fields.
- Computational relations and rewriting: Make Proper polymorphic and use relations in `Type`. Long standing limitation, e.g. for MathClasses. Useful for HoTT as well.
- Let use declare all universes and constraints?
- Resizing rules.

The End

That's all folks!