Universe Polymorphism and Inference in Coq

Matthieu Sozeau
IAS, Princeton

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

Project Team πτ²
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
Implicit universes with cumulativity, à la Russell.

In the kernel, build up a set of universe constraints $\Theta$.

$$\Gamma; \Theta \vdash T : \text{Type}_i \rightsquigarrow \Theta_1 \quad \Gamma, x : T; \Theta_1 \vdash U : \text{Type}_j \rightsquigarrow \Theta_2 \quad \text{PROD}$$

$$\Gamma; \Theta \vdash \Pi x : T.U : \text{Type}_{\max(i,j)} \rightsquigarrow \Theta_2$$

$$\Gamma; \Theta \vdash t : U \rightsquigarrow \Theta_1$$

$$\Gamma; \Theta_1 \vdash V : s \rightsquigarrow \Theta_2 \quad \Theta_2 \vdash U \leq_{\alpha\beta\delta\iota} V \rightsquigarrow \Theta_3 \quad \text{CONV}$$

$$\Gamma; \Theta \vdash t : V \rightsquigarrow \Theta_3$$
\[
\frac{\Theta \vdash \text{Type}_i \leq_{\alpha\beta\delta_i} \text{Type}_j \rightsquigarrow \Theta \cup i \leq j}{\text{CUMUL-SORT}}
\]

\[
\frac{\Theta \vdash U =_{\alpha\beta\delta_i} U' \rightsquigarrow \Theta_1 \quad \Theta_1 \vdash T \leq_{\alpha\beta\delta_i} T' \rightsquigarrow \Theta_2}{\Theta \vdash \Pi x : U.T \leq_{\alpha\beta\delta_i} \Pi x : U'.T' \rightsquigarrow \Theta_2 \quad \text{CUMUL-PROD}}
\]
Algebraic universes and constraints:

- **levels**
  - $i, j, le, lt \in \mathbb{N}$

- **universes**
  - $u, v ::= i \mid \max(\vec{le}, \vec{lt})$

- **successor**
  - $i + 1 ::= \max(\emptyset, i)$

- **order**
  - $\mathcal{O} ::= = \mid < \mid \leq$

- **atomic constraint**
  - $c ::= i \mathcal{O} j$

- **constraints**
  - $\Theta ::= \epsilon \mid c \cup \Theta$
Some definitions

Algebraic universes and constraints:

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

The kernel only handles constraints of the form \( u O j \) by translation to atomic constraints:

\[
\text{max}(i, j, \emptyset) \leq k \iff i \leq k \cup j \leq k
\]
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.
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
The new setup

universe context \( \Psi ::= \vec{i} \models \Theta \)

Constraints are generated once at refinement time:

Inference: \( \Gamma; \Psi \vdash t \uparrow \leadsto \Psi' \vdash t' : T \)

Checking: \( \Gamma; \Psi \vdash t \downarrow T \leadsto \Psi' \vdash t' : T \)
The new setup

universe context \( \Psi ::= \vec{i} \models \emptyset \)

Constraints are generated once at refinement time:

\begin{align*}
\text{Inference: } & \Gamma; \Psi \vdash t \uparrow \rightsquigarrow \Psi' \vdash t' : T \\
\text{Checking: } & \Gamma; \Psi \vdash t \downarrow T \rightsquigarrow \Psi' \vdash t' : T
\end{align*}

\[
\frac{\theta \vdash \text{Type}_{i+1} \leq_{\alpha\beta\delta_i} T \rightsquigarrow \theta'}{\Gamma; us \vdash \theta \vdash \text{Type} \downarrow T \rightsquigarrow us, i \vdash \theta' \vdash \text{Type}_i : T} \quad \text{CHECK-TYPE}
\]

\[
\frac{(id : T) \in \Sigma}{\Gamma; \Psi \vdash id \uparrow \rightsquigarrow \Psi \vdash id : T} \quad \text{INFER-CST}
\]
The new setup

universe context  \( \Psi ::= \vec{i} \vdash \Theta \)

Constraints are generated once at refinement time:
Inference:  \( \Gamma; \Psi \vdash t \uparrow \rightsquigarrow \Psi' \vdash t' : T \)
Checking:  \( \Gamma; \Psi \vdash t \downarrow T \rightsquigarrow \Psi' \vdash t' : T \)

\[
\begin{align*}
\theta \vdash \text{Type}_{i+1} \leq_{\alpha \beta \delta_i} T & \rightsquigarrow \theta' \\
\Gamma; \mathit{us} \vdash \theta \vdash \text{Type} \downarrow T & \rightsquigarrow \mathit{us}, i \vdash \theta' \vdash \text{Type}_i : T
\end{align*}
\]

\[
\sigma \vdash \text{Infer-Cst} \quad \text{Check-Type}
\]

\[
\begin{align*}
\mathit{id} : T \in \Sigma \\
\Gamma; \Psi \vdash \mathit{id} \uparrow \rightsquigarrow \Psi \vdash \mathit{id} : T
\end{align*}
\]

\(\Gamma; \Psi \vdash t : T\)

\(\Gamma; \Psi \vdash t : T\)

The kernel just checks constraints:  \( \Gamma; \Psi \vdash t : T \)

All universes and constraints that appear in the derivation (including conversions) must be in \( \Psi \).
Now we can introduce universe polymorphism. Suppose a top-level definition $id := t : T$. 
Now we can introduce universe polymorphism. Suppose a top-level definition \( \text{id} := t : T \).

1. We infer \( T : \Gamma \vdash T \uparrow \sim \Psi \vdash T' : s \).
Now we can introduce universe polymorphism. Suppose a top-level definition 

\[ \text{id} := t : T. \]

1. We infer \( T : \Gamma; \vdash T \uparrow \leadsto \Psi \vdash T' : s \)
2. We check \( t : \Gamma; \Psi \vdash t \uparrow \leadsto i \models \theta \vdash t : T \)
Now we can introduce universe polymorphism. Suppose a top-level definition \( \text{id} := t : T \).

1. We infer \( T : \Gamma; \vdash T \uparrow \rightsquigarrow \Psi \vdash T' : s \)
2. We check \( t : \Gamma; \Psi \vdash t \uparrow \rightsquigarrow i \models \theta \vdash t : T \)
3. We can consider \( t, T \) as abstracted over the universes \( i \) and add a polymorphic definition \( \text{id} : \forall i \models \theta, T \) to the environment.
To use \texttt{id}, we change elaboration of constants to:

\[
\frac{(\texttt{id} : \forall i \models \theta, T) \in \Sigma \quad \bar{i} : i \notin \bar{u}}{
\Gamma; \bar{u} \models \Theta \vdash \texttt{id} \uparrow \leadsto \bar{u}, \bar{i} \models \Theta \cup \theta[\bar{i}/i] \vdash \texttt{id}_{\bar{i}/i} : T[\bar{i}/i]}
\]

\textbf{Infer-Cst}

\[\Rightarrow \text{Constants now carry their universe substitution-instance.} \]
\[\text{Same as Harper and Pollack (TCS'91).}\]
Advantages to elaboration

- Reduced trusted code base: checking vs inference. No gensym in the kernel! Reduced polymorphism-specific code.
Advantages to elaboration

- Reduced trusted code base: checking vs inference. 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 inference. 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.
Disadvantage (for me): tactics must be made universe-aware.

- Unification of $\text{id}_i$ and $\text{id}_j$: 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 $\text{id}_i$ and $\text{id}_j$: Syntactic equality of $i$ and $j$ or add constraints? Adding equality constraints for now.

▶ Should universe instances be levels or algebraic universes?

Suppose $\text{id} : \forall i \models \Pi A : \text{Type}_i, A \rightarrow A$.

$\Gamma = A : \text{Type}_i, P : \text{fibration}_{i,j} A \vdash \Sigma A \ P : \text{Type}_{\text{max}(i,j)}$
Implementation choices

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

- Unification of $\text{id}_i$ and $\text{id}_j$: Syntactic equality of $i$ and $j$ or add constraints? Adding equality constraints for now.

- Should universe instances be levels or algebraic universes?
  Suppose $\text{id} : \forall i \vdash \Pi A : \text{Type}_i, A \to A$.
  $\Gamma = A : \text{Type}_i, P : \text{fibration}_{i,j} A \vdash \Sigma A P : \text{Type}_{\max(i,j)}$
  Should we elaborate

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

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

  $$\Gamma; \vec{u} \vdash \theta \vdash \text{id} (\Sigma A P) \upharpoonright \vec{u}, k \vdash \theta \cup \max(i, j) \leq k \vdash \text{id}_k(\Sigma A P) \ldots$$
That’s a lot of fresh universe variables!!

Typical example:

\[ \Gamma; \Psi \vdash \text{id \ true} \uparrow \leadsto \Psi \cup i \models \text{Set} \leq i \vdash @\text{id}_i \text{ bool true : bool} \]
That’s a lot of fresh universe variables!!

Typical example:

\[ \Gamma; \Psi \vdash \text{id \ true} \uparrow \leadsto \Psi \cup i \models \text{Set} \leq i \vdash \@\text{id}_i \text{ bool true} : \text{bool} \]

We’d want: \( \@\text{id}_{\text{Set}} \text{ bool true} : \text{bool} \), no new universe, no additional constraint.
That’s a lot of fresh universe variables!!

Typical example:

$$\Gamma; \Psi \vdash \text{id } \text{true} \uparrow \rightsquigarrow \Psi \cup i \models \text{Set} \leq i \vdash @\text{id}_i \text{bool } \text{true} : \text{bool}$$

We’d want: $@\text{id}_{\text{Set}} \text{bool } \text{true} : \text{bool}$, no new universe, no additional constraint. Requires no upper constraints on $i$ ($i \triangleleft j$).

$$\Rightarrow \text{Minimization}$$
At the end of elaboration: \( \vec{i} \models \Theta \vdash t : T \).

Find a minimal set of universes variables \( \vec{i}' \subset \vec{i} \), universes \( \vec{u} \), a substitution \( \sigma : \vec{i} \rightarrow \vec{u} \) and constraints \( \Theta' \) s.t. \( \vec{i}' \models \Theta' \cup \Theta \sigma \) and \( \vec{i}' \models \Theta \sigma \Rightarrow \Theta' \).

- Mark \( i' \)'s that are fresh universe variables as candidates for unification + restriction for universes “on the left”.
At the end of elaboration: $\vec{i} \vdash \Theta \vdash t : T$.

Find a minimal set of universes variables $\vec{i}' \subset \vec{i}$, universes $\vec{u}$, a substitution $\sigma : \vec{i} \rightarrow \vec{u}$ and constraints $\Theta'$ 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 $\Theta$ w.r.t equalities (except globals)
We now have $\Theta$ with only inequality constraints and a set $f$ of “flexible” universe variables.

- Let $i \in f$, compute its l.u.b.: $\max(\vec{j}), j \ O i \in \Theta$. If $i$ has no lower constraints it must be kept.
- Generate upper constraints $\{\text{lub } O j \mid i \ O j \in \Theta\}$
- Substitute $\text{lub} / i$ except if $\text{lub}$ algebraic and $i$ appears “on the left”
Correctness proof: TODO but rather hopeful.

This is *not* endangering the consistency of Coq!
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 $\Gamma \vdash t : T \leadsto \Theta'$.  
3. It does not check $\Theta \Rightarrow \Theta'$: it ignores $\Theta$. 
Minimization IV

Correctness proof: TODO but rather hopeful.

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

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

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

⇒ 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 polymorphism: WIP (universe inconsistency in `UnivalenceImpliesFunext.v`). No noticeable change. Most definitions polymorphic on 6 universes at most.
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
Related work

- 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.
That’s all folks!