Universe Polymorphism and Inference in Coq
Work in progress
Matthieu Sozeau

Types 2013
April 25th 2013
Toulouse

Project Team π²
INRIA & PPS, Paris 7
IAS, Princeton
Without polymorphism

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

$\vdash \text{id} : \Pi (A : \text{Type}_l), \ A \rightarrow A : \text{Type}_{\max (l+1, l)}$

$\not\vdash \text{id} (\Pi (A : \text{Type}_l), \ A \rightarrow A) \ \text{id} : (\Pi (A : \text{Type}_l), \ A \rightarrow A)$
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
   - Benchmarks

3. The past & the future
Implicit universes with cumulativity, à la Russell.

_In the kernel_, build up a set of universe constraints $\Theta$.

\[
\frac{
\Gamma; \Theta \vdash T : \text{Type}_i \rightsquigarrow \Theta_1 \\
\Gamma, x : T; \Theta_1 \vdash U : \text{Type}_j \rightsquigarrow \Theta_2 
}{
\Pi_{x : T.U : \text{Type}^\text{max}(i,j)} \rightsquigarrow \Theta_2 
} \quad \text{PROD}
\]

\[
\frac{
\Gamma; \Theta \vdash t : U \rightsquigarrow \Theta_1 \\
\Gamma; \Theta_1 \vdash V : s \rightsquigarrow \Theta_2 \\
\Theta_2 \vdash U \leq V \rightsquigarrow \Theta_3 
}{
\Gamma; \Theta \vdash t : V \rightsquigarrow \Theta_3 
} \quad \text{CONV}
\]
\[ \Theta \vdash \text{Type}_i \leq \text{Type}_j \leadsto \Theta \cup i \leq j \quad \text{Cumul-Sort} \]

\[ \Theta \vdash U = U' \leadsto \Theta_1 \quad \Theta_1 \vdash T \leq T' \leadsto \Theta_2 \quad \text{Cumul-Prod} \]

\[ \Theta \vdash \Pi x : U.T \leq \Pi x : U'.T' \leadsto \Theta_2 \]
## Some definitions

Algebraic universes and constraints:

<table>
<thead>
<tr>
<th>Concept</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>levels</td>
<td>$i, j, le, lt \in \mathbb{N} \cup {\text{Prop, Set}}$</td>
</tr>
<tr>
<td>universes</td>
<td>$u, v ::= i \mid \max(\vec{le}, \vec{lt})$</td>
</tr>
<tr>
<td>successor</td>
<td>$i + 1 ::= \max([], i)$</td>
</tr>
<tr>
<td>order</td>
<td>$O ::= = \mid &lt; \mid \le$</td>
</tr>
<tr>
<td>atomic constraint</td>
<td>$c ::= i O j$</td>
</tr>
<tr>
<td>constraints</td>
<td>$\Theta ::= \epsilon \mid c \cup \Theta$</td>
</tr>
</tbody>
</table>
Some definitions

Algebraic universes and constraints:

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

Only handles constraints of the form $u O j$ by translation to atomic constraints:

$$\text{max}(i j, k) \leq l \iff i \leq l \cup j \leq l \cup k < l$$

Invariant on typing ensures this is the shape of inferred constraints (Herbelin, TYPES).
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
   - Benchmarks

3. The past & the future
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: $\Gamma; \Psi \vdash t \downarrow T \leadsto \Psi' \vdash t' : T$
universe context \( \Psi ::= \overrightarrow{i} \vdash \Theta \)

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

\[
\frac{\theta \vdash \text{Type}_{i+1} \leq T \leadsto \theta'}{\Gamma; us \vdash \theta \vdash \text{Type} \downarrow T \leadsto us, i \vdash \theta' \vdash \text{Type}_i : T} \quad \text{CHECK-TYPE}
\]

\[
\frac{(id : T) \in \Sigma}{\Gamma; \Psi \vdash id \uparrow \leadsto \Psi \vdash id : T} \quad \text{INFER-CST}
\]
Constraint checking

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

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

\(\begin{align*}
\theta \vdash \text{Type}_{i+1} \leq T \leadsto \theta' \\
\Gamma; us \vdash \theta \vdash \text{Type} \downarrow T \leadsto us, i \vdash \theta' \vdash \text{Type}_i : T
\end{align*}\)

**Check-Type**

\(\begin{align*}
(id : T) \in \Sigma \\
\Gamma; \Psi \vdash id \uparrow \leadsto \Psi \vdash id : T
\end{align*}\)

**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 \( \Psi \).
Now we can introduce universe polymorphism. Suppose a top-level definition $\text{id} := t : T$. 
Now we can introduce universe polymorphism. Suppose a top-level definition \( \text{id} := t : T \).

1. \( \Gamma ; \vdash T \uparrow \leadsto \Psi \vdash T' : s \)
Universe Polymorphic definitions

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

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

1. $\Gamma ; \vdash T \uparrow \leadsto \Psi ; \vdash T' : s$
2. $\Gamma ; \Psi \vdash t \downarrow T' \leadsto i \models \theta \vdash t : T'$
3. Add $\text{id} : \forall i \models \theta, T' := t$ to the environment.

⇒ Guiding principle: constants are transparent, indistinguishable from their bodies.
To use \( \text{id} \), we change elaboration of constants to:

\[
\frac{(\text{id} : \forall i \vdash \theta, T) \in \Sigma \quad \vec{i}' : i \notin \vec{u}}{
\Gamma; \vec{u} \vdash \Theta \vdash \text{id} \uparrow \rightsquigarrow \vec{u}, \vec{i}' \vdash \Theta \cup \theta[\vec{i}' / \vec{i}] \vdash \text{id}_{\vec{i}'} : T[\vec{i}' / \vec{i}]}
\]

\( \Rightarrow \) Constants now carry their universe substitution/instance.

\( \Rightarrow \) Inductives and constructors treated the same way.
Advantages to elaboration

- Reduced trusted code base: checking vs inference. Avoid global gensym. Reduced polymorphism-specific code.
Advantages to elaboration

- Reduced trusted code base: checking vs inference. Avoid global gensym. Reduced polymorphism-specific code.
- User-level control on generated universes and form of constraints (simplification, declaration...).
Advantages to elaboration

- Reduced trusted code base: checking vs inference. Avoid global gensym. Reduced polymorphism-specific code.
- User-level control on generated universes and form of constraints (simplification, declaration...).
- Mixing polymorphic and monomorphic definitions.
Disadvantage (for me): unification and tactics must be universe-aware.

- Universe instances are levels: Suppose

\[ \text{id} : \forall i \models, \Pi A : \text{Type}_i, A \to A \]

\[ \Gamma = A : \text{Type}_i, P : \text{fibration}_{i,j} A \vdash \Sigma_{i,j} A P : \text{Type}_{\max(i,j)} \]

Levels only, adding constraint if an algebraic would appear:

\[ \Gamma; \vec{u} \models \theta \vdash \text{id} (\Sigma A P) \upharpoonright \vec{u}, k \models \theta \cup \max(i, j) \leq k \vdash \text{id}_k (\Sigma_{i,j} A P) \ldots \]
Design choices

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

Universe instances are levels: Suppose

\[ \text{id} : \forall i \models, \Pi A : \text{Type}_i, A \rightarrow A \]

\[ \Gamma = A : \text{Type}_i, P : \text{fibration}_{i,j} A \vdash \Sigma_{i,j} A \ P : \text{Type}_{\max(i,j)} \]

Levels only, adding constraint if an algebraic would appear:

\[ \Gamma; \bar{u} \models \theta \vdash \text{id} (\Sigma A \ P) \upharpoonright \bar{u}, k \models \theta \cup \max(i, j) \leq k \vdash \text{id}_k (\Sigma_{i,j} A \ P) \ldots \]

Unification of \( \text{id}_i \) and \( \text{id}_j \): Syntactic equality of \( i \) and \( j \)? Do nothing? Adding equalities for now: can break transparency.
Minimization

That’s a lot of fresh universe variables!!

Typical example:

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

That’s a lot of fresh universe variables!!

Typical example:

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

We’d want: \( \text{@id}_{\text{Set}} \text{ bool true : bool} \), no new universe, no additional constraint, just as general.
Minimization

That’s a lot of fresh universe variables!!

Typical example:

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

We’d want: \( \text{id}_{\text{Set}} \) \text{bool} \ \text{true} : \text{bool}, \) no new universe, no additional constraint, just as general.
Requires no upper constraints on \( i (i \not< j) \).

⇒ Minimization: compute a minimal set of universe variables.

See Cardelli’s greedy algorithm for \( F \leq \) inference, local type inference (Pierce & Turner).
Correctness proof: WIP.

This is *not* endangering the consistency of CoQ!

We have *conservativity*: unfolding universe polymorphic definitions gives correct typings in the original system. Might just not be the most general ones.
First-order unification of universes

Due to (notoriously heuristic) first-order unification/conversion of constants...we get too strict universe constraints.

Definition $U_2 := \text{Type}_i$.
Definition $U_1 : U_2 := \text{Type}_j \rightsquigarrow j < i$
Definition $U_0 : U_1 := \text{Type}_k \rightsquigarrow k < j$
Definition $U_{02} : U_2 := U_0 \rightsquigarrow k < i$

\[
\text{id}_j\ U_{02} \sim \text{id}_i\ U_0 \rightsquigarrow i = j
\]

But: $\text{id}_j\ U_{02} \rightarrow^* (U_0 \rightarrow U_0)$ and $\text{id}_i\ U_0 \rightarrow^* (U_0 \rightarrow U_0)$

$\Rightarrow$ Analyse variance of universes to relax first-order unification.

Varicance can help minimization too.

E.g. for $\text{id}_i\ t \sim \text{id}_j\ u$, $i$ and $j$ do not have to be compared.
Implementation still in progress but:

- Runs the Homotopy Type Theory Coq library with full universe polymorphism. No noticeable slowdown. Most definitions polymorphic on 6 universes at most.
- A very generic formalization of weak 2-groupoids + an interpretation of CC in 2-groupoids.
Universe polymorphism and inference in Coq, Work in progress

1. The current setup
   - Definitions
   - Issues

2. The new setup
   - Universe polymorphic definitions
   - The good, the bad and the ugly
   - Minimizing the ugly
   - Benchmarks

3. The past & the future
Related work

- 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.
Nice things that become possible

- Universe polymorphic developments: reuse definitions and lemmas at different levels.
- Polymorphism for universes appearing *inside* structures: old discrepancy between parameters and fields.
- Computational relations and rewriting: Long standing limitation, e.g. for MathClasses. Useful for HoTT as well.
- Let us *declare* universes and constraints.
- Resizing rules.
That’s all folks!
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 $\vec{i}' \models \Theta\sigma \Rightarrow \Theta'$.

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

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 $\vec{i}' \models \Theta \sigma \Rightarrow \Theta'$.

- First normalize the constraints w.r.t. loops $(l \leq r \land r \leq l)$ and equalities.
- Canonicalize $\Theta$ 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”.
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 $\vec{i}' \models \Theta \sigma \Rightarrow \Theta'$.

- First normalize the constraints w.r.t. loops ($l \leq r \land r \leq l$) and equalities.
- Canonicalize $\Theta$ 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”.
We now have $\Theta$ with only inequality constraints and a set $f$ of flexible universe variables.

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