Universe Polymorphism and Inference in Coq

Work in progress

Matthieu Sozeau

Project Team πr^2
INRIA & PPS, Paris 7
IAS, Princeton

TYPES 2013
April 25th 2013
Toulouse

Without polymorphism

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

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

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

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

Kernel Type Inference

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

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

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

Kernel Conversion

$$\frac{}{\Theta \vdash \mathtt{Type}_i \leq \mathtt{Type}_j \rightsquigarrow \Theta \cup i \leq j} \textsc{Cumul-Sort}$$

$$\frac{\Theta \vdash U = U' \rightsquigarrow \Theta_1 \qquad \Theta_1 \vdash T \leq T' \rightsquigarrow \Theta_2}{\Theta \vdash \Pi x : U.T \leq \Pi x : U'.T' \rightsquigarrow \Theta_2} \textsc{Cumul-Prod}$$

Some definitions

Algebraic universes and constraints:

levels	i, j, le, lt	$\in$	$\mathbb{N} \cup \{\texttt{Prop}, \texttt{Set}\}$
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	$\in$	$\mathbb{N} \cup \{\texttt{Prop}, \texttt{Set}\}$
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	$::=$	$\epsilon \mid c \cup \Theta$

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

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

Invariant on typing ensures this is the shape of infered 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

Constraint checking

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

Constraints are generated once at refinement time:

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

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

Constraint checking

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

Constraints are generated once at refinement time:

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

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

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

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

Constraint checking

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

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

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

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

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

Universe Polymorphic definitions

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

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

Universe Polymorphic definitions

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

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

$\Rightarrow$ Guiding principle: constants are *transparent*, indistinguishable from their bodies.

Universe Polymorphic definitions

To use id, we change elaboration of constants to:

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

$\Rightarrow$ 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.

Design choices

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

- 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:

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

Design choices

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

- 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:

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

- Unification of id_i and 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:

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

Minimization

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, just as general.

Minimization

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, just as general.
Requires no upper constraints on i ($i\ \mathcal{O}\ j$).

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

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

Minimization

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.

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

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

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

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

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

But: $\mathrm{id}_j\ U02 \to^* (U0 \to U0)$ and $\mathrm{id}_i\ U0 \to^* (U0 \to U0)$

$\Rightarrow$ Analyse *variance* of universes to relax first-order unification.
Variance can help minimization too.

E.g. for $\mathrm{id}_i\ t \sim \mathrm{id}_j\ u$, i and j do not have to be compared.

Benchmarks

Implementation still in progress but:

- Runs the Homotopy Type Theory Coq library with full universe polymophism. 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.

Universes in Coq

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.

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.

The End

That's all folks!

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

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

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

- 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: $\vec{i} \vDash \Theta \vdash t : T$.
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 $\vec{i'} \vDash \Theta\sigma \Rightarrow \Theta'$.

- 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: $\max(\vec{j}), j \ \mathcal{O} \ i \in \Theta$. 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.