Universe Polymorphism: Subtyping and Unification

Matthieu Sozeau

Project Team πr^2
INRIA & PPS, Paris 7

Agda Implementers Meeting
May 27th 2014
Paris

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 universes $\mathtt{Type}_0 < \mathtt{Type}_1 < \ldots \mathtt{Type}_n$ used to avoid the $\mathtt{Type} : \mathtt{Type}$ paradox (system U^-, first version of Martin-Löf Type Theory (MLTT)).

- Replicates Russell's paradox of $\{x \mid x \notin x\}$, the set of all sets etc....
- You can think of $\mathtt{Type}_0$ as sets, $\mathtt{Type}_1$ as classes etc...

Coq's theory

We call the sort of t the type of the type of t, it is necessarily a $\texttt{Type}_i$ (see e.g. Type-prod).

$$\frac{\vdash \Gamma \quad (i \in \mathbb{N})}{\Gamma \vdash \texttt{Type}_i : \texttt{Type}_{i+1}}\ \textsc{Type-intro} \qquad \frac{\Gamma \vdash A : \texttt{Type}_i \qquad \Gamma, x : A \vdash B : \texttt{Type}_j}{\Gamma \vdash \Pi x : A.B : \texttt{Type}_{\mathsf{max}(i,j)}}\ \textsc{Type-prod}$$

For our purposes, think of Coq's type theory as a stratified, predicative System F + primitive inductive/algebraic datatypes. The dependent quantification on terms is not relevant here.

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 works (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.

In practice, translation from a source language with anonymous `Type`s to a core theory with $\texttt{Type}_i$s only.

Floating universes

But in general many i's can work!

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

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

or

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

or ...

$\Rightarrow$ Allow *universe variables*.

Floating universes and constraints

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

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

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

Floating universes and constraints II

Proposition (Correctness)

If $\Gamma \vdash_{float} t : T \leadsto \Theta$ *and* Θ *is satisfiable (with assignemnt* σ*) then* $\Gamma[\sigma] \vdash t[\sigma] : T[\sigma]$*.*

Constraints $\Rightarrow$ graph structure based on union-find. $l < l'$ is an arc, consistency is cycle-freeness, easy to check incrementaly.

Without polymorphism

Floating levels give a false sense of polymorphism:

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

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

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

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

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

With polymorphism

Real, bounded polymorphism:

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

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

$\Rightarrow$ l is quantified at the definition level now, but second-class, we need *instantiation*:

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

Kernel Type Inference

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

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

$$\frac{\Gamma; \Theta \vdash t : U \leadsto \Theta_1 \qquad \Gamma; \Theta_1 \vdash V : s \leadsto \Theta_2 \qquad \Theta_2 \vdash U \leq V \leadsto \Theta_3}{\Gamma; \Theta \vdash t : V \leadsto \Theta_3}\ \textsc{Conv}$$

Kernel Conversion

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

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

Some definitions

Algebraic universes and constraints:

levels	i, j, le, lt	$\in$	$\mathbb{N} \cup \{\texttt{Prop}, \texttt{Set}\}$
universes	u, v	$::=$	$i \mid \mathsf{max}(\overrightarrow{le}, \overrightarrow{lt})$
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	$::=$	$i \mid \mathsf{max}(\overrightarrow{le}, \overrightarrow{lt})$
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 global, unorderly 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.

Constraint checking

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

Constraints are generated once at refinement time (outside the kernel):

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

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

Constraints are generated once at refinement time (outside the kernel):

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

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

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

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

Constraint checking

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

Constraints are generated once at refinement time (outside the kernel):

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

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

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

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

- The kernel just checks constraints: $\Gamma; \Psi \vdash t : T$
- All universes and constraints that appear in the derivation

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. $\Gamma; \Psi \vdash t \Downarrow T' \rightsquigarrow i \models \theta \vdash t : T'$

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:

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

⇒ Constants now carry their universe substitution/instance.

⇒ Inductives and constructors treated the same way.

Universe Polymorphic definitions: conversion

$$\text{R-}\delta\text{-L}\;\frac{\underline{\mathrm{c}}_{\vec{i}} \to_\delta t \qquad t\ \vec{a} =^R_\psi u}{\underline{\mathrm{c}}_{\vec{i}}\ \vec{a} =^R_\psi u} \qquad \text{R-}\delta\text{-R}\;\frac{\underline{\mathrm{c}}_{\vec{i}} \to_\delta u \qquad t =^R_\psi u\ \vec{a}}{t =^R_\psi \underline{\mathrm{c}}_{\vec{i}}\ \vec{a}}$$

$$\text{R-FO}\;\frac{\vec{as} =^{=}_\psi \vec{bs} \qquad \color{red}{\psi \models \vec{u} = \vec{v}}}{\underline{\mathrm{c}}_{\vec{u}}\ \vec{as} =^R_\psi \underline{\mathrm{c}}_{\vec{v}}\ \vec{bs}}$$

Advantages to elaboration

- Reduced trusted code base: checking vs inference.
- Reduced polymorphism-specific code (actually no, thanks to backward compatibility).
- 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.

Design choices

Disadvantage (for me and some of you): unification and tactics must become universe-aware.

```
Universes.constr_of_global :
  global_reference -> constr in_universe_context
```

Unification of id_i and id_j: Syntactic equality of i and j? Do nothing?

First-order unification of universes: a first bold idea

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

`Definition` $U2 := \texttt{Type}_i$.
`Definition` $U1 : U2 := \texttt{Type}_j \rightsquigarrow j < i$
`Definition` $U0 : U1 := \texttt{Type}_k \rightsquigarrow k < j$
`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 could help minimization too.

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

$\Rightarrow$ But it looks like the analysis is hard, not compositional and requires full normalisation of the constant bodies.

Pragmatic, heuristic solution

```
Evd.fresh_global : ?rigid:rigid -> env -> evar_map ->
  global_reference -> evar_map * constr

type rigid =
  | UnivRigid
  | UnivFlexible of bool (* can be algebraic? *)
```

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

Unification with universes

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

$$\textsc{Elab-R-}\delta\textsc{-left}\quad \frac{\underline{\mathrm{c}}_{\vec{i}} \to_\delta t \qquad t\ \vec{a} \equiv_{\psi}^{R} u \rightsquigarrow \psi'}{\underline{\mathrm{c}}_{\vec{i}}\ \vec{a} \equiv_{\psi}^{R} u \rightsquigarrow \psi'}$$

$$\textsc{Elab-R-}\delta\textsc{-right}\quad \frac{\underline{\mathrm{c}}_{\vec{i}} \to_\delta u \qquad t \equiv_{\psi}^{R} u\ \vec{a} \rightsquigarrow \psi'}{t \equiv_{\psi}^{R} \underline{\mathrm{c}}_{\vec{i}}\ \vec{a} \rightsquigarrow \psi'}$$

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

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

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

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

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

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

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

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:

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

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.

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

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

- Requires no other lower constraints on i $(j \not\mathcal{O}\ i)$.
- 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 $\mathtt{Prop}$

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

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

Dealing with $\texttt{Prop}$

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

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

Fact: our universe polymorphism is *incompatible* with the implicit use of impredicativity (which has computational content according to homotopy models, see hProp's in the HoTT book). The implicit $\texttt{Prop} \leq \texttt{Type}$ rule also causes problems for models and syntax of proof-irrelevance (Werner *et al*)...

Ideal Solution: Let's get the Rooster and the Syntactic Bracket (Herbelin & Spiwack).

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:

$$\underline{\mathrm{subset}}_i\ (A : \mathtt{Type}_i)\ (P : A \to \mathtt{Prop}) : \mathtt{Type}_i \triangleq (\Sigma_{i,\mathtt{Prop}}\ A\ P : \mathtt{Type}_{\mathsf{max}(i,\mathtt{Prop})})$$

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 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). 1min if deactivating universes.

Implementation

- Key technique: hash-consing for fast comparison of universe levels, universe instances and algebraic universes. Imperfect as deserialization breaks hashconsing (WIP with T. Braibant, PMP, ...).
- Minimization is fast.
- Currently using naive structures, i.e. the kernel side graph based on union-find does not do compression and the user-side (inference) substitution of universes does not use union-find. Optimize last!
- Universes have to be normalized at the end of inference now: `nf_evars_universes : evar_map -> constr -> constr`

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.
- Pierce and Turner (JFP): Local type inference (based on Cardelli's greedy inference algorithm).

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 (no user syntax yet).
- Resizing rules.

The End

That's all folks!

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 $\sigma : \overrightarrow{i} \rightarrow \overrightarrow{u}$ 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 $\sigma : \overrightarrow{i} \rightarrow \overrightarrow{u}$ 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.