Coq 8.5: what's new, what's next?

M. Dénès [†] & M. Sozeau [*]

[†]University of Pennsylvania
[*]πr^2 - Inria & PPS

Coq POPL Meeting
January 21st 2014
San Diego, USA

Coq 8.5

1. Up and coming
   - Native compilation – M. Dénès
   - Incremental development – E. Tassi
   - Universe polymorphism – M. Sozeau
   - Fast record projections – M. Sozeau
   - New proof engine – A. Spiwack
   - New coqdoc – Y. Régis-Gianas
   - Misc – Coq dev team & contributors
   - `opam` – T. Braibant
   - User libraries, plugins
   - The inconsistency
   - Tentative planning
2. The close future
   - Performance
   - CoqMT – P.Y. Strub
3. The distant future
   - HoTT or OTT?
   - Coq as a programming language

Native compilation – M. Dénès

- NbE style (?)
- Coq $\rightarrow^{\texttt{nativecomp}}$ ML $\rightarrow^{\texttt{ocamlopt}}$ ASM
- Faster than `vm_compute`

+ *Optional*

++/– *Faster at runtime, compilation is slow*

Incremental development – E. Tassi

- `git` model
- Granularity: opaque lemmas
- `.vi` interfaces: cross-file compilation
- CoqIDE only (emacs interface hard to implement)

-/++ *Not optional, backwards-compatible*

+++ *Faster interaction and compilation*

Universe polymorphism – M. Sozeau

- *Checked* universes.
- Global definitions (and inductives) or local with prenex, “bounded” quantification. *Conservativity proof*
- *Elaboration* from typical ambiguity + implicit generalization.
- Checks `HoTT/Coq`, B. Barras and T. Coquand’s semi-simplicial model and another (1-)groupoid model.

Universe polymorphism – M. Sozeau

- *Checked* universes.
- Global definitions (and inductives) or local with prenex, "bounded" quantification. *Conservativity proof*
- *Elaboration* from typical ambiguity + implicit generalization.
- Checks `HoTT/Coq`, B. Barras and T. Coquand's semi-simplicial model and another (1-)groupoid model.
- Better error messaging, unification sensitive to universes.
- Generalized rewriting in `Type`

=/++ *Optional except for checked universes, impacts the ML hacker only*. Backwards-compatibility layer.

=+/+ *Comparable or better performance, more expressive*

Fast record projections – M. Sozeau

- Economic representation
- η-rule (for non-empty records only):
  `Build_rec (p1 r) .. (pn r)` is canonical
- Fast reduction (address of `r` + offset)
- No more stupid unfoldings

+/+− *Optional, backwards-compatibility layer, small source-level incompatibilities.*

$+^{\omega}$ *Exponentially better performance*

TODO: *MS: Graph*

New proof engine – A. Spiwack

- Existential variables + `LogicT`
- Proof-search semantics (`;`, `+`, …)
- Cleaner code: abstract monadic interface
- Native `refine`, dependent goals, goal manipulation

−/++ *Not optional, backwards-compatibility layer*

- *0-15% time overhead (still invesigating,* OCaml *issue likely)*

New coqdoc – Y. Régis-Gianas

- Based on the AST and *not* lexing anymore
- All semantic information available
- Plugable interface

+ *Not optional but legacy `coqdoc` code included*

= *Same performance*

Misc – Coq dev team & contributors

- Interfaces, documentation, and OCaml best practices (P.M. Pédrot, ...)
- Tactics in terms: `$(tac)$` (P.M. Pédrot)
- Module system simplifications (P. Letouzey)
- Generalized intro patterns (H. Herbelin, P. Letouzey)
- Better(?) guard condition (P. Boutillier, H. Herbelin)
- Rewriting with strategies (M. Sozeau)

opam – T. Braibant

- Coq, HoTT/Coq, Coq+Ssreflect, any version, git included
- Some user contribs: Containers, Coccinelle, Ergo, ...
- Submit a pull request! Try! Test!

http://opam.ocaml.org/

https://github.com/braibant/opam-coq-repo

TODO: *TB: USB key?*

User libraries, plugins

- SSREFLECT 1.5 (compatibility with TCs, split between SSR and MATH-COMP, more theories: http://www.msr-inria.fr/projects/mathematical-components/)
- MTAC, CYBELE: “internalized” tactic language
- PACO: coinduction, logically
- EXTLIB, TLC: extensions of the standard library
- WHY3: call SMT solvers from COQ

The inconsistency

- Due to a (very) extensional axiom, related to *equality* on types.
- A good fix is known, but breaks the stdlib's use of the smart guard condition (ha ha). All definitions can be adapted. Work by B. Barras, C. Paulin, M. Dénès, P. Boutilier, …

= Happens in Agda too!

- Argues for less syntax and more logic.

Tentative planning

- February 2014: 8.5-α release
- April 2014: 8.5-β release and opening of new branch (only bug fixes in trunk between α and β)
- August 2014: final release

The close future

1. Up and coming
   - Native compilation – M. Dénès
   - Incremental development – E. Tassi
   - Universe polymorphism – M. Sozeau
   - Fast record projections – M. Sozeau
   - New proof engine – A. Spiwack
   - New coqdoc – Y. Régis-Gianas
   - Misc – Coq dev team & contributors
   - `opam` – T. Braibant
   - User libraries, plugins
   - The inconsistency
   - Tentative planning
2. The close future
   - Performance
   - CoqMT – P.Y. Strub
3. The distant future
   - HoTT or OTT?
   - Coq as a programming language

Performance

- Caching of typechecked terms (E. Tassi, M. Sozeau)
- Generalized hashconsing (T. Braibant, M. Sozeau, P. M. Pédrot, ...)

CoqMT – P.Y. Strub

- Extend the definitional equality with a (first-order) decidable theory, e.g. Presburger arithmetic, constructors

$\rightarrow$ A *strict* equality avoiding coercions, a slightly larger TCB.

- Limited to weak eliminations (SN proof available)
- Waiting for feedback (?)
- To be integrated as an optional feature

The distant future

1. Up and coming
   - Native compilation – M. Dénès
   - Incremental development – E. Tassi
   - Universe polymorphism – M. Sozeau
   - Fast record projections – M. Sozeau
   - New proof engine – A. Spiwack
   - New coqdoc – Y. Régis-Gianas
   - Misc – Coq dev team & contributors
   - `opam` – T. Braibant
   - User libraries, plugins
   - The inconsistency
   - Tentative planning
2. The close future
   - Performance
   - CoqMT – P.Y. Strub
3. The distant future
   - HoTT or OTT?
   - Coq as a programming language

HoTT or OTT?

HoTT ideas, math oriented:

- HITs (prototype by B. Barras)
- Resizing of universes
- *Definitionally*: $\beta\delta$, η?, ι?
- *Propositionally*: proof-irrelevance, functional extensionality, isomorphic types are equal (from an axiom).

OTT/Epigram ideas, CS oriented:

- Levitation of inductive types
- Clear `Prop`/`Set` separation
- *Definitionally*: $\beta\delta\eta\iota$, proof-irrelevance, functional extensionality.
- *Propositionally*: defined, heterogeneous on values, structural on types ($\neq$ iso). With coercion and coherence operators.

Ideally, we should mix the two. . .

Coq as a programming language

Usability, effiency woes:

- I/O, FFI (**TODO:** *MD: Demo Hello world! in Coq*)
- A sliding scale for efficiency/safety in the kernel (ideas: tainting code/data, signatures). Allowing, unchecked recursive definitions, custom reductions. . .
- Definitional proof-irrelevance based on bracket types (useful for DTP and recursion encodings)
- Type-based termination (sized-types, J. L. Sachini)