Review on Coq development

Hugo Herbelin, Pierre Letouzey, Matthieu Sozeau
(for the Coq development team)

Coq Workshop 2013

22 July 2013

Coq releases

Coq 8.4 out since August 2012; currently patch level release is 8.4pl2

Coq 8.5 beta estimated for next winter, in agreement with the usual 2 years release time lapse

Towards Coq V8.5: main ongoing changes

- Second phase of the new proof engine (A. Spiwack):
  - providing multi-success (so that `apply H; eauto` can backtrack on solutions of the first branch which are incompatible with the second branch)
  - deep backtracking (e.g. `(apply H1 + apply H2); apply H3` which behaves like `(apply H1; apply H3) || (apply H2; apply H3)`)
  - multi-goal application (`all:`)
  - a `refine` supporting setting dependent existential variables as goals
  - new tacticals (e.g. `once`)
- Full universe polymorphism (M. Sozeau)
- Evaluation of Coq programs via native compilation to OCaml (M. Dénès)
- New support for asynchronous evaluation of commands, as part of the Paral-ITP ANR project (E. Tassi)

Towards Coq V8.5: main ongoing changes (continued)

- CoqIDE improvements
  - new layout, using gtksourceview (P. Boutillier)
  - searching, more informative coloring, semantical auto-completion (P. Boutillier, P.-M. Pédrot, A. Spiwack)
  - robustness and better responsiveness thanks to replacement of OCaml threads by events, callback and cps-style (P. Letouzey, P.-M. Pédrot)
  - an XML-based communication (P. Boutillier, P.-M. Pédrot)

Towards Coq V8.5: main ongoing changes (continued)

- Efficiency, robustness
  - native representation of record projections, leading to significant improvement in efficiency (M. Sozeau)
  - OCaml code made more modular, private OCaml libraries made more uniform (P. Letouzey, P.-M. Pédrot)
  - more robust management of exceptions (P. Letouzey, P.-M. Pédrot)
  - modules: avoid some duplication in leading to smaller vo files (-30% on the stdlib) (P. Letouzey)
  - lazy load of opaques terms: revised implementation, cleaner code, smaller vo files (-20% on the stdlib)
  - better hash-consing (P. Letouzey, P.-M. Pédrot)

Towards Coq V8.5: other changes

- Guard condition: make it propagate uniformly through β-redexes blocked by a `match` (P. Boutillier)
- Type inference: various improvements of the unification algorithm: better error reporting, better unification in the presence of `match`, management of universes (P. Boutillier, H. Herbelin, M. Sozeau)
- Reduction strategies: new strategy `cbn` for evaluation with fixpoint refolding (P. Boutillier)
- Tactics: `destruct/induction` extended into "small inversion" (P. Boutillier and T. Braibant)
- Tactics: new introduction pattern `[= ...]` for `injection/discriminate` on the fly (Gonthier-inspired)
- Tactics: Rewriting with strategies, `rewrite_strat`, subsuming `autorewrite` (M. Sozeau)
- and various other miscellaneous improvements in tactics

Towards Coq V8.5: work in progress liable to be eventually integrated

- Native persistent arrays, native `int31` (e.g. for efficient verification of SAT traces) (B. Grégoire, M. Dénès)
- Experimental implementation of native higher inductive types (B. Barras) under evaluation

Coq for Homotopy Type Theory

- Foundational results: Homotopy Theory suggests a new interpretation of type theory, beside the proof-as-program, type-as-formula correspondence:

$$\begin{array}{lcl} \text{Type} & = & \text{Space} \\ \text{Equality proof} & = & \text{Path} \end{array}$$

  In particular, Homotopy Theory justifies the non-provability of the Uniqueness of Identity Proofs ($\forall xy : A\, \forall pq : (x =_A y)\, [p =_{x=_A y} q]$): equality is relevant since there might be more than one path up to deformation between two points!

- The key new concepts brought by Homotopy Theory: *univalence* and *higher inductive types*, leading to Homotopy Type Theory (HoTT)

  - Univalence: Equality of types reduces to bijective correspondence ("univ. extensionality")

  - Higher Inductive Types

- Coq used as a foundation for developing results of HoTT: see *Homotopy Type Theory* `http://homotopytypetheory.org/book` and `https://github.com/HoTT`

Coq for Homotopy Type Theory (continued)

- Homotopy Type Theory provides with new insights and directions for research regarding the status of equality in Martin-Löf's type theory (and hence Coq):
  - rethinking equality over A as defined by induction on the type structure of A
  - rethinking rewriting of t by u in $P(t)$ as an operation defined by induction on P

  $\hookrightarrow$ provides computational content to functional extensionality and univalence
- More generally, Homotopy Type Theory suggests to provide
  - explicit access to a strict (i.e. proof-irrelevant) extensional generalization of definitional equality $\equiv_{ext}$ ("extensional" as in Martin-Löf's Extensional Type Theory)
  - cohabiting together with a fully extensional relevant equality $=_{ext}$ ("extensional" as in "function/universe extensionality")
  - to reconsider the conversion rule as a purely technological issue, supporting *any* arbitrary subset of $\equiv_{ext}$ that is provably decidable, hence mechanizable; a prototypical example of this approach is P.-Y. Strub's CoqMT

Other long-term perspectives

- Type-based guard (B. Barras, J. Sacchini)
- Support for K (in Set), inductive-recursive types, a revision of the hierarchy of sorts (M. Sozeau)
- An evolution of Ltac? Towards a typed or dependently-typed, compiled tactic language, see `Cybele` (G. Claret, L. D. C. Gonzalez Huesca, Y. Régis-Gianas), `Mtac` (B. Ziliani *et al*), `VeriML` (Z. Shao, A. Stampoulis), ...

Community

- Coq-Club: more than 1000 mail addresses subscribed
- Coq-bugs
- The `Cocorico` wiki
- The `coq` irc channel
- Summer school (OPLSS, INRIA, Asian's), classes
- Coq'Art, Software Foundations, Certified Programming with Dependent Types, ...
- Coq workshop
- *New*: The ACM SIGPLAN Programming Language Software 2013 Award
- *New*: Towards a Coq consortium