Coq 8.5: what’s new, what’s next?

M. Sozeau – Inria

6th Coq Workshop
July 18th 2014
Vienna, Austria
1. Up and coming

2. The close future

3. The distant future
Incremental development

Enrico Tassi
Asynchronous and parallel processing of definitions.
Huge gain in user productivity.

−/+ ++ \textit{Not optional, backwards-compatible}

+++ \textit{Faster interaction and parallel compilation}
New proof engine

Arnaud Spiwack
Clear proof-search semantics and dependent subgoals.

-/-++ Not optional, backwards-compatibility layer
  - 0-15% time overhead, unnoticeable as the rest of the system got faster
Universe polymorphism

Matthieu Sozeau
Universe polymorphism – M. Sozeau

Truly polymorphic definitions and inductives, cleaner kernel.

=+/++ Kernel change - impacts the ML hacker only.
Backwards-compatibility layer.

=+// Comparable or better performance, more expressive
Native compilation

Maxime Dénès & Benjamin Grégoire
Down to assembly through OCaml.
Useful for large reflection proofs.

+ Optional

++/- Faster at runtime, compilation is slow
Fast record projections

Matthieu Sozeau
Faster conversion and type-checking, smaller memory footprint.

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

$+^\omega$ Exponentially better performance
New coqdoc

Yann Régis-Gianas
A new architecture
A generic documentation tool based on the existing Coq parser.

- Not optional but legacy coqdoc code included
- Same performance
Interfaces, documentation, and OCAML best practices (P.M. Pédrot, ...). Performance is better than 8.4, close to 8.3

Tactics in terms: $(tac)$ (P.M. Pédrot)

Module system simplifications (P. Letouzey)

Tactic improvements (i.e. intro patterns in injection) (H. Herbelin, P. Letouzey, ...)

More expressive guard condition (P. Boutillier, H. Herbelin)

Rewriting with strategies (M. Sozeau)
opam

Thomas 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
Tentative planning

- By the end of the summer: 8.5-\(\beta\) release. Still one big issue with the new tactic engine remaining.
- By the end of the year: final release
The close future

1. Up and coming

2. The close future

3. The distant future
A sliding scale for efficiency/safety in the kernel (ideas: tainting code/data, signatures). Allowing, unchecked recursive definitions, custom reductions…

Machine integers and persistent arrays (M. DÉNÈS, B. GRÉGOIRE)

Caching of typechecked terms (E. TASSI, M. SOZEAU): typecheck once, reuse as many times at 0 cost!

Maximal sharing / generalized hash-consing of terms (T. BRAIBANT, M. SOZEAU, P. M. PÉDROT, …)
CoqMT – P.Y. Strub et al

- Extend the definitional equality with a (first-order) decidable theory, i.e. Presburger arithmetic, constructors
- Some limitations on the use of equalities in context
- Current state: useful but not as well-integrated as it could be, still at 8.3.
- To be integrated as an optional feature

+ Optional

+/− Smoother programming with dependent types, slows down unification/conversion
The distant future

1 Up and coming

2 The close future

3 The distant future
HoTT or OTT?

- **HoTT ideas (Voevodsky et al), math oriented (an ideal foundation):**
  - HITs (prototype by B. Barras, J. Gross will explain).
  - *Propositionally*: proof-irrelevance, functional extensionality, isomorphic types are equal.

- **OTT/Epigram ideas (McBride et al), CS oriented (another ideal foundation):**
  - Generic programming on inductive types universe.
  - *Definitionally*: $\beta\delta\eta$ (as usual), proof-irrelevance.
  - *Propositionally*:
    Functional extensionality, iso. types are *not* equal

Ideally, we should mix the two.
Usability, efficiency woes:

- I/O, FFI (with tainting)
- Proof-irrelevance in conversion: gives “true” dependent pattern-matching, true subset/refinement types.
- Type-based termination (sized-types, J. L. Sacchini).
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/](http://www.msr-inria.fr/projects/mathematical-components/))
- **Mtac, Cybele**: “internalized” tactic language (demo by B. Ziliani)
- **Paco**: co-induction, logically
- **Extlib, TLC**: extensions of the standard library (demos by G. Malecha, A. Charguéraud)
- **Why3**: call SMT solvers from Coq (demo by J.C. Filliâtre)
- **CoqMT** demo by P.Y. Strub
- **Equations**: dependent pattern-matching compilation (demo by M. Sozeau)
The End
Due to a (very) extensional axiom, related to equality on types interacting with the untyped guard condition.

A fix is known, but breaks a few legitimate uses of the recently extended guard condition in the stdlib. All definitions can be adapted. Work by B. Barras, C. Paulin, M. Dénès, P. Boutillier, ... 

Happens in Agda too!

Argues for less syntax and more logic and/or certified metatheoretical proofs.