
# Introduction
# –
# Coq internals

M. Sozeau
INRIA

Coq Implementors Workshop
June 12th 2017

# This week

Goals:

- Have fun making Coq a better project
- Learning about the upcoming features and discussing the roadmap
- Learning about Coq's internals, benefitting from the core developer's presence.

# Outline

- Practical infos
- A bird's eye view of Coq
- Demo: a walkthrough the template-coq plugin
- Roundtable

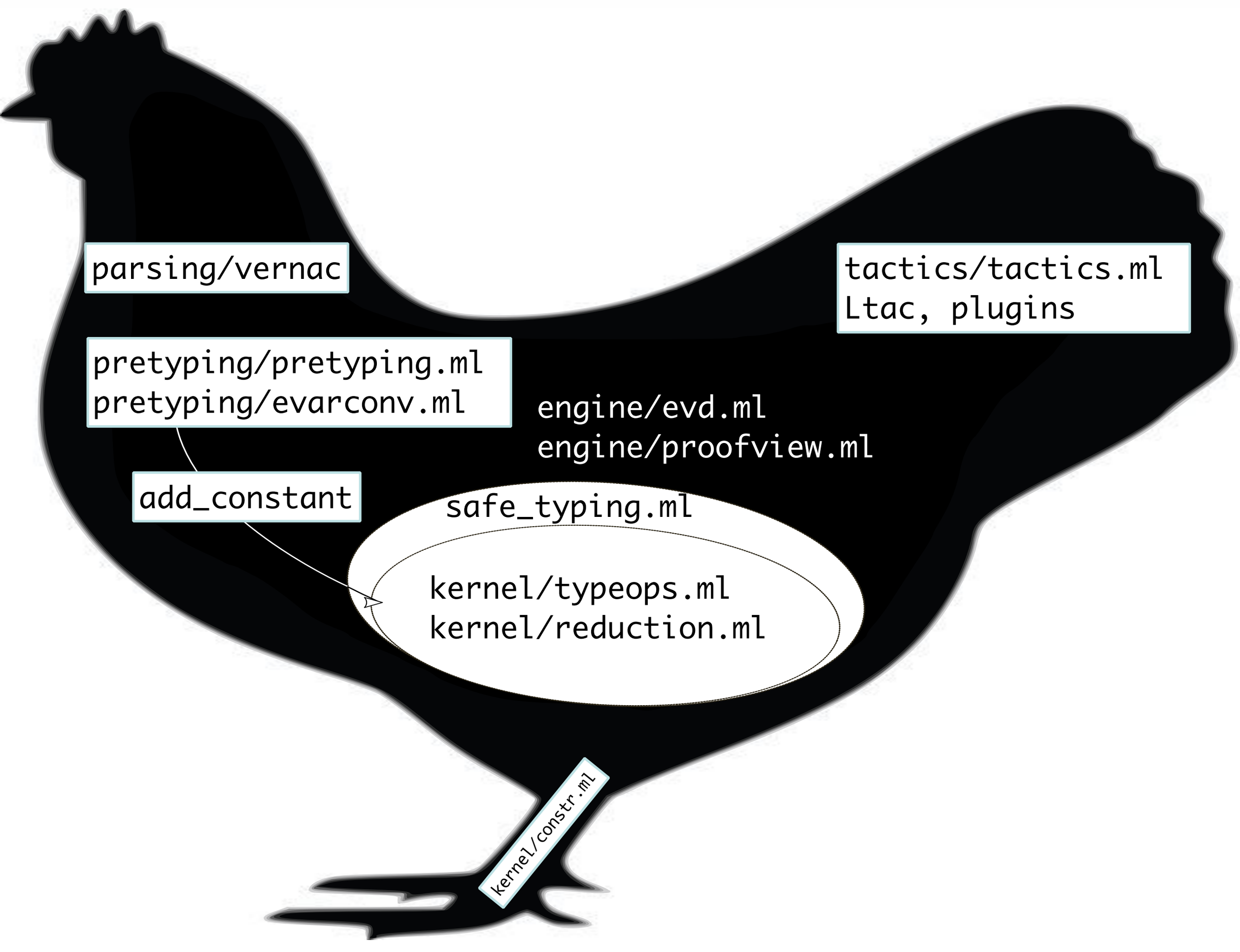
# Schedule

| | 6/12 **MONDAY** | 6/13 **TUESDAY** | 6/14 **WEDNESDAY** | 6/15 **THURSDAY** | 6/16 **FRIDAY** |
|---|---|---|---|---|---|
| 9:00 AM | Check-in | Coding | Coding | Coding | Coding |
| 9:45 AM | | Cumulative Inductive Types<br>A. Timany | Parametricity in Coq<br>A. Anand | Ltac 2<br>P.M. Pédrot | Beautifier and Notations<br>H. Herbelin |
| 10:30 AM | | Break | Break | Break | Break |
| 11:00 AM | | Coq and User Interfaces<br>E.J.G. Arias | Definitional Proof Irrelevance<br>Gaëtan Gilbert | ELPI<br>E. Tassi | Recap / Debriefing |
| 11:45 AM | | PR / Roadmap discussions | PR / Roadmap discussions | PR / Roadmap discussions | |
| 12:30 PM | Lunch | Lunch | Lunch | Lunch | Lunch |
| 2:00 PM | Introduction | Coding / Working group session | Coding / Working group session | Coding / Working group session | Check-out |
| 3:00 PM | Round-table | | | | |
| 3:30 PM | Break | Break | Break | Break | |
| 4:00 PM | Coding/Working group session | Coding / Working group session | Coding / Working group session | Coding / Working group session | |
| 5:00 PM | | | | | |
| 6:00 PM | | | | | |
| 7:00 PM | | | | | |
| 7:30 PM | Diner | Diner | Diner | Diner | |

# Contributions

- Plugins: put your code on github
- Patches: pull-requests on github
- Log your activity on cocorico

# A bird's eye view of Coq's internals


parsing/vernac
tactics/tactics.ml
Ltac, plugins
pretyping/pretyping.ml
pretyping/evarconv.ml
engine/evd.ml
engine/proofview.ml
add_constant
safe_typing.ml
kernel/typeops.ml
kernel/reduction.ml
kernel/constr.ml

```
Definition foo :=
   fun x => x = 3.
Print foo.
```



```
foo : fun x : nat => x = 3
```

# Data types and transformations

string

```
Definition foo :=
  fun x => x = 3.
Print foo.
```

parsing

constr_expr (AST)

```
VernacDefinition( “foo”,
  DefinedBody(
    CLambdaN([ “x”, CHole],
      CNotation( “_ = _”,
        [ CRef “x”; CPrim 3 ])))
VernacPrint (PrintName “foo”)
```

internalization
(notations, globals, implicit args)

glob_constr (untyped)

```
GLambda( “x”, GHole,
  GApp( GRef “Coq.Init.Logic.eq”,
    [ GHole;
      GVar “x”;
      GApp( GRef “Coq.Init.Datatypes.S”,
        […GRef “Coq.Init.Datatypes.O”...])]))
```

*pretyping*
*(De Bruijn idxs,coercions,…)*

constr (typed)

```
Lambda( “x”, Ind “Coq.Init.Datatypes.nat”,
  App( Ind “Coq.Init.Logic.eq”,
    [ Ind “Coq.Init.Datatypes.nat”;
      Rel 1;
      App(Construct “Coq.Init.Datatypes.S”,
        […Construct “Coq.Init.Datatypes.O”...])]))
```

# Data types involved

string

```
fun x : nat => x = 3.
```

printing

constr_expr

```
CLambdaN([ “x”, CRef “nat” ],
  CNotation( “_ = _”,
    [ CRef “x”; CPrim 3 ])))
```

externalization

glob_constr

```
GLambda( “x”, GRef “Coq.Init.Datatype.nat”,
  GApp( GRef “Coq.Init.Logic.eq”,
      [ GRef “Coq.Init.Datatypes.nat”;
        GVar “x”;
        GApp( GRef “Coq.Init.Datatypes.S”,
          […GRef “Coq.Init.Datatypes.O”...])]))
```

detyping

constr

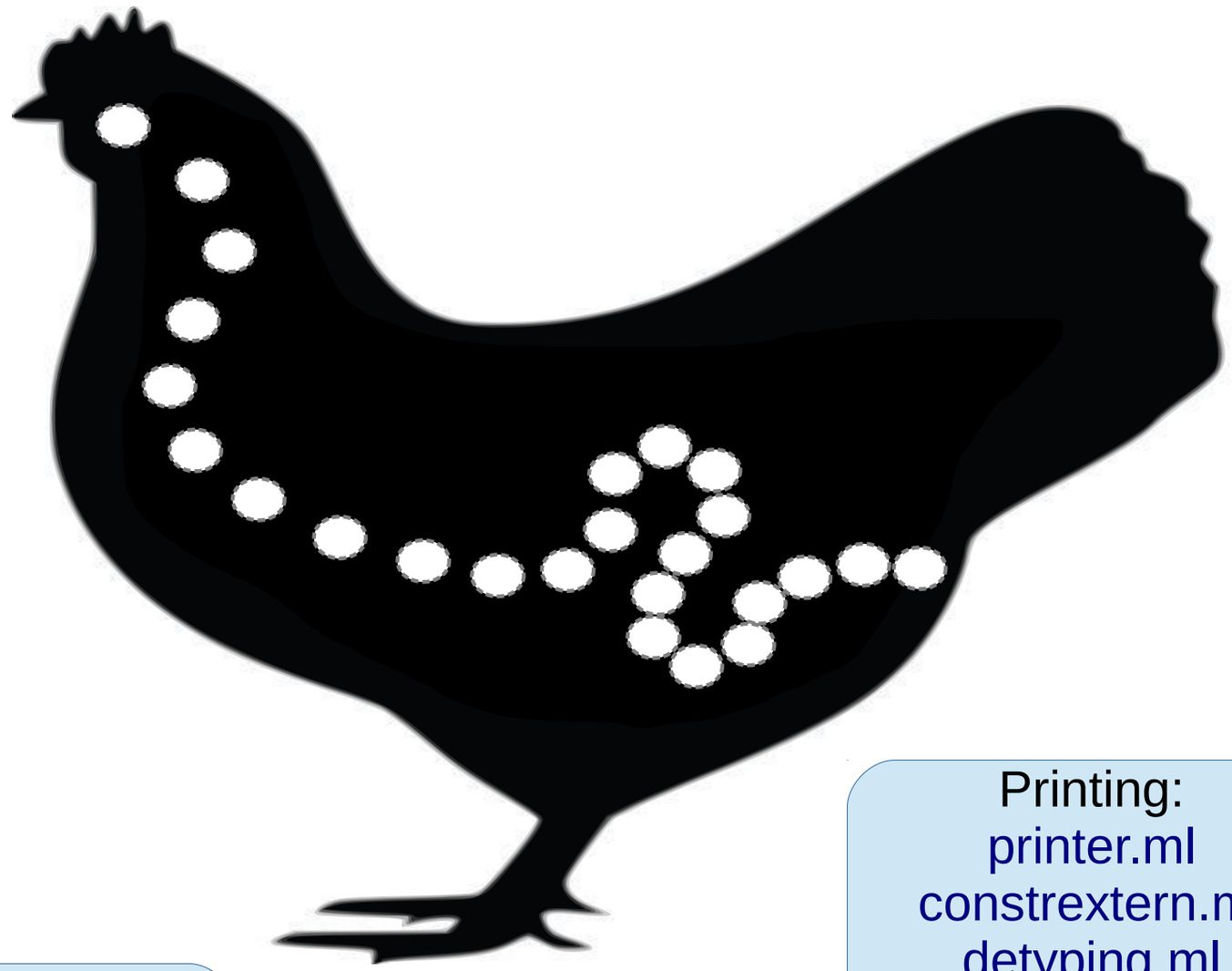
```
Lambda( “x”, Ind “Coq.Init.Datatypes.nat”,
  App( Ind “Coq.Init.Logic.eq”,
      [ Ind “Coq.Init.Datatypes.nat”;
        Rel 1;
        App( Construct “Coq.Init.Datatypes.S”,
          […Construct “Coq.Init.Datatypes.O”...])]))
```

# Where's the code?

Frontend:
vernac.ml

Parsing:
g_vernac.ml4 g_constr.ml4
vernac_expr constr_expr

Interpreter:
vernacentries.ml
( dumbglob.ml )



Term internalization:
constrintern.ml
notation.ml
glob_constr

Type inference:
pretyping.ml
constr

Printing:
printer.ml
constrextern.ml
detyping.ml

# Terms

- constr.ml: the term language (termops.ml for operations on it)
- econstr.ml: terms with evars, seen during typechecking and tactic execution
- glob_constr.ml: abstract syntax with names resolved, implicits inserted
- constr_expr.ml: AST built from parsing

# Directories

- bin/: built tools
- checker/: the coqchk tool
- dev/: info and utilities for developers (dev/doc/* is really useful)
- doc/ : the reference manual
- engine/ : the engine for type checking and tactics (terms with metavariables and interactive proof engine)
- grammar/ extensible camlp5 parsing grammar (for TACTIC EXTEND, VERNAC COMMAND EXTEND)
- ide/ coqide tool
- interp/ internalisation, manipulation of constr_expr AST and extensible parsing
- intf/ type definitions for AST expressions (constr_expr, vernac_expr) and globalized ones (glob_term.ml)
- kernel/ the kernel type checker, including conversion procedures, and safe environment handling interface, module checking
- lib/ general data structures library, + global flag handling
- library/ handling of "libraries", environment with extra state (e.g. implicit args), name resolution
- ltac/ (plugins/ltac): the ltac language, including extra tactics
- parsing/ camlp5 parser to the AST expressions
- plugins/ cc (congruence closure), extraction, firstorder, fourier, funind, micromega, nsatz, omega, quote, romega, rtauto, setoid_ring, ssrmatching, syntax (numeral notations)
- pretyping/ type inference, pattern-matching compilation, unification, coercions, inductive scheme construction
- printing/ pretty-printers of ASTs
- proofs/ proof handling
- stm/ transactional machine to handle the document as a whole, scheduling proofs to workers, based on a revision system model
- tactics/ basic tactics (apply, rewrite, destruct, induction, injection, discriminate, auto/eauto, autorewrite…)
- test-suite/ regression test-suite, includes scripts from bugs and tests of features (not really unit-tests though)
- theories/ the Coq standard library
- tools/ coqdep/coq_makefile/…
- toplevel/ the coqtop top-level
- vernac/ vernacular commands implementation (Definition, Inductive, Section, Arguments, …)

# Developer Ressources

- PRs on github
- coqdev
- Bugzilla coq.inria.fr/bugs
- cocorico, see e.g. last year's presentations at CoqIW2016
- dev/doc/*
- gitter.im channel

# Simple demo

- A plugin that implements a simple [intro] tactic and a vernacular command [Myprint]

- http://github.com/mattam82/example-plugin

# Demo: Template-Coq

- A walkthrough of a realistic plugin with options, matching on terms and building
- Template-Coq reifies/quotes Coq declarations (Defs, Inductives) into a Coq datatype for the term syntax and environment.
- It also includes a meta-command to interpret this Coq datatype into terms (unquoting).

# Roundtable

- Everybody with a project in mind talks about it for 5’
- So that we know what each one is going to do
- So that we know how to organize the work together, what expertise you might need.