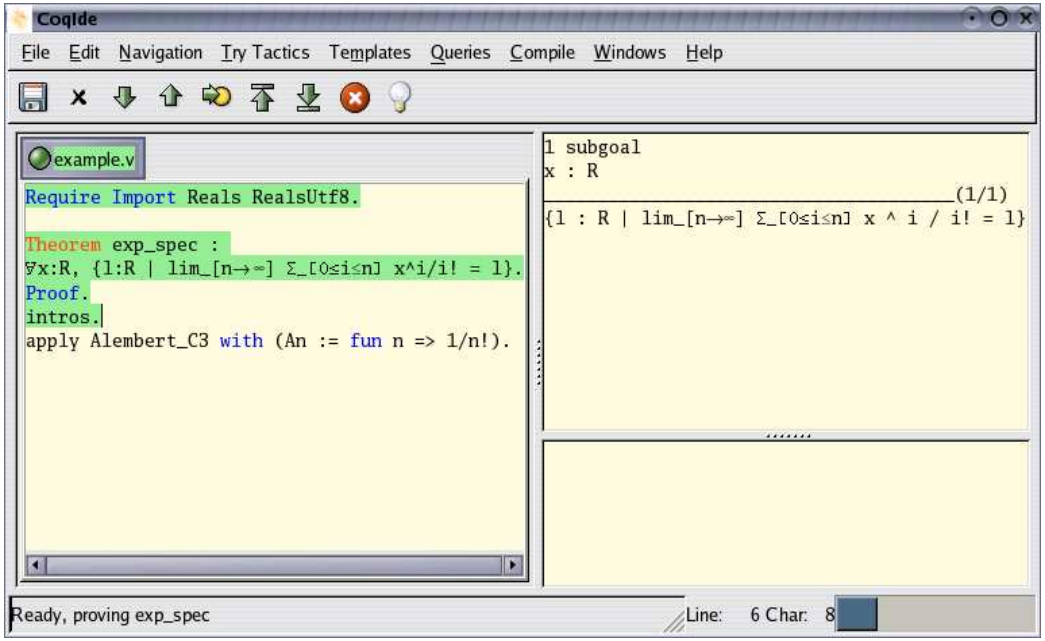
Coq,
a formal proof development environment combining logic and programming



Hugo Herbelin

Coq in a nutshell
(http://coq.inria.fr)

A logical formalism that embeds an executable typed programming language: the Calculus of Inductive Constructions *[Coquand-Paulin 1988]*

A small trustable kernel *that checks that proofs are correct*

A high-level concrete language of functions and specifications

```
Inductive bool := true | false.
Definition neg c := match c with true => false | false => true end.
Theorem neg_involutive : forall c:bool, neg (neg c) = c.
```

A language of semi-interactive tactics to solve specifications and theorems

```
Theorem neg_involutive : forall c:bool, neg (neg c) = c.
Proof. induction c; auto. Qed.
```

Libraries on arithmetics, analysis, logic, data types, algebra

An extraction mechanism to ML, Haskell and Scheme

40 direct contributors to the archive over 24 years + a community of users

Top applications

Certification of a C compiler (Xavier Leroy and the CompCert team)

Full formalization of the 4-color theorem proof (Gonthier - Werner)

Extensive repository of constructive mathematics (Nijmegen)

Certification of a commercial JavaCard applet (Gemalto)

Exact real arithmetic

Hoare logic toolbox (Y-not, Boston & Cambridge)

SAT certification

...

for an excerpt, see `http://coq.inria.fr`, link contributions

The Calculus of Inductions Constructions (CIC)

Terms and types

t, u, T, U	$::=$	$\texttt{fun}\ x : T\ \Rightarrow\ t \mid x \mid t\,t \mid \texttt{let}\ x{:=}t\ \texttt{in}\ u \mid c$	*λ-calculus with definitions*
	$\mid$	$\texttt{forall}\ x : T.\ T$	*dependent types*
	$\mid$	$\texttt{Prop} \mid \texttt{Type}_\texttt{n}$	*sortes*
	$\mid$	I	*inductive/coinductive types*
	$\mid$	$\texttt{match}\ t\ \texttt{with}\ \overrightarrow{c\ \overrightarrow{x} \Rightarrow t}\ \texttt{end} \mid C$	*pattern-matching on constructors*
	$\mid$	$\texttt{fix}_i\ \overrightarrow{f := t : T} \mid \texttt{cofix}_i \overrightarrow{f := t : T}$	*structural recursion*

Contexts of declarations

$$
\begin{array}{rcl}
\Gamma & ::= & \epsilon \\
& \mid & \Gamma,\ \texttt{Definition}\ c := t : T \\
& \mid & \Gamma,\ \texttt{Parameter}\ x : T \\
& \mid & \Gamma,\ \texttt{Inductive}\ I_1\ \overrightarrow{(x : T)} : T_1 := \overrightarrow{C_1 : U_1} \ldots\ \texttt{with}\ I_n\ \overrightarrow{(x : T)} : T_n := \overrightarrow{C_n : U_n}
\end{array}
$$

Judgments

$\Gamma \vdash T : \texttt{Prop}$	means	T is a formula
$\Gamma \vdash T : \texttt{Type}_n$	means	T is a type
$\Gamma \vdash t : T$	means	t is a program or a data of type T (if T is a type)
$\Gamma \vdash t : T$	means	t is a proof of T (if T is a formula)

$\hookrightarrow$ relies on the Curry-Howard "identity" between proofs and programs

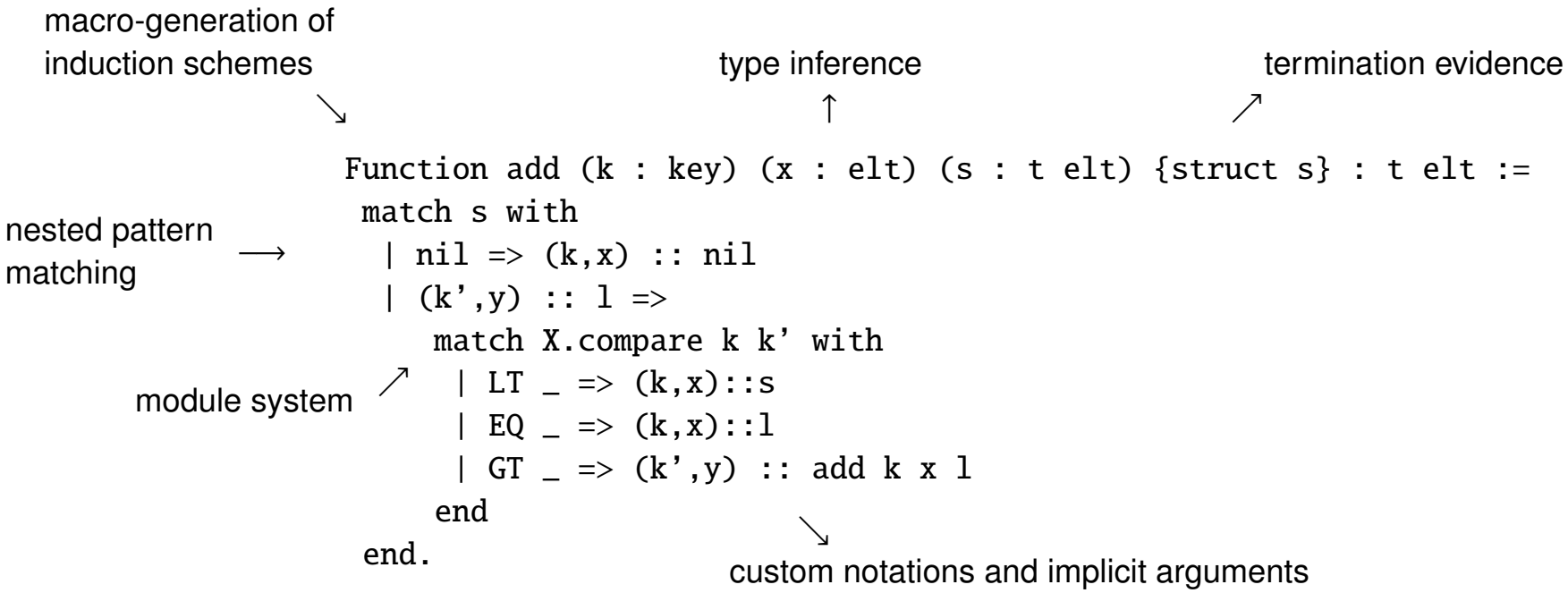
The verification kernel

8 000 lines of Objective Caml code out of 190 000 lines for the full Coq

CIC terms provide with a notion of cross-verifiable disk-storable proof-object

Partly certified in Coq itself (Bruno Barras)

The high-level specification and programming languages



The proof languages

A set of tactics ranging from low-level logical rules...

e.g. lemma application or induction

... to high-level automation tactics

e.g. decision of Presburger arithmetic, closed equational reasoning, semi-decision of first-order logic, approximate decision of closed real fields, decision of equality in rings and fields, ...

$\mathcal{L}_{tac}$*: a dedicated functional language for building tactics*

A mathematical-style proof language

Various third-party toolboxes of tactics (written in $\mathcal{L}_{tac}$*, in Coq itself, or using ML plugins, or using pipe communicating with Coq*

The “mathematical” proof language

```
Lemma even_double_n: (forall m, even (double m)).
proof.
  assume m:nat.
  per induction on m.
    suppose it is 0.
      thus thesis.
    suppose it is (S mm) and thesis for mm.
      then H:(even (S (S (mm+mm)))).
      have (S (S (mm + mm)) = S mm + S mm)  using omega.
      hence (even (S mm +S mm)) by H.
  end induction.
end proof.
Qed.
```

Other toolboxes or tactics

Gonthier's `ssreflect` *language for reasoning on decidable structures and more*

U. Penn's `LNgen` *for reasoning on languages with binders*

Connections to SML solvers and Computer Algebra Systems (`Z3`, `simplify`, `maple`, *...)*

Nowak's toolbox for cryptographic protocols at AIST

Termination proof toolboxes

The Why, Caduceus and Krakatoa specification languages for certifying ML, C and Java programs

Y-not toolbox for reasoning in Hoare logic

...

Libraries

Bundled with Coq

- Peano arithmetics in $\mathbb{N}$
- Efficient machine-word-based arithmetics in $\mathbb{N}$, $\mathbb{Z}$, $\mathbb{Q}$
- Classical real analysis
- Programming libraries: lists, finite sets, finite maps
- Logical axioms

Third-party libraries

- Constructive real analysis
- Algebra
- Group theory
- Programming libraries: finger trees, union-find, ...

An example

Advertising the first Asian Coq Summer School

Dates: August 24-31, 2009

Location: Tsinghua, Beijing

Organization: joint Tsinghua-INRIA team FORMES (Pr. Jean-Pierre Jouannaud and Ming Gu)

Web site: http://coqschool-asia.org/wiki/Start

Main proof assistants

Dedicated to the formalization of mathematics: Mizar

Dedicated to proof certification: HOL, ACL2

Hybrid systems: Isabelle-HOL and Isabelle-ZF, HOL-light, PVS

Hybrid systems combining a functional language and a logic: Coq, Agda2

Other systems: Twelf, Matita, ...