A new generation of Secure Proof Assistants integrating Small Proof Engines

F. Blanqui[1] J.-P. Jouannaud[2] P.-Y. Strub[2]

(1) INRIA - LORIA

(2) Ecole Polytechnique - LIX

Outline

Proof assistant

- A programming language dedicated to processing mathematics.
- A set of deduction and computation rules characterizing the logic chosen for expressing mathematical statements and their proofs.
- Proof tactics helping the user building proofs.
- A tactic language for writing new tactics.
- Libraries of proved theorems.
- Security of the proof assistant is ensured by a proof-checking algorithm, called the kernel.

Coq successive frameworks I

- The Calculus of Constructions
  CC : [Coquand and Huet, 1985]

Paradigm:
readability of a small kernel ensures security.

- The Calculus of Inductive Constructions
  CIC : [Coquand and Paulin, 1990]
- The Calculus of Guarded Constructions
  CGC : [Gimenez, 1996]
- The Calculus of Modular Constructions
  CMC : [Chrzaczsz, 2003]

Paradigm:
provability of a medium-size kernel ensures security.

Coq successive frameworks II

- The Calculus of Algebraic Constructions
  CAC : [Blanqui, 2001]
- The Calculus of Compiled Constructions
  [Gregoire, 2004]

Paradigm:
securiry requires proving a large kernel.

- The Calculus of Congruent Inductive Constructions
  CCIC : [Blanqui, Jouannaud and Strub, 2007]

Paradigm:
incremental provability of a small number of stable medium-size kernels ensures security.

Outline

Major problems

- Specifications and proofs should be close to the mathematical practice.
- Proofs may need complex *computations*: the four color theorem completed late 2004 by Georges Gonthier and Benjamin Werner.
- Transparent computations are powerful, change our style of making proofs (proofs by reflexion), and are required for complex tasks [Gonthier]
- Computations should not require user's assistance.
- Despite some important progress, these problems remain unsolved.

Major practical objective

- Close the gap between mathematical practice and development of formal proofs
- by automating proofs of decidable subgoals
- without compromising the security of using Coq.

Automating proofs of decidable subgoals

Building as a black box

arbitrary decision procedures

with proof certificates

into the computational part of

the Calculus of Inductive Constructions

Outline

The **Calculus of Constructions** is a higher-order typed calculus integrating polymorphism and dependent types in a Curry-Howard style, and a computation mechanism via its conversion rule:

$$\frac{\Gamma \vdash t : T \quad T \sim_\Gamma T'}{\Gamma \vdash t : T'} \text{ [CONV]}$$

- computation power expressed by $\sim_\Gamma$
- type checking is decidable as long as $\sim_\Gamma$ is decidable

A more powerful conversion $\sim_\Gamma$ implies

- *bigger deduction steps*
- *hence smaller proofs*
- *more terms are typable*, hence
- *more proposition can be expressed*, and
- *dependent types become usable.*
- But big risk to run into undecidability.

Conversion relations (1/4)

- β-convertibility (**CC** [Coquand, Huet 80])
- β-convertibility + recursors for inductive types (**CIC** [Paulin 89], [Altenkirch 93], [Werner 94])

$$nat := 0 : nat \mid S : nat \to nat$$

$$0 + b \xrightarrow{\iota} b \qquad S(a) + b \xrightarrow{\iota} S(a + b)$$

$$2 + S(x) \sim_\Gamma S^3(x) \text{ but } \neg(x + 0 \sim_\Gamma x)$$

($+$ is defined by induction on its first argument)

Conversion relations (2/4)

- β-convertibility + a rewrite system R
  (**CAC** [Barbanera 90], [Fernandez, Geuvers 93], [Barthe 98], [Blanqui, Jouannaud, Okada 01], [Blanqui 05])

  R can be a higher-order rewrite system, contain non-linear patterns, can mimic ι-reduction, rewrite module AC symbols...

  $$0 : nat \quad S : nat \to nat \quad + : nat \to nat \to nat$$
  $$0 + b \to b \qquad b + 0 \to b \qquad S(a) + b \to S(a + b)$$

  We recover $x + 0 \sim_\Gamma 0$ and $0 + x \sim_\Gamma x$

Kernel can get very big, compromising security

Conversion relations (3/4)

- all terms provably equal are convertible
  (**CCext** [Hofmann 95], [Oury 05])

$$\Gamma \vdash t = u \implies t \sim_\Gamma u$$

type checking is not decidable

subject reduction and type unicity are lost

strong normalization of β is lost too

Conversion relations (4/4)

- Open Calculus of Construction
  (**OCC** [Stehr 02])

  Allow the use of any equational theory in the conversion rule.

Same drawbacks as for ECC

Our Calculus of Congruent Inductive Constructions

A calculus of inductive constructions including:

- $\beta\iota$-conversion (as in CIC)
- an arbitrary first order theory $\mathcal{T}$ over equality with decidable entailment, e.g., Presburger arithmetic
- relevant *closed* equational hypotheses from Γ

  $x = y + 2, f(y + 2) = g(x) \in \Gamma \implies f(x) \sim_\Gamma g(y)$
- whose type checking is decidable
- whose conversion relation incorporates entailment in $\mathcal{T}$ as a black box procedure.

  (i.e. the theory $\mathcal{T}$ is not hard-coded in CCIC)

CCIC has more proofs by reflexivity of equality

$$\begin{aligned}\Gamma = [x\ y\ z\ t : nat], [f : nat \to nat],\\ [p_1 : t \doteq 2], [p_2 : f\,(x+3) \doteq x+2],\\ [p_3 : f\,(y+t)+2 \doteq y+z], [p_4 : y = x+1]\end{aligned}$$

Under assumptions in Γ, a proof by *reflexivity* of $3 \doteq 3$ should be a proof of $z \doteq \mathrm{S}\,t$.

Not true of Coq, but true of CCIC.

CCIC: has more typable terms

- reverse : $\forall n : nat, \mathbf{list}\, n \to \mathbf{list}\, n$
- $\_\,\_ : \forall n_1, n_2 : nat, \mathbf{list}\, n_1 \to \mathbf{list}\, n_2 \to \mathbf{list}\,(n_1 + n_2)$
- We want to prove that
  $\forall n_1, n_2 : nat\ \forall l_1 : \mathbf{list}\, n_1\ \forall l_2 : \mathbf{list}\, n_2, \mathrm{reverse}(l_1\, l_2 l_2 l_1) = l_1\, l_2 l_2 l_1$
- But $\mathrm{reverse}(l\, l') = \mathrm{reverse}(l')\ \mathrm{reverse}(l)$ is typable only if $n + n' \sim_\Gamma n' + n$ since

$$\begin{array}{rll} \mathrm{reverse}(l\, l') & \text{has type} & \mathbf{list}(n + n') \\ \mathrm{reverse}(l')\,\mathrm{reverse}(l) & \text{has type} & \mathbf{list}(n' + n) \end{array}$$

Typing fails in Coq but succeeds in CCIC.

Outline

Our Calculus: restricted for exposition to

- building in Presburger arithmetic only
- weak recursor over natural numbers

We use the { usual terms of CC
symbols for arithmetic

$$t, u, T \quad ::= \overbrace{s \in \{\star, \Box\} \mid x \in \mathcal{X} \mid t\,u \mid (\forall x : T)u \mid [\lambda x : T]u}^{\text{CC part}}$$

$$\mid \quad \underbrace{nat \mid 0 \mid \mathsf{S} \mid \dot{+} \mid \mathsf{Rec}^{\mathcal{W}}_{\mathbb{N}}(t, T)\{t_0, t_S\}}_{\text{natural numbers}} \quad \underbrace{\mid \doteq \mid \mathsf{Eq}_T(t)}_{\text{eq. reasoning}}$$

- $\doteq$ is a polymorphic equality symbol
- $\mathsf{Eq}_T(t)$ is a proof of $t \doteq_T t$

Typing rules (1/2)

- We keep the usual rules of CC
- Some axioms for typing symbols

$$\frac{}{\vdash 0 : nat} \qquad \frac{}{\vdash \mathsf{S} : nat \to nat} \qquad \text{etc...}$$

- Recursor rule:

$$\frac{\Gamma \vdash t : nat \quad \Gamma \vdash Q : nat \to \star \quad \Gamma \vdash f_0 : nat \quad \Gamma \vdash f_S : \forall (n :^u nat).\, Q\, n \to Q\, (\mathsf{S}\, n)}{\Gamma \vdash \mathsf{Rec}^{\mathcal{W}}_{\mathbb{N}}(t, Q)\{f_0, f_S\} : Q\, t}\ [\iota\text{-ELIM}]$$

- Eq-rules:

$$\frac{\Gamma \vdash t_1 : T \quad \Gamma \vdash t_2 : T \quad \Gamma \vdash p : \forall (P : T \to \star).\, P\, t_1 \to P\, t_2}{\Gamma \vdash \mathsf{Eq}(p) : t_1 \doteq_T t_2}\ [\text{EQ-INTRO}]$$

Typing rules (2/2)

All computations go to the conversion rule

$$\frac{\Gamma \vdash t : T \quad \Gamma \vdash T' : s' \quad T \sim_\Gamma T'}{\Gamma \vdash t : T'}$$

$\sim_\Gamma$ will include

- β-conversion
- weak recursor for natural numbers

$$\mathit{RecW}(0, Q)\{f_0, f_S\} \xrightarrow{\iota} f_0$$

$$\mathit{RecW}(\mathsf{S}\, t, Q)\{f_0, f_S\} \xrightarrow{\iota} f_S\, t\, (\mathrm{Rec}^{\mathcal{W}}_{\mathbb{N}}(t, Q)\{f_0, f_S\})$$

- Validity entailment for Presburger arithmetic $\mathcal{T}$

Defining $\sim_\Gamma$ (1/3) - Algebraisation

- Needs extracting equations from typing environments
- $\Longrightarrow$ communication between CC and the algebraic world

Green locations are in the algebraic cap
Red locations are alien positions

$\mathsf{APos}(t) \rightarrow$ *alien* positions of t
$\mathsf{CPos}(t) \rightarrow$ *algebraic* positions of t

$\mathcal{R} \rightarrow$ equivalence relation
$\pi_{\mathcal{R}}(t) \rightarrow$ s.t. $\pi_R(t) = \pi_R(u) \Leftrightarrow t\,\mathcal{R}\,u$
($\pi_{\mathcal{R}}(t)$ in a fresh set of variables)

$\mathsf{cap}_{\mathcal{R}}(t) = t[p \leftarrow \pi_{\mathcal{R}}(t_{|p})]_{p \in \mathsf{APos}(t)}$

Defining $\sim_\Gamma$ (2/3) - Equations extraction

$$\frac{\Gamma, [x : t_1 \doteq t_2] \vdash \circ : u}{\Gamma \vdash \circ : (\forall x : t_1 \doteq t_2) u}$$

$\sim_\Gamma$ must anticipate the fact that the equation $t_1 \doteq t_2$ will appear in the environment when typing $(\forall x : t_1 \doteq t_2) u$.

i.e. if $u \sim_{\Gamma[x:T]} v$ then $(\forall x : T) u \sim_\Gamma (\forall x : T) v$

$\sim_\Gamma$ is a family of context-dependent congruences.

Inference system defining $\sim_\Gamma$ (3/3)

$$\frac{t =_{\beta\iota} u}{t \sim_\Gamma u}\ [\beta\iota] \qquad \frac{\Gamma = \Gamma_1, [z : W], \Gamma_2 \quad W \xrightarrow{\beta}_* u \doteq v}{u \sim_\Gamma v}\ [\text{EQ}]$$

$$\frac{\mathcal{T}, \{\text{cap}_{\sim_\Gamma}(u) = \text{cap}_{\sim_\Gamma}(v) \mid u \sim_\Gamma v\} \vDash \text{cap}_{\sim_\Gamma}(s) = \text{cap}_{\sim_\Gamma}(t)}{s \sim_\Gamma t \quad [\mathcal{T}, \Gamma, s, t]}\ [\text{DED}]$$

$$\frac{t_1 \sim_\Gamma u_1 \quad t_2 \sim_{\Gamma,[x:t_1]} u_2}{(\forall x : t_1) t_2 \sim_\Gamma (\forall x : u_1) u_2}\ [\text{PROD}]$$

+ same rule for abstraction + contextual rules.

Precise definition of CCIC is more complex:

- Definition of $\sim_\Gamma$ and calculus must use a set of annotations used for restricting
  $\left\{\begin{array}{l}\text{some kind of applications}\\ \text{and equations extraction}\end{array}\right.$
  which would otherwise lift inconsistencies from the environment to types.

- Details in
  [Blanqui, Jouannaud and Strub : Building decision procedures in the Calculus of Inductive Constructions]
  [Strub : The Calculus of Congruent Inductive Construction, PhD thesis, april 2008]

Sketch of consistency proof

- Calculus has standard properties of Pure type Systems:
  Stability by substitution
  Inversion lemma
  Subject reduction
  ...
- Strong normalization of $\beta\iota$ for well formed terms follows from proof irrevelance

  [Barthe, ICALP 1998]

Outline

Decidability

- use a saturation based algorithm *a la* Nelson-Oppen-Shostak
- equations are purified between
  the algebraic and non-algebraic worlds
- deductions are propagated between both worlds via
  equations between variables
- since some hypotheses can appear locally,
  all literals will be marked with a set of labels
  indicating their availability.

Decidability

We suppose $\Gamma \vdash t : T$ and $\Gamma \vdash u : U$

Does $t \sim_\Gamma u$ holds ?

We will deal with configurations $[E, A, N]$ where

- E is a set of mixed literals
- A is a set of pure algebraic literals
- N is a set of non algebraic equations in solved-form

Each literal is annotated with a set of labels
taken from a fresh set of variables $\mathcal{Y}$.

Decidability

- Configurations are reduced by a normalizing and confluent reduction system
- We start with the configuration $[\mathrm{Eq}(\Gamma) \cup \{t \neq u\}, \emptyset, \emptyset]$ with

$$\mathrm{Eq}(\Gamma) = \{v_1 = v_2 \mid [x : v_1 \doteq v_2] \in \Gamma_{\downarrow\beta}\}$$

  in which literals are annotated with an empty set of labels)
- $t \sim_\Gamma u$ if and only if $[\mathrm{Eq}(\Gamma) \cup \{t \neq u\}]$ reduces to $\bot$

Example (1/6)

Under Assumption: $\Gamma = [p : y \doteq x + 2]$ prove:

$$[\lambda q : f\,(y-2) \doteq g\,y].\, f\,x \sim_\Gamma [\lambda q : f\,(y-2) \doteq g\,y].\, g\,y$$

Purification gives:

N	A
	$X_1 \neq X_2$
	$y = x + 2$
$X_1 \mapsto [\lambda q : E_1]C_1$	$A_1 = f(y-2)$
$X_2 \mapsto [\lambda q : E_2]C_2$	$A_2 = f(y-2)$
$E_1 \mapsto A_1 \doteq B_1$	$B_1 = g(y)$
$E_2 \mapsto A_2 \doteq B_2$	$B_2 = g(y)$
	$C_1 =_{\{E_1\}} f(x)$
	$C_2 =_{\{E_2\}} g(y)$

Example (2/6)

Trivial deductions in A yield

N	A
	$X_1 \neq X_2$
	$y = x + 2$
$X_1 \mapsto [\lambda q : E_1]C_1$	$A_1 = A_2 = f(y-2)$
$X_2 \mapsto [\lambda q : E_2]C_2$	$B_1 = B_2 = g(y)$
$E_1 \mapsto A_1 \doteq B_1$	$C_1 =_{\{E_1\}} f(x)$
$E_2 \mapsto A_2 \doteq B_2$	$C_2 =_{\{E_2\}} g(y)$
	$A_1 = A_2$
	$B_1 = B_2$

We can now substitute A_2 (resp. B_2) by A_1 (resp. B_1) in N, and deduce $E_1 = E_2$ in N.

Example (3/6)

N	A
$X_1 \mapsto [\lambda q : E_1]C_1$ $X_2 \mapsto [\lambda q : E_2]C_2$ $E_1 \mapsto A_1 \doteq B_1$ $E_2 \mapsto A_1 \doteq B_1$ --- $E_1 = E_2$	$X_1 \neq X_2$ $y = x + 2$ $A_1 = f(y - 2)$ $B_1 = g(y)$ $C_1 =_{\{E_1\}} f(x)$ $C_2 =_{\{E_2\}} g(y)$

We can now substitute E_2 by E_1 in N and A, getting rid of E_2.

Example (4/6)

N	A
$X_1 \mapsto [\lambda q : E_1]C_1$ $X_2 \mapsto [\lambda q : E_1]C_2$ $E_1 \mapsto A_1 \doteq B_1$	$X_1 \neq X_2$ $y = x + 2$ $A_1 = f(y - 2)$ $B_1 = g(y)$ $C_1 =_{\{E_1\}} f(x)$ $C_2 =_{\{E_1\}} g(y)$

From N, the equation $A_1 =_{\{E_1\}} B_1$ is added to A:

Example (4/6)

N	A
	$X_1 \neq X_2$
	$y = x + 2$
$X_1 \mapsto [\lambda q : E_1]C_1$	$A_1 = f(y - 2)$
$X_2 \mapsto [\lambda q : E_1]C_2$	$B_1 = g(y)$
$E_1 \mapsto A_1 \doteq B_1$	$C_1 =_{\{E_1\}} f(x)$
	$C_2 =_{\{E_1\}} g(y)$
	$A_1 =_{\{E_1\}} B_1$

Example (5/6)

Now, Presburger arithmetic gives us in A:

$\{y = x + 2, f(y - 2) = g(y)\} \vdash f(x) = g(y)$

hence the new configuration:

N	A
$X_1 \mapsto [\lambda q : E_1]C_1$ $X_2 \mapsto [\lambda q : E_1]C_2$ $E_1 \mapsto A_1 \doteq B_1$	$X_1 \neq X_2$ $y = x + 2$ $A_1 = f(y - 2)$ $B_1 = g(y)$ $C_1 =_{\{E_1\}} f(x)$ $C_2 =_{\{E_1\}} g(y)$ $A_1 =_{\{E_1\}} B_1$ $C_1 =_{\{E_1\}} C_2$

Example (6/6)

We now substitute C_2 by C_1 in N:

N	A
$X_1 \mapsto [\lambda q : E_1]C_1$ $X_2 \mapsto [\lambda q : E_1]C_1$ $E_1 \mapsto A_1 \doteq B_1$ $X_1 = X_2$	$X_1 \neq X_2$ $A_1 = A_2 = f(x+2)$ $B_1 = g(x)$ $C_1 =_{\{E_1\}} f(x)$ $A_1 =_{\{E_1\}} B_1$

Then substituting X_2 by X_1 yields the contradiction $X_1 \neq X_1$ in A.

Outline

Certificates

- [_] holds the certificate returned by $\sim_\Gamma$
- A certificate is issued for each deduction in $\mathcal{T}$
- Certificate verification becomes part of proof checking
- Certificates are expected to be small and verifiable in linear time
- Security of the system relies on both the kernel and the certificate chechers.
- The kernel does not change over time
- Certificates exist for many first-order decision procedures

Conclusion

- CCC extends to the Calculus of Inductive Constructions, yielding the Calculus of Congruent Inductive Constructions (CCIC)
- Shostak's method can be used for combining decision procedures (incrementality)
- Security will be based on a fixed CIC based checkable kernel plus a certificate checker for Shostak's generic mechanism plus certificate checkers for basic theories

Thank you

Details in Pierre-Yves Strub's coming PhD (before summer)
`http://pierre-yves.strub.nu/`