Quelques activités avec Coq à Verimag

Jean-François MONIN

Coq

- Assistant interactif à la preuve (Inria, etc.)
- Extrêmement sûr (noyau, de confiance, preuves-objet)
- Utilisé en informatique et en maths
  compilation certifiée, algèbre, topologie...
- Interactivité entre
  - mécanique fastidieuse
  - cerveau humain imaginatif
- Fondement théoriques :
  - Logique d'ordre supérieur
  - Lambda-calcul
  - Théorie des types
  - Types inductifs

Recherches à Verimag : vérification et méthodes formelles

Applications (suivant l'air du temps)

- Compîlation certifiée
  - CompCert (langage C) cf. exposé Sylvain Boulmé
  - Estérel
- Algorithmes distribués
- OS certifié(CertiKos)
- Sécurité

Outillage

- Raisonnements par congruence
- Petites inversions

Algorithmes distribués

Projet PADEC

- algorithmes auto-stabilisants (très robustes, tolérants aux fautes)
- terminaison
- propriétés quantitatives (-bornes, complexité)
- raisonnements par co-induction, quotients

Petites inversions

Raisonnement par inversion

- Forme de raisonnement très courante sur les structures inductives, en particulier pour raisonner sur des sémantiques de langages
- Applications multiples (Verimag : compilation certifiée, PADEC, Esterel, congruence, enseignement, etc.)
- Permet de donner un peu de pouvoir calculatoire aux relations pensées comme des fonctions
- En Coq standard : `inversion` historique
  Fonctionne très bien mais. . .

Inversion, simple example

Even natural numbers

```
Inductive even : ∀ n, Prop :=
  | Ev0 :                even 0
  | Ev2 n : even n → even (S (S n)).
```

Basic usage

```
Lemma even_plus_left n m : even n → even (n + m) → even m.

IHen : even (n + m) → even m
enm : even (S (S (n + m)))
==============================
even m
```

Basic small inversion on even

```
Inductive even : ∀ n, Prop :=
  | Ev0 :              even 0
  | Ev2 n : even n → even (S (S n)).

Inductive even0 : Prop := even0_Ev0 : even0.
Inductive even1 : Prop :=.
Inductive even2 n : Prop := even2_Ev2 : even n → even2 n.
Definition even_inv {n} (e : even n) :
    match n return Prop with
    | 0       => even0
    | 1       => even1
    | S (S n) => even2 n
    end.
Proof. destruct e; constructor; assumption. Defined.
```

Basic small inversion on even

```
Inductive even : ∀ n, Prop :=
  | Ev0 :              even 0
  | Ev2 n : even n → even (S (S n)).

Inductive even0 : Prop := even0_Ev0 : even0.
Inductive even1 : Prop :=.
Inductive even2 n : Prop := even2_Ev2 : even n → even2 n.
Definition even_inv {n} (e : even n) :
    match n return Prop with
    | 0        => even0
    | 1        => even1
    | S (S n) => even2 n
    end.
Proof. destruct e; constructor; assumption. Defined.
```

Inversion

Purpose

Extract the information contained in a hypothesis H of type T

- where T is an inductive relation
- with some arguments having an inductive type

Expectations

- For each case (constructor), decompose H into ALL its components
- In particular, remove irrelevant cases

Essentially : (subtle) case analysis on H

- Simultaneous case analysis on H and its arguments
- game on dependent pattern-matching

Smaller inversion (part of the Braga method)

Joint work with Dominique Larchey Wendling [TYPES'18], [Proof&Computation II 2021]

Half of even numbers

```
Fixpoint half n (e: even n) {struct e} : nat :=
  match n return even n → nat with
  | 0       => λ _, 0
  | 1       => λ e, match even_inv e with end
  | S (S n) => λ e, S (half n (πeven e))
  end e.
```

Projection: getting ONE STRUCTURALLY SMALLER component

```
Definition πeven n (e: even (S (S n))) : even n :=
  match e in even m return
     let n := match m with S (S n) => n | _ => n end in
     let G := match m with S (S n) => True | _ => False end in G → even n
  with
  | Ev2 n e => λ _, e
  | _         => λ fa, match fa with end
 end I.
```

Smaller inversion (part of the Braga method)

Joint work with Dominique Larchey Wendling [TYPES'18], [Proof&Computation II 2021]

Half of even numbers

```
Fixpoint half n (e: even n) {struct e} : nat :=
  match n return even n → nat with
  | 0       => λ _, 0
  | 1       => λ e, match even_inv e with end
  | S (S n) => λ e, S (half n (πeven e))
  end e.
```

Projection: getting ONE STRUCTURALLY SMALLER component

```
Definition πeven n (e: even (S (S n))) : even n :=
  match e in even m return
      let n := match m with S (S n) => n | _ => n end in
      let G := match m with S (S n) => True | _ => False end in G → even n
  with
  | Ev2 n e => λ _, e
  | _          => λ fa, match fa with end
 end I.
```

Reasoning on half

Easy (induction on e)

```
Lemma double_half :   n e, half n e + half n e = n.
```

Less easy: induction on e and inversion on e'

```
Lemma half_pirr : ∀ n (e e' : even n), half n e = half n e'.

e : even n
e' : even (S (S n))
================================
S (half n e) = half (S (S n)) e'
```

Improved small inversion on even with built-in injectivity

```
Inductive even : ∀ n, Prop :=
  | Ev0 :             even 0
  | Ev2 n : even n → even (S (S n)).

Inductive is_Ev0 : even 0 → Prop := is_Ev0_intro : is_Ev0 Ev0.
Inductive no_Ev1 : even 1 → Prop :=.
Inductive is_Ev2 n : even (S (S n)) → Prop :=
  is_Ev2_intro :   (e : even n), is_Ev2 n (Ev2 n e).

Definition even_inv {n} (e :  even n) :
    match n return even n → Prop with
    | 0       => is_Ev0
    | 1       => no_Ev1
    | S (S n) => is_Ev2 n
    end e.
Proof. destruct e; constructor. Defined.

(* Basic version *)
Inductive even0 : Prop := even0_Ev0 : even0.
Inductive even1 : Prop :=.
Inductive even2 n : Prop := even2_Ev2 : even n → even2 n.
```

Thèse en cours

Automatisation des petites inversions

- Stage M1 Corentin Thomazo
- Basile Gros, démarré en oct 2023

Material

The Braga method

`https://github.com/DmxLarchey/The-Braga-Method`

Dominique Larchey-Wendling and Jean-François Monin.
The Braga Method: Extracting Certified Algorithms from Complex Recursive Schemes in Coq, chapter 8, pages 305–386.
In Klaus Mainzer, Peter Schuster, and Helmut Schwichtenberg, editors.
Proof and Computation II: From Proof Theory and Univalent Mathematics to Program Extraction and Verification.
World Scientific, September 2021.

Small inversions

`http://home/jf/www/Proof/Small_inversions/2022/`