A Quantifier Elimination Procedure over ACFs in Coq using Maple

Micaela Mayero[1] & David Delahaye[2]

[1]LCR-LIPN
Micaela.Mayero@lipn.univ-paris13.fr
http://www-lipn.univ-paris13.fr/~mayero/

[2]CPR-CEDRIC
CNAM (Paris)
David.Delahaye@cnam.fr
http ://cedric.cnam.fr/~delahaye/

Introduction

We focus on :
– Theorem Proving
– Computer Algebra

Two distinct domains with :
– Their own communities
– Rather few interactions

However, they are quite complementary,
especially w.r.t. their *weak points*.

In a Deduction System (DS),
→ hard to perform efficient computations.

In a Computer Algebra System (CAS),
→ no notion of consistency.

Approaches

CASs $\rightarrow$ dedicated to computations
DSs $\rightarrow$ dedicated to validation
$\Rightarrow$ make them interact.

Several approaches :

1. To import validation into CASs
2. To import computations into DSs
   - *believing approach*
   - *skeptical approach*
   - *autarkic approach*
3. To build a system with both

Our approach

We focus on the method (2), skeptical approach :
$\rightarrow$ Interface between Coq and Maple.

Based on the design of the tactic `Field` :
$\rightarrow$ To prove *automatically* equalities over fields.

With this interface, we can :
– Use, in Coq, Maple functions over fields
– Prove these computations with `Field`

The computation/proof paradigm (1/2)

Pure computations :

- No verification needed
- No knowledge of the imported functions required
- New computational behavior
- Very general (not only for Computer Algebra)

The computation/proof paradigm (2/2)

Validating computations :
– When used inside a proof
– Verified w.r.t. the computation/deduction rules

$$\Gamma \vdash P(t),\ t' = f_{\mathsf{ext}}(t)$$

$$\frac{\Gamma \vdash t = t' \qquad \Gamma \vdash P(t')}{\Gamma \vdash P(t)}\ (=_{\mathsf{Cont}})$$

$t = t'$ must be proved :
– Either using a function g_{prf} s.t. $g_{\mathsf{prf}}(t) = t'$
– Or using the deduction rules
– Either manually (*oracle*)
– Or automatically (tactic)

Interfacing Coq and Maple

Pure computations :
`Eval` < Maple *function* > `in` < Coq *term* >

New Coq tactics :
< Maple *function* > < Coq *term* > `.`

Imported Maple functions :

Maple function	Action
`simplify`	Simplify an expression
`factor`	Factor a multivariate polynomial
`expand`	Expand an expression
`normal`	Normalize a rational expression

Implementation

Term transfer between Coq and Maple
$\rightarrow$ Reflexive coding of `Field`.

Computations in proofs :
– Cut of $t = t'$ + rewriting of t into t'
– Proof of $t = t'$ with `Field`

Regarding the interface :
– Realized in a very light way
– No specific architecture, no state
– A simple pipe created for each call
– Asymmetric role : Coq $\rightarrow$ Maple

Available as a Coq (V7.3$\uparrow$) contribution :
– Chalmers/MapleMode, about 300 lines of ML (quite short)

Examples (1/2)

$\forall x, y \in \mathbb{R}$, if $x \neq 0$ and $y \neq 0$ then :

$$(\frac{x}{y} + \frac{y}{x})\ x.y - (x.x + y.y) + 1 > 0$$

```
simp1 < intros.

  x,y : R
  H : x <> 0
  H0 : y <> 0
  ============================
   ( x / y + y / x ) * x * y - ( x * x + y * y ) + 1 > 0

simp1 < simplify (x/y+y/x)*x*y-(x*x+y*y)+1.
2 subgoals

  H : x <> 0
  H0 : y <> 0
  ============================
   1 > 0

 subgoal 2 is:
 y * x <> 0
```

Examples (2/2)

Computation of a characteristic polynomial :

$$\begin{vmatrix} -1-x & 4 & 2 \\ -5 & 7-x & 5 \\ 7 & -6 & -6-x \end{vmatrix}$$

Expansion w.r.t. the first column :

```
Coq < Definition car1 (x:R) := Eval Factor in
Coq < (-1-x)*((7-x)*(-6-x)+30)+5*(4*(-6-x)+12)+
Coq <   7*(20-2*(7-x)).
car1 is defined

Coq < Print car1.
car1 = fun x : R => -( x + -1 ) * ( x + -2 ) * ( x + 3 )
     : R->R
```

Extensions

This interface can be (easily) extended :

- To other functions with an arity of 1 (direct)
- To other function with higher arities

- To other computations (limits, derivatives, gcd's, etc)

Quantifier elimination (1/3)

Algebraically closed field (ACF) :
$\forall P \in K[X].\deg(P) > 0 \Rightarrow \exists x \in K.P(x) = 0$

Formal statement :

Given the polynomials $P_{i,\ i=1..n}$ and $Q_{j,\ j=1..m}$.
The problem (Φ)
$\exists x \in K.P_1(x) = 0 \wedge \ldots \wedge P_n(x) = 0 \wedge Q_1(x) \neq 0 \wedge \ldots \wedge Q_m(x) \neq 0$
is decidable.

which is equivalent to show $\Phi \vee \neg\Phi$ (constructively)
$\hookrightarrow \Phi$: ACF's definition
$\hookrightarrow \neg\Phi$: it fails

Quantifier elimination (2/3)

Auxiliary lemmas

Proposition 1 :
$G = \gcd(P_1, P_2)$. We have :
$\exists x.P_1(x) = 0 \wedge P_2(x) = 0$ iff $\exists x.G(x) = 0$.

Proposition 2 :
If $Q \neq 0$ then $\exists x.Q(x) \neq 0$.

Proposition 3 :
P and Q are relatively prime. We have :
$\exists x.P(x) = 0 \wedge Q(x) \neq 0$ iff $\exists x.P(x) = 0$.

Proposition 4 :
$G = \gcd(P, Q)$ and P_1 s.t. $P(x) = G(x)P_1(x)$. We have :
$\exists x.P(x) = 0 \wedge Q(x) \neq 0$ iff $\exists x.P_1(x) = 0 \wedge G(x) \neq 0$.

Proposition 5 :
$G = \gcd(P, Q)$ and P_1 s.t. $P(x) = G(x)P_1(x)$. If P and Q are not relatively prime then $\deg(P_1) < \deg(P)$.

Quantifier elimination (3/3)

Algorithm

I. if $n \neq 0$ then compute $P(x) = \gcd(P_{i,i=1..n}(x))$ else go to III.

II. if $m = 0$: apply proposition 1. This is equivalent to show $\exists x.P(x) = 0$:
   1. if $\deg(P) = 0$ or $P = 0$: if $P = 0$ then conclude trivially else it fails
   2. else conclude applying ACF's defi nition.

III. if $m \neq 0$:
   1. compute $Q(x) = \prod_{i=1}^{m} Q_i(x)$
   2. if $n = 0$ then apply proposition 2
   3. if $n \neq 0$ then this is equivalent to show $\exists x.P(x) = 0 \wedge Q(x) \neq 0$:
      a. if $\deg(Q) = 0$ or $Q = 0$: if $Q = 0$ then it fails else go to II
      b. if $\deg(P) = 0$ or $P = 0$: if $P = 0$ then go to III2 else it fails
      c. else compute $G(x) = \gcd(P(x), Q(x))$:
         i. if $\deg(G) = 0$, $G(x) = a$ and $a \neq 0$ then apply proposition 3
         ii. else apply proposition 4 and re-apply the algorithm.

Use of Maple

We need :
– gcd (G)
– quotients (P_1, Q_1)
– cofactors (A, B)

Maple : computation of G, P_1, Q_1, A and B.
Coq : Bézout's theorem (converse) to show (automatically) :

$$\begin{cases} P = P_1 G \\ Q = Q_1 G \\ G = AP + BQ \end{cases}$$

Bézout's theorem (converse) :
If G divides P and Q, and if there exists two polynomials A and B s.t. $AP + BQ = G$ then G is the gcd of P and Q.

Example

$parabola(x) = x^2 + 1$
$cubic(x) = x^3 + x^2 + x + 1$
$line(x) = x + 1$

$$\exists x. x^2 + 1 = 0 \wedge x^3 + x^2 + x + 1 = 0 \wedge x + 1 \neq 0$$

3 solutions for $x^2 + 1 = 0$ and $x^3 + x^2 + x + 1 = 0$: i, $-i$ and -1.
The solution -1 does not satisfy $x + 1 \neq 0$.

In Coq :

```
Coq < Goal exists x:C, (UPeval x parabola)=C0/\
Coq <                  (UPeval x cubic)=C0/\~(UPeval x line)=C0.
Unnamed_thm < unfold parabola cubic line; qelim.
Proof completed.
```

Conclusion

Several goals have been achieved :
– Computation/proof paradigm $\rightarrow$ CAS/DS
– Interface between Coq and Maple
– Examples inside and outside proofs
– Extension to the gcd $\Rightarrow$ QElim (ACF, univariate polynomials)

This interface increases the power of computation of Coq
but also its power of automation !

Related work :
– CAS/DS : HOL/Maple (1998), PVS/Maple (2001)
– Quantifier elimination : `Tarski` tactic (2002)

Future work :
– Other computations (limits, derivatives, etc)
– Quantifier elimination :
multivariate case, real closed fields, other algebras (<), etc