A formal quantifier elimination for algebraically closed fields

Cyril Cohen, Assia Mahboubi

INRIA Saclay – Île-de-France
LIX École Polytechnique
INRIA Microsoft Research Joint Centre
[Cyril.Cohen|Assia.Mahboubi]@inria.fr

8 octobre 2010

This work has been partially funded by the FORMATH project, nr. 243847, of the FET program within the 7th Framework program of the European Commission.

CAS

Computer Algebra Systems

$\Downarrow$

Formalize, encode mathematical objects

and *Compute*

Computer Algebra Systems

$\Downarrow$

Formalize, encode mathematical objects
and *Compute*

Examples :

- Formalize operations and factorize, expand expressions
- Formalize polynomials and find their roots (as expressions of the coefficients)

Proof Assistant : formalize more

Allows to formalize a bit more :

- abstract algebraic structures (e.g. groups, rings, etc ...)
- logics and statements (theorems, properties, etc ...)
- proofs
- ...

Proof Assistant : compute too

Allows to compute : evaluate function calls

Proof Assistant : prove

Allows to write proofs and *verify* them

Aim

First Order Theory of Algebraically Closed Fields

First order formulas

- terms :

$$t ::= x \mid k \mid t_1 + t_2 \mid -t \mid t_1 \cdot t_2 \mid t^{-1}$$

where x is a variable and $k \in F$

- formulas :

$$\phi ::= t_1 = t_2 \mid \top \mid \bot \mid \phi_1 \wedge \phi_2 \mid \phi_1 \vee \phi_2 \mid \phi_1 \Rightarrow \phi_2 \mid \neg \phi \mid \exists x, \phi \mid \forall x, \phi$$

First order formulas

- terms :

$$t ::= x \mid k \mid t_1 + t_2 \mid -t \mid t_1 \cdot t_2 \mid t^{-1}$$

  where x is a variable and $k \in F$

- formulas :

$$\phi ::= t_1 = t_2 \mid \top \mid \bot \mid \phi_1 \wedge \phi_2 \mid \phi_1 \vee \phi_2 \mid \phi_1 \Rightarrow \phi_2 \mid \neg\phi \mid \exists x, \phi \mid \forall x, \phi$$

Formalized (encoded) in Coq

First order logic

No quantification on predicates, functions or families.

Example :

- $\forall x\ y,\ \exists z, z * x = y$ is a first order formula
- $\forall P, \exists x, P(x) = 0$ isn't,

First order logic

No quantification on predicates, functions or families.

Example :

- $\forall x\ y,\ \exists z, z * x = y$ is a first order formula
- $\forall P, \exists x, P(x) = 0$ isn't,
  - even if P is only a polynomial : in

$$\forall n, \forall a_0 \dots a_n, \exists x,\ \sum_{i=0}^{n} a_i x^i = 0$$

the number of coefficients depend on n.

First order logic

No quantification on predicates, functions or families.

Example :

- $\forall x\ y,\ \exists z, z * x = y$ is a first order formula
- $\forall P, \exists x, P(x) = 0$ isn't,
    - even if P is only a polynomial : in

      $$\forall n, \forall a_0 \dots a_n, \exists x,\ \sum_{i=0}^{n} a_i x^i = 0$$

      the number of coefficients depend on n.
    - except if n is fixed ...

Axiom schema

A formula for each n :

$$\forall a_0\ a_1,\ a_1 \neq 0 \Rightarrow \exists x, a_1 x + a_0 = 0$$

$$\forall a_0\ a_1\ a_2,\ a_2 \neq 0 \Rightarrow \exists x,\ a_2 x^2 + a_1 x + a_0 = 0$$

$$\forall a_0\ a_1\ a_2\ a_3,\ a_3 \neq 0 \Rightarrow \exists x,\ a_3 x^3 + a_2 x^2 + a_1 x + a_0 = 0$$

$$\dots$$

Expresses that any non constant polynomial has a root.

Algebraically closed field

Field axioms

$+$

Any non constant polynomial has a root

Algebraically closed field

Field axioms

$+$

Any non constant polynomial has a root

Formalized (encoded) in Coq

Deciding First Order Formulas

Goal :

- Given first order formula
  (e.g. $\forall y, xy = 0 \vee (\exists z, zx = 1 \wedge x = z + 2)$)
- Given an evaluation of the parameters (e.g. x)
- Decide whether the formula is true of false.

Deciding First Order Formulas

Goal :

- Given first order formula
  (e.g. $\forall y, xy = 0 \vee (\exists z, zx = 1 \wedge x = z + 2)$)
- Given an evaluation of the parameters (e.g. x)
- Decide whether the formula is true of false.

Coq function :

- input : formal `formula` and values for parameters
- output : `true` or `false`

Quantifier Elimination (entails decidability)

input : $\forall y, xy = 0 \lor (\exists z, zx = 1 \land x = z + 2)$

↓ `q_elim`

output : $x = 0 \lor x = 1 \lor x = 2$

(equivalent formulation, quantifier free)

Quantifier Elimination (entails decidability)

input : $\forall y, xy = 0 \vee (\exists z, zx = 1 \wedge x = z + 2)$

↓

`q_elim`

(For algebraically and real closed fields : Tarski 1957)

output : $x = 0 \vee x = 1 \vee x = 2$

(equivalent formulation, quantifier free)

Datatypes

mathematical objects (polynomials, formulas)

Program

`q_elim`

Proof Assistant

Datatypes

mathematical objects (polynomials, formulas)

Program

`q_elim`

Formal proof

`q_elim` preserves the validity of a formula

Datatypes

mathematical objects (polynomials, formulas)

Program + Formal proof

`q_elim`

`q_elim` preserves the validity of a formula

Coq

(high order)

Reducing the problem : eliminating an $\exists x$

Build a function proj that turns a formula $\exists x, \phi$, where ϕ is quantifier free, into a quantifier free formula.

Reducing the problem : eliminating an $\exists x$

Build a function proj that turns a formula $\exists x, \phi$, where ϕ is quantifier free, into a quantifier free formula.
Then we can define $\mathrm{q_{elim}}$ by :

- $\mathrm{q_{elim}}(\exists x, \phi) = \mathrm{proj}(\exists x, \mathrm{q_{elim}}(\phi))$
- $\mathrm{q_{elim}}(\forall x, \phi) = \neg \mathrm{proj}(\exists x, \mathrm{q_{elim}}(\neg \phi))$

Reducing the problem : eliminating an $\exists x$

Build a function proj that turns a formula $\exists x, \phi$, where ϕ is quantifier free, into a quantifier free formula.
Then we can define $\mathrm{q_{elim}}$ by :

- $\mathrm{q_{elim}}(\exists x, \phi) = \mathrm{proj}(\exists x, \mathrm{q_{elim}}(\phi))$
- $\mathrm{q_{elim}}(\forall x, \phi) = \neg\mathrm{proj}(\exists x, \mathrm{q_{elim}}(\neg\phi))$
- $\mathrm{q_{elim}}(\phi) = \phi$ if ϕ is an atom
- $\mathrm{q_{elim}}(\phi \wedge \psi) = \mathrm{q_{elim}}(\phi) \wedge \mathrm{q_{elim}}(\psi)$
- $\mathrm{q_{elim}}(\phi \vee \psi) = \mathrm{q_{elim}}(\phi) \vee \mathrm{q_{elim}}(\psi)$
- $\mathrm{q_{elim}}(\neg\phi) = \neg\mathrm{q_{elim}}(\phi)$

Without loss of generality, we can suppose that the argument $\exists x, \phi$ of `proj` is of the form :

$$\exists x, \bigwedge_i p_i(x) = 0 \wedge \bigwedge_j q_j(x) \neq 1$$

Where p_i and p_j are polynomials.

Finding the output formula

Find a formula that tells whether :

$\exists x, \bigwedge_i p_i(x) = 0 \wedge \bigwedge_j q_j(x) \neq 0$ is true.

Finding the output formula

Find a formula that tells whether :
$\exists x, \bigwedge_i p_i(x) = 0 \wedge \bigwedge_j q_j(x) \neq 0$ is true.

Algebraic characterization : $\mathrm{size}\left(\mathrm{gdco}_{\prod_j q_j}(\mathrm{gcd}(p_i))\right) \neq 1$

Finding the output formula

Find a formula that tells whether :
$\exists x, \bigwedge_i p_i(x) = 0 \wedge \bigwedge_j q_j(x) \neq 0$ is true.

Algebraic characterization : $\mathrm{size}\left(\mathrm{gdco}_{\prod_j q_j}(\mathrm{gcd}(p_i))\right) \neq 1$

But this *characterization* is not a first order formula as such (size, gdco, gcd defined by *schemata*)

Finding the output formula : example

Example :

$$\exists x, xy + 1 = 0$$

Algebraic characterization :

$$size(yX + 1) \neq 1$$

Finding the output formula : example

Example :

$$\exists x, xy + 1 = 0$$

Algebraic characterization :

$$size(yX + 1) \neq 1$$

$\Rightarrow$ How to express it as a first order formula ?

Finding the output formula : example

Example :

$$\exists x, xy + 1 = 0$$

Algebraic characterization :

$$size(yX + 1) \neq 1$$

$\Rightarrow$ How to express it as a first order formula ?

Intuitively : $y \neq 0$

Transformation : what ?

- We know the program that computes the truth value of :
  $\text{size}\left(\text{gdco}_{\prod_i q_i}(\text{gcd}(p_i))\right) \neq 1$
- Transform it into a program that returns a *formula*
- Characterize the space of parameters

Transformation : how ?

Using Continuation Passing Style !

Turn

$$F : (a : A) \to B$$

into :

$$F_{\text{cps}} : (a : A) \to (k : (B \to \texttt{formula})) \to \texttt{formula}$$

k is called *continuation*

example : handling the degree during a computation

```
function size_cps : input p, k
  if p is a*X^n + q              (a = leading coef of p)
  then return   ( a != 0 /\    k(n+1) )
             \/ ( a = 0  /\ size_cps(q,k) )
  else return k(0)
```

example : $\exists x, xy + 1 = 0$

$\mathrm{size}_{\mathrm{cps}}(yX + 1; k)$ where $k(n) := (n \neq 1)$

example : $\exists x, xy + 1 = 0$

$$\mathrm{size}_{\mathrm{cps}}(yX + 1; k) \qquad \text{where } k(n) := (n \neq 1)$$
$$= \quad (y \neq 0 \wedge k(2)) \vee (y = 0 \wedge \mathrm{size}_{\mathrm{cps}}(1; k))$$

example : $\exists x, xy + 1 = 0$

$$\begin{aligned} & \text{size}_{\text{cps}}(yX + 1; k) \qquad\qquad \text{where } k(n) := (n \neq 1) \\ = \quad & (y \neq 0 \wedge k(2)) \vee (y = 0 \wedge \text{size}_{\text{cps}}(1; k)) \\ = \quad & (y \neq 0 \wedge k(2)) \vee (y = 0 \wedge (1 \neq 0 \wedge k(1)) \vee (1 = 0 \wedge \text{size}_{\text{cps}}(0; k))) \end{aligned}$$

example : $\exists x, xy + 1 = 0$

$$
\begin{aligned}
&\mathrm{size}_{\mathrm{cps}}(yX + 1; k) \qquad\qquad \text{where } k(n) := (n \neq 1) \\
&= (y \neq 0 \wedge k(2)) \vee (y = 0 \wedge \mathrm{size}_{\mathrm{cps}}(1; k)) \\
&= (y \neq 0 \wedge k(2)) \vee (y = 0 \wedge (1 \neq 0 \wedge k(1)) \vee (1 = 0 \wedge \mathrm{size}_{\mathrm{cps}}(0; k))) \\
&= (y \neq 0 \wedge k(2)) \vee (y = 0 \wedge ((1 \neq 0 \wedge k(1)) \vee (1 = 0 \wedge k(0))))
\end{aligned}
$$

example : $\exists x, xy + 1 = 0$

$$
\begin{aligned}
& \mathrm{size_{cps}}(yX + 1; k) \qquad\qquad\qquad\qquad \text{where } k(n) := (n \neq 1) \\
= \quad & (y \neq 0 \wedge k(2)) \vee (y = 0 \wedge \mathrm{size_{cps}}(1; k)) \\
= \quad & (y \neq 0 \wedge k(2)) \vee (y = 0 \wedge (1 \neq 0 \wedge k(1)) \vee (1 = 0 \wedge \mathrm{size_{cps}}(0; k))) \\
= \quad & (y \neq 0 \wedge k(2)) \vee (y = 0 \wedge ((1 \neq 0 \wedge k(1)) \vee (1 = 0 \wedge k(0)))) \\
= \quad & (y \neq 0 \wedge 2 \neq 1) \vee (y = 0 \wedge ((1 \neq 0 \wedge 1 \neq 1) \vee (1 = 0 \wedge 0 \neq 1)))
\end{aligned}
$$

example : $\exists x, xy + 1 = 0$

$$
\begin{array}{ll}
\multicolumn{2}{l}{\mathrm{size_{cps}}(yX + 1; k) \qquad\qquad \text{where } k(n) := (n \neq 1)} \\
= & (y \neq 0 \wedge k(2)) \vee (y = 0 \wedge \mathrm{size_{cps}}(1; k)) \\
= & (y \neq 0 \wedge k(2)) \vee (y = 0 \wedge (1 \neq 0 \wedge k(1)) \vee (1 = 0 \wedge \mathrm{size_{cps}}(0; k))) \\
= & (y \neq 0 \wedge k(2)) \vee (y = 0 \wedge ((1 \neq 0 \wedge k(1)) \vee (1 = 0 \wedge k(0)))) \\
= & (y \neq 0 \wedge 2 \neq 1) \vee (y = 0 \wedge ((1 \neq 0 \wedge 1 \neq 1) \vee (1 = 0 \wedge 0 \neq 1))) \\
\sim & y \neq 0 \qquad\qquad \text{(as foreseen intuitively)}
\end{array}
$$

Conclusion

- We had a procedure and its proof on the paper
- Contributions
  - We write the quantifier elimination procedure and prove it *formally*
  - We show how to turn a boolean procedure into a procedure that outputs a formula, using CPS.
  - We show that CPS is a useful tool to program and prove this kind of procedures
- The procedure is not able to compute in reasonable time yet, because we used naive procedures to compute division, gcd, etc ...

Future work

- Make the procedure compute in reasonable time
- Reuse the CPS trick for Real Closed Fields

The End

Thank you for your attention. Any questions ?