Connecting Gröbner bases programs with Coq to do proofs in algebra, geometry and arithmetics

Loïc Pottier
INRIA Sophia Antipolis

Abstract

We describe how we connected three programs that compute Gröbner bases [1] to Coq [11], to do automated proofs on algebraic, geometrical and arithmetical expressions. The result is a set of Coq tactics and a certificate mechanism [1]. The programs are: F4 [5], GB [4], and gbcoq [10]. F4 and GB are the fastest (up to our knowledge) available programs that compute Gröbner bases. Gbcoq is slow in general but is proved to be correct (in Coq), and we adapted it to our specific problem to be efficient. The automated proofs concern equalities and non-equalities on polynomials with coefficients and indeterminates in R or Z, and are done by reducing to Gröbner computation, via Hilbert's Nullstellensatz. We adapted also the results of [7], to allow to prove some theorems about modular arithmetics. The connection between Coq and the programs that compute Gröbner bases is done using the "external" tactic of Coq that allows to call arbitrary programs accepting xml inputs and outputs. We also produce certificates in order to make the proof scripts independant from the external programs.

1 Introduction

Proof assistants contain now more and more automatic procedures that generate proofs in specific domains. In the Coq system, several tactics exist, for example the `omega` tactic which proves inequalities between linear expressions with integer variables, the `fourier` tactic which does the same thing with real numbers, the `ring` and `field` tactic, which proves equalities between expressions in a ring or a field, the `sos` tactic which proves some inequalities on real polynomials. We describe here a new tactic, called `gb`, which proves (non-)equalities in rings using other (non-)equalities as hypotheses. For example $\forall xy : R, x^2 + xy = 0,\ y^2 + xy = 0 \Rightarrow x + y = 0$, or $\forall x : R, x^2 \neq 1 \Rightarrow x \neq 1$.

This tactic uses external efficient programs that compute Gröbner bases, and their result to produce a proof and a certificate.

We wrote such a tactic several years ago [9], but using only the gbcoq program, which were rather slow. So the tactic remained experimental and was not included in the Coq system. There are also similar tactics in other proof systems: in hol-light, John Harrison wrote a program that computes Gröbner bases to prove polynomial equalities, specially in arithmetics [7]. This program was adapted in Isabelle by Amine Chaieb and Makarius Wenzel for the same task [2]. We show on examples that our tactic is faster.

This paper is organized as follow. In section 2 we explain the mathematical method we use to reduce the problem to Gröbner bases computations. In section 3 we detail the tactic and the way it builds a proof in Coq. In section 4 we show how we connected Coq to the specialized programs that computes Gröbner bases. Section 5 details the complete tactics that proves also non-equalities, and section 6 shows how to produce certificates and then save time in the proof script. In section 7 we give some examples of utilisations of the tactic in algebra, geometry and arithmetics, with comparisons with hol-light[6] . Section 8 contains the conclusion and perpectives of this work.

Rudnicki P, Sutcliffe G., Konev B., Schmidt R., Schulz S. (eds.);
Proceedings of the Combined KEAPPA - IWIL Workshops, pp. 67-76
[1] downloadable at `http://www-sop.inria.fr/marelle/Loic.Pottier/gb-keappa.tgz`

2 Hilbert Nullstellensatz

Hilbert Nullstellensatz shows how to reduce proofs of equalities on polynomials to algebraic computations (see for example [3] for the notions introduced in this section).

It is easy to see that if a polynomial P in $K[X_1,\ldots,X_n]$ verifies $P^r = \sum_{i=1}^{s} Q_i P_i$, with r a positive integer, Q_i and P_i also in $K[X_1,\ldots,X_n]$, then P is zero whenever polynomials $P_1,...,P_s$ are zero.

Then we can reduce the proof of $P_1 = 0,\ldots,P_s = 0 \Rightarrow P = 0$ to find $Q_1,\ldots,Q_s$ and r such that $P^r = \sum_i Q_i P_i$.

The converse is also true when K is algebraically closed: this is the Hilbert Nullstellensatz. In this case, the method is complete.

Finding $P^r = \sum_i Q_i P_i$ can be done using Gröbner bases, as we will explain now.

Recall that an *ideal* $\mathscr{I}$ of a ring is an additive sub-group of the ring such that $ax \in \mathscr{I}$ whenever $a \in \mathscr{I}$. The ideal *generated* by a family of polynomials is the set of all linear combinations of these polynomials (with polynomial coefficients).

A *Gröbner basis* of an ideal is a set of polynomials of the ideal such that their head monomials (relative to a choosen order on monomials, e.g. lexicographic order, or degree order) generates the ideal of head monomials of all polynomials in the ideal. The main property of a Gröbner basis is that it provides a test for the membership to the ideal: a polynomial is in the ideal iff its euclidian *division* by the polynomials of the basis gives a zero remainder. The division process is a generalisation of the division of polynomials in one variable: to divide a polynomial P by a polynomial $aX^\alpha - Q$ we write $P = aX^\alpha S + T$ where T contains no monomial that is multiple of X^α. Then change P with $QS+T$ and repeat divison. The last non zero T is the remainder of the division. To divide a polynomial by a family of polynomials, we repeat this process with each polynomial of the family. In general, the remainder depends on the order we use the polynomials of the family. But with a Gröbner basis, this remainder is unique (this is a characteristic property of Gröbner basis).

2.1 Method 1: how to find $Q_1,\ldots,Q_s$ such that $1 = \sum_i Q_i P_i$

Compute a Gröbner base of the polynomials $\{tP_i - e_i,\ e_i e_j,\ e_i t\}_{i,j}$ (where $t, e_1,\ldots,e_s$ are new variables) with an order such that $t > X_i > e_i$.

Suppose that, in this basis, there is a polynomial of the form $t - \sum_i Q_i e_i$. This polynomial is then in the ideal generated by $\{tP_i - e_i,\ e_i e_j,\ e_i t\}_{i,j}$, so is a linear combination of these polynomials:

$t - \sum_i Q_i e_i = \sum_i h_i(tP_i - e_i) + \sum_{ij} g_{ij} e_i e_j + \sum_i k_i e_i t$

e_i are formal variables, so we can substitute formally e_i with tP_i, and we obtain $t(1 - \sum_i Q_i P_i) = 0 + t^2(\sum_{ij} g_{ij} P_i P_j + \sum_i k_i P_i)$.

Then the coefficient of t in this equation must be zero: $1 - \sum_i Q_i P_i = 0$, and we are done.

Note that the polynomials $\{e_i t,\ e_i e_j\}$ are not necessary, but their presence much speed up the computation of the Gröbner basis[2].

2.2 Method 2: how to find $Q_1,\ldots,Q_s$ and r such that $P^r = \sum_i Q_i P_i$

Use the standard trick: search to write $1 = \sum_i h_i P_i + h(1 - zP)$ $(*)$, where z is a new variable. This can be done with the previous method. Suppose we succeed. Let r be the max degree in z of polynomials h_i.

Substitute formally z with $1/P$, and multiply the equation (*) by P^r. Then we obtain $P^r = \sum_i Q_i P_i$, as required, where $Q_i = P^r h_i[z \leftarrow 1/P]$

[2] thanks to Bernard Mourrain for this trick

2.3 Completness

It is easy to see that methods 1 and 2 are complete in the sense that if $P^r = \sum_i Q_i P_i$ holds, there will find such an equation:

- method 1: suppose $1 - \sum_i Q_i P_i = 0$. Then $t = \sum_i Q_i t P_i$, and $t - \sum_i Q_i e_i = \sum_i Q_i (t P_i - e_i)$. Hence $t - \sum_i Q_i e_i$ belongs to the ideal of which we have computed a Gröbner basis. Because of the order we have choosen on variables, this implies that there is a polynomial $t - \sum_i h_i e_i$ in the Gröbner basis.
- method 2: suppose $P^r = \sum_i Q_i P_i$. We have $1 - z^r P^r = (1 + zP + \ldots + z^{r-1} P^{r-1})(1 - zP)$. Replacing P^r with $\sum_i Q_i P_i$ we obtain $1 = z^r (\sum_i Q_i P_i) + (1 + zP + \ldots + z^{r-1} P^{r-1})(1 - zP)$.

2.4 Example

Take $p = x + y,\ p_1 = x^2 + xy,\ p_2 = y^2 + xy$. With the previous method, the Gröbner basis is:

$$
\begin{array}{l}
t - zye_0 - zxe_0 - z^2 e_1 - z^2 e_2 - e_0 \\
y^2 e_0 - x^2 e_0 + zye_1 - zxe_2 - e_1 + e_2 \\
yxe_0 + x^2 e_0 + zye_2 + zxe_2 - e_2 \\
e_0^2 \\
xe_1 - ye_2 \\
e_0 e_1, e_1^2, e_0 e_2, e_1 e_2, e_2^2
\end{array}
$$

we obtain $r = 2,\ Q_1 = 1,\ Q_2 = 1$, and then $(x+y)^2 = 1 \times (x^2 + xy) + 1 \times (y^2 + xy)$. Which proves that $x^2 + xy = 0,\ y^2 + xy = 0 \Rightarrow x + y = 0$.

3 Proof in Coq

Coq [11] is a proof assistant based on type theory, where we can interactively build proofs of *goals*, which are logical assertions of the form $\forall H_1 : T_1, \ldots, \forall H_n : T_n, C(H1, \ldots, H_n)$. Using tactics, we can simplify the goal, while the system builds the corresponding piece of proof.

Typically we will treate goals of the form:

```
x : Z
y : Z
H : x ^ 2 + x * y = 0
H0 : y ^ 2 + x * y = 0
============================
 x + y = 0
```

Here hypotheses are variables belonging in a ring or a field, and equalities between polynomials.

We explain now how to compute and use the Nullstellensatz equation to build a proof of this goal in Coq. The steps are: syntaxification, Gröbner basis computation, and building the proof from the Nullstellensatz equation.

3.1 Syntaxification

We begin by building polynomials from the three equations in this goal. This is done in the tactic language of Coq (LTAC, which is a meta-language for computing tactics and executing them) by first computing the list of variables:

```
lv = (cons y (cons x nil))
```

and the list of polynomials:

```
lp = (cons (Add (Pow (Var 2) 2) (Mul (Var 2) (Var 1)))
     (cons (Add (Pow (Var 1) 2) (Mul (Var 2) (Var 1)))
     (cons (Sub (Add (Var 2) (Var 1)) (Const 0 1))
      nil)))
```

Variables are represented by their rank in the list of variables. Polynomials are elements of an inductive type, and we can recover the equations by interpreting them in Z with the list of variables. For example,

```
(interpret (Add (Pow (Var 2) 2) (Mul (Var 2) (Var 1)))
           lv)
```

evaluates in `x^2 + x * y`.

We used parts of the code of the `sos`[8] tactic, written by Laurent Théry.

3.2 Calling Gröbner basis computation

We call the external program gb (see section 4) with the list of polynomials; here we choose the program F4 to compute Gröbner basis:

```
external "./gb" "jcf2" lp
```

The result is the term:

```
(cons
   (Pow
     (Add
        (Add Zero
           (Mul
              (Add (Add Zero (Mul (Const 0 1) (Const 1 1)))
                 (Mul (Const 1 1) (Pow (Var 1) 1))) (Const 1 1)))
        (Mul (Const 1 1) (Pow (Var 2) 1)))
      2)
(cons (Const 1 1) (cons(Const 1 1) (cons (Const 1 1) nil))))
```

which has the structure

```
(cons (Pow p d) (cons c lq))
```

such that the Nullstellensatz equation holds:

$$cp^d = \sum_{q_i \in lq} q_i p_i$$

Here, we have $lq = q_1, q_2$, $q_1 = q_2 = 1$

3.3 Building the proof from the Nullstellensatz equation

After interpreting the polynomials q_1 and q_2 in Z using the original list of variables, we get and prove easily the goal

```
1 * (x + y)^2 = 1 * (x ^ 2 + x * y) + 1 * (y ^ 2 + x * y)
```

by the ring tactic.

To prove the original goal, it is now sufficient to rewrite `x^2 + x * y` and `y^2 + x * y` by 0, getting `1 * (x + y)^2 = 0`, and, using a simple lemma, we get `x + y = 0` and we are done.

4 Connecting F4, GB, and gbcoq to Coq

Coq allows to call arbitrary external programs via a function called "external". It sends Coq terms in xml format (i.e. as tree) to the standard output of the external program, and gets its standard output (also in xml format) as a resulting Coq term. We use this function to compute a Gröbner basis of a list of polynomials, via a single interface to three specialized programs: F4, GB, and gbcoq. This interface, called "gb" is written in ocaml. It translates the list of polynomials given as standard input in xml format in the format of the choosen program (F4, GB or gbcoq), call it with the good arguments, get its result (a Gröbner basis, if no error occured), selects its useful information, translates it in xml and sends it as result to standard output. More precisely:

- F4 is a C library, and has only an interface for Maple. We wrote a simple parser of polynomials to use it on command line, helped by J.C. Faugère.
- GB is also written in C and has a command line interface, or accept inputs in a file; with a Maple-like syntax for polynomials.
- Gbcoq is written in ocaml, so is integrated to gb. This program uses an Buchberger-like algorithm which has been extracted from Coq. So it is proven to be correct. We added recently an optimisation which reduces drastically the time to compute Nullstellensatz equations: each time we add a new polynomial during the completion via the reduction of critical pairs, we divide the polynomial that we want to test if it is in the ideal, by the current family of polynomials. If this gives zero, then we stop:, and return the Nullstellensatz coefficients, deduced from the divisions we made. More we also try its powers (up to a parametrized limit). Then, when we have computed the whole Gröbner basis, we can compute the Nullstellensatz coefficients, without having to verify that the remaining critical pairs reduce to zero. More, this is often the case that the polynomial reduces to zero with a partial Gröbner basis! The time is sometimes divided by 1000 with such a technique, and always much reduced. Note that such an improvement cannot be made in a blackbox program such as the programs of JC Faugère, which are free but not opensource.

5 The gbR and gbZ tactics in Coq

We wrote two tactics: gbR for real numbers, gbZ for integers. The set of integer is not a field, but we can simulate computations in the field of rational numbers using only integers. In this case, the Nullstellensatz equation become $cp^d = \sum_i q_i p_i$, where c is an integer, and the q_i have integer coefficients.

We can allow negations of equations in the conclusion. For example $xy = 1 \Rightarrow x \neq 0$. The trick is to replace $x \neq 0$ with $x = 0 \Rightarrow 1 = 0$, which is equivalent to add a new equation in hypotheses, and replace the equation to prove with $1 = 0$.

In the case of real numbers, we can allow also negations of equations in hypotheses. For example $x^2 \neq 1 \Rightarrow x \neq 1$. This can be done by introducing new variables, remarking that $p \neq 0 \Leftrightarrow \exists t, p * t = 1$.

In the example, this gives $t(x^2-1)=1 \Rightarrow x \neq 1$. The negation in conclusion can be removed and leads to $t(x^2-1)=1,\ x-1=0 \Rightarrow 1=0$, which is proven using the Nullstellensatz equation $1=1\times(t(x^2-1)-1)+(t+tx)\times(x-1)$

Finally, the tactics use first the program F4. If it fails (for memory limits), then the tactics try GB. If it fails too, then the tactics uses gbcoq. We have also specialised tactics, allowing the user to choose which program to use, between F4, GB, and gbcoq. Indeed, experiments show that no one is better than others.

6 Certificates

Once the Nullstellensatz equation is computed, we can change the proof script, replacing the tactic gb with a similar tactic, called "check_gb" which will not call external programs, but instead it will take as arguments all the components of the Nullstellensatz equation. So, next time we will execute the proof script, for compilation for example, it will not need external Gröbner computation[3]. Let us give an example. Suppose we want to prove:

```
Goal forall x y z:R, x^2+x*y=0 -> y^2+x*y=0 -> x+y=0.
```

we execute the tactic gbR, which proves the goal, and prints these lines in the standard output of Coq:

```
(* with JC.Faugere algorithm F4 *)
gbR_begin; check_gbR
(x + y - 0)
(List.cons (x * (x * 1) + x * y) (List.cons (y * (y * 1) + x * y) List.nil))
(List.cons y (List.cons x List.nil))

(lceq
   (Pow
      (Add
         (Add Zero
            (Mul
               (Add (Add Zero (Mul (Const 0 1) (Const 1 1)))
                  (Mul (Const 1 1) (Pow (Var 1) 1))) (Const 1 1)))
         (Mul (Const 1 1) (Pow (Var 2) 1))) 2)
   (lceq (Const 1 1) (lceq (Const 1 1) (lceq (Const 1 1) lnil))))
 .
```

Then, we can replace the line calling gbR with these tactics lines, which contains no more than the components of the needed Nullstellensatz equation $(x+y)^2=1\times(x^2+xy)+1\times(y^2+xy)$, and then need much less time to evaluate, because it doesn't need Gröbner basis computation.

7 Examples

In this section we give several examples of use of the tactics gbR and gbR.

[3] thanks to Julien Narboux for this suggestion

7.1 Algebra

The following examples uses the symetric expressions of coefficients with roots of a polynomial.

First in degree 3: if x,y,z are the three complex roots of $X^3 + a*X^2 + b*X + c$ then we have $a = -(x+y+z)$, $b = x*y+y*z+z*x$, and $c = -x*y*z$. And then we can prove that $x+y+z = 0 \Rightarrow x*y+y*z+z*x = 0 \Rightarrow x*y*z = 0 \Rightarrow x = 0$, because then the polynomial becomes X^3, and has only 0 as a root.

```
Require gbZ.

Goal forall x y z:Z,
  x+y+z=0 -> x*y+y*z+z*x=0 -> x*y*z=0 -> x=0.
gbZ.
Qed.
```

More complicated, the same thing in degrees 4 and 5:

```
Goal forall x y z u:Z,
  x+y+z+u=0 ->
  x*y+y*z+z*u+u*x+x*z+u*y=0 ->
  x*y*z+y*z*u+z*u*x+u*x*y=0 ->
  x*y*z*u=0 -> x=0.
gbZ.
Qed.

Goal forall x y z u v:Z,
  x+y+z+u+v=0 ->
  x*y+x*z+x*u+x*v+y*z+y*u+y*v+z*u+z*v+u*v=0->
  x*y*z+x*y*u+x*y*v+x*z*u+x*z*v+x*u*v+y*z*u+y*z*v+y*u*v+z*u*v=0->
  x*y*z*u+y*z*u*v+z*u*v*x+u*v*x*y+v*x*y*z=0 ->
  x*y*z*u*v=0 -> x^5=0.
gbZ.
Qed.
```

Last example takes less than 1s with F4 and GB, and gbcoq. With hol-light, it takes 1s.

7.2 Geometry

Desargues theorem is too complicated to be proved with Gröbner bases. But Pappus theorem can. We formalize in Coq the set of points in the real plane:

```
Open Scope R_scope.
Record point:Type:={
 X:R;
 Y:R}.
```

Then we give two definitions of colinearity of three points (the theorem is false if we use only the second definition, because of degenerated configurations):

```
Definition colinear(C A B:point):=
  exists a:R,
 (X C)=a*(X A)+(1-a)*(X B) /\ (Y C)=a*(Y A)+(1-a)*(Y B).

Definition colinear2(A B C:point):=
  (X A)*(Y B)+(X B)*(Y C)+(X C)*(Y A)
  =(Y B)*(X C)+(Y C)*(X A)+(Y A)*(X B).
```

Then we state and prove the Pappus theorem, in a specialized (but without lost of generality) configuration:

```
Lemma pappus: forall A B C A' B' C' D E F:point,
  (X A')=0 -> (X B')=0-> (X C')=0 ->
  (Y A)=0 -> (Y B)=0 -> (Y C) = 0 ->
  colinear D A B' -> colinear D A' B ->
  colinear E A C' -> colinear E A' C ->
  colinear F B C' -> colinear F B' C ->
  colinear2 D E F.
...
gbR_choice 2.
Qed.
```

In this example, F4 fails, GB takes 9s, and gbcoq takes 3s. We also tried hol-light with this example, which takes 77s:

```
./hol

prioritize_int();;

let t1 = Unix.time();;

int_ideal_cofactors
[`XD -( x4 * XA )`;
`YD -((&1 - x4) * YB1)`;
`XD -( (&1 - x3) * XB)`;
`YD - (x3 * YA1)`;
` XE - x2 * XA`;
`YE -  (&1 - x2) * YC1`;
` XE - (&1 - x1) * XC`;
` YE - x1 * YA1`;
` XF - x0 * XB`;
` YF -  (&1 - x0) * YC1`;
` XF - (&1 - x) * XC`;
` YF - x * YB1`]
` XD * YE + XE * YF + XF * YD -(YE * XF + YF * XD + YD * XE)`;;
```

```
Unix.time()-.t1;;
```

The general case of Pappus theorem is too complicated to compute.

7.3 Arithmetics

Following the idea of [7], we can prove statements about coprimality, gcd and divisions. We have to do some work for that, because the tactic gbZ is not sufficient. But the problem is again an ideal membership one, then solvable by Gröbner basis computation. We have written a tactic doing that, called `gbarith`. Here are examples of its use in Coq:

```
Definition divides(a b:Z):= exists c:Z, b=c*a.
Definition modulo(a b p:Z):= exists k:Z, a - b = k*p.
Definition ideal(x a b:Z):= exists u:Z, exists v:Z, x = u*a+v*b.
Definition gcd(g a b:Z):= divides g a /\ divides g b /\ ideal g a b.
Definition coprime(a b:Z):= exists u:Z, exists v:Z, 1 = u*a+v*b.

Goal forall a b c:Z, divides a (b*c) -> coprime a b -> divides a c.
gbarith.
Qed.

Goal forall m n r:Z, divides m r -> divides n r -> coprime m n -> divides (m*n) r.
gbarith.
Qed.

Goal forall x y a n:Z, modulo (x^2) a n -> modulo (y^2) a n -> divides n ((x+y)*(x-y)).
gbarith.
Qed.
```

7.4 Computation times, comparison with hol-light

Previous examples, and more we made, show that no one among F4, GB, gbcoq and is better than others. hol-light is sometimes better than F4 and GB, but gbcoq is much better than hol-light. The reason is simple: we often stop computations before obtaining a Gröbner basis.

8 Conclusion

The ”external” tactic of Coq is a very good tool to use efficient programs to produce proofs in specific domains. We have shown how to use efficient Gröbner bases computations in this context. The use of certificates should be developped to reduce time of re-verification of proofs. The certificate can be written explicitely in the proof script, as we have shown here, but it could be stored in a cache. We have shown the interest of using external programs, but also their limits, as soon as it is impossible or difficult to adapt them to specific use of proof systems. We plan to investigate other decisions procedures, for example polynomial system solving, to produce new tactics in the same spirit.

Acknowledgements: we thank anonymous referees for their suggestions on the redaction of this paper and bibliographical completions.

References

[1] Bruno Buchberger. Bruno buchberger's phd thesis 1965: An algorithm for finding the basis elements of the residue class ring of a zero dimensional polynomial ideal. Journal of Symbolic ComputationVolume 41, Issues 3-4, Logic, Mathematics and Computer Science: Interactions in honor of Bruno Buchberger (60th birthday),, 2006.

[2] Amine Chaieb and Makarius Wenzel. Context aware calculation and deduction. In Manuel Kauers, Manfred Kerber, Robert Miner, and Wolfgang Windsteiger, editors, *Calculemus/MKM*, volume 4573 of *Lecture Notes in Computer Science*, pages 27–39. Springer, 2007.

[3] David Eisenbud. Commutative algebra with a view toward algebraic geometry. Graduate Texts in Mathematics 150. Springer-Verlag, 1999.

[4] Jean-Charles Faugère. Gb. `http://fgbrs.lip6.fr/jcf/Software/Gb/index.html`.

[5] Jean-Charles Faugère. A new efficient algorithm for computing grobner bases (f4). volume 139, issues 1-3 of *Journal of Pure and Applied Algebra*, pages 61–88, 1999.

[6] John Harrison. Towards self-verification of hol light. In Ulrich Furbach and Natarajan Shankar, editors, *Proceedings of the third International Joint Conference, IJCAR 2006*, volume 4130 of *Lecture Notes in Computer Science*, pages 177–191, Seattle, WA, 2006. Springer-Verlag.

[7] John Harrison. Automating elementary number-theoretic proofs using gröbner bases. In Frank Pfenning, editor, *Proceedings of the 21st International Conference on Automated Deduction, CADE 21*, volume 4603 of *Lecture Notes in Computer Science*, pages 51–66, Bremen, Germany, 2007. Springer-Verlag.

[8] John Harrison. Verifying nonlinear real formulas via sums of squares. In Klaus Schneider and Jens Brandt, editors, *Proceedings of the 20th International Conference on Theorem Proving in Higher Order Logics, TPHOLs 2007*, volume 4732 of *Lecture Notes in Computer Science*, pages 102–118, Kaiserslautern, Germany, 2007. Springer-Verlag.

[9] Loïc Pottier Jérôme Créci. Gb: une procédure de décision pour le système coq. Journées Francaises des Langages Applicatifs, Sainte-Marie-de-Ré, pages http://jfla.inria.fr/2004/actes/actes–jfla–2004.tar.gz, 2004.

[10] Loïc Pottier Laurent Théry. gbcoq, 1998. `http://www-sop.inria.fr/croap/CFC/Gbcoq.html`.

[11] The Coq Development Team. The coq proof assistant, 2008. `http://coq.inria.fr/V8.1pl3/refman/index.html`.