Proving the group law for elliptic curves formally

Laurent Théry

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Proving the group law for elliptic curves formally

Laurent Théry

N° 0330

February 2007

Thème SYM

Rapport technique

ISSN 0249-0803 ISRN INRIA/RT--0330--FR+ENG

Proving the group law for elliptic curves formally

Laurent Théry

Thème SYM — Systèmes symboliques
Projet Marelle

Rapport technique n° 0330 — February 2007 — 16 pages

Abstract: This report presents a formal proof of the group law for elliptic curves done in Coq.

Key-words: formal proof, elliptic curves, group law

Unité de recherche INRIA Sophia Antipolis
2004, route des Lucioles, BP 93, 06902 Sophia Antipolis Cedex (France)
Téléphone : +33 4 92 38 77 77 — Télécopie : +33 4 92 38 77 65

Prouver les propriétés de groupe des courbes elliptiques formellement

Résumé : Ce rapport présente une preuve des propriétés de groupe des courbes elliptiques qui a été effectuée dans le système Coq.

Mots-clés : preuve formelle, courbes elliptiques, loi de groupe

1 Introduction

Elliptic curves are a special type of cubic curves. We could expect their formalisation inside a proof system to be straightforward. Unfortunately this is not the case. If defining elliptic curves is easy, proving the properties of the group law is far from being trivial. Associativity is the difficult part. In the literature, this is usually done by a geometric argument using, for example, the nine associated points theorem. Proving this theorem formally would require a huge effort of formalisation. In this paper, we present a simpler algebraic approach that relies on basic polynomial manipulations only. Our main source of inspiration has been the note by Stefan Friedl that presents an elementary proof [3]. To translate this 7 page long paper proof in a theorem prover was a real challenge. In fact, the proof relies on some non-trivial computations that the author advises to check using a computer algebra system such as CoCoa [2]. The main difficulty has been to find an effective way to cope with these computations inside our proof system. Also, in order to keep the formalisation work to a minimum, we modified some of the arguments of the initial proof so to get an even more elementary proof: our formal proof relies on the basic properties of the polynomial ring only.

The paper is structured as follows. In a first section, we recall what elliptic curves are, giving not only the usual informal definition but also our formal one. In a second section, we explain how we deal with the necessary computation steps inside our proof. In a third section, we present our proof.

2 Defining elliptic curves and the group law

Elliptic curves are defined over an arbitrary field $K[0, 1, +, -, *, /]$ of characteristic other than 2. Curves are parametrised by two values A and B such that the discriminant $4A^3 + 27B^2$ is not equal to zero. Elements of the curve are 0, the zero element, and the points (x, y) such that $y^2 = x^3 + Ax + B$. Given a point p of the curve, we define $-p$ as

- If $p = 0$ then $-p = 0$
- If $p = (x, y)$ then $-p = (x, -y)$

Given two points p_1 and p_2 on the curve we define $p_1 + p_2$ as

- If $p_1 = 0$ then $p_1 + p_2 = p_2$
- If $p_2 = 0$ then $p_1 + p_2 = p_1$
- If $p_1 = -p_2$ then $p_1 + p_2 = 0$
- If $p_1 = (x, y) = p_2$ and $y \neq 0$ then let $l = (3x^2 + A)/2y$ and $x_1 = l^2 - 2x$ in $p_1 + p_2 = (x_1, -y - l(x_1 - x))$.
- If $p_1 = (x_1, y_1)$ and $p_2 = (x_2, y_2)$ and $x_1 \neq x_2$ then let $l = (y_2 - y_1)/(x_2 - x_1)$ and $x_3 = l^2 - x_1 - x_2$ in $p_1 + p_2 = (x_3, -y_1 - l(x_3 - x_1))$.

We also define substraction as $p_1 - p_2 = p_1 + (-p_2)$.

The formal definition is somewhat more intricate. Being in a proof system like Coq, it amounts in defining a type `elt` that represents exactly the element of a curves. It is represented by the following definition

```
Inductive elt: Set :=
  inf_elt: elt
| curve_elt x y (H: is_eq y^2  (x^3 +  A * x + B) = true):  elt.
```

An element of type `elt` is either 0 (`inf_elt`) or an element of the curve (`curve_elt`) that contains an `x`, a `y` and a proof `H` that insures that the point $(\mathtt{x}, \mathtt{y})$ is on the curve. The particular formulation of the statement for `H` is a bit technical. It uses a function `is_eq` that tests the equality of two elements of K, i.e (`is_eq` x y) returns `true` iff $x = y$. With this particular formulation, we have that (`curve_elt` x_1 y_1 H_1) = (`curve_elt` x_2 y_2 H_2) iff $x_1 = x_2$ and $y_1 = y_2$ since there is only one proof of `true` = `true` in the logic of Coq.

A consequence of having proofs inside curve elements is that the definition of the two operations $-$ (`opp`) and $+$ (`add`) require the proofs that the operations are internal. This can be seen in the definition of the opposite

```
Definition opp p :=
   match p with inf_elt =>
     inf_elt
   | curve_elt x y H => curve_elt x (-y) opp_lem end.
```

where the `opp_lem` represents the proof that if we know that $y^2 = x^3 + Ax + B$ then $(-y)^2 = x^3 + Ax + B$. This proof is trivial, this is not the case anymore for the addition

```
Definition add: p1 p2 =>
   match p1 with
     inf_elt => p2
  | curve_elt x1 y1 H1 =>
     match p2 with
       inf_elt => p1
     | curve_elt x2 y2 H2 =>
         if is_eq x1 x2 then
            if is_eq y1 (-y2) then inf_elt
            else
                let l := (3*x1*x1 + A)/(2*y1) in
                let x3 := l^2 - 2 * x1 in
                  curve_elt x3 (-y1 - l * (x3 - x1)) add_lem1
         else
           let l := (y2 - y1)/(x2 - x1) in
           let x3 := l ^ 2 - x1 -x2 in
             curve_elt x3 (-y1 - l * (x3 - x1)) add_lem2
     end
```

```
end.
```

where the two proofs `add_lem1` and `add_lem2` represent the property that the operation is internal in the tangent case and the generic case respectively.

3 Deciding equalities

In order to illustrate which kind of automation is needed in proving properties of the group law, let us consider the proof of `add_lem1`. It amounts in proving the following goal

let $l = (3x^2 + A)/2y$ in
let $x_1 = l^2 - 2x$ in
$(-y - l(x_1 - x))^2 - (x_1^3 + Ax_1 + B) = 0$

under the assumption that $y^2 = x^3 + Ax + B$. The expression to be proved equal to zero is rational. The first step is to normalise it into the form $N/D = 0$ where N and D are two polynomial expressions. So the initial goal can be reduced to $N = 0$. In our example, a naive normalisation gives

$$2^{10}y^8 - 2^{10}y^6x^3 - 2^{10}Ay^6x - 2^{10}By^6 = 0$$

Now, we replace y^2 by $x^3 + Ax + B$

$$2^{10}(x^3+Ax+B)^4 - 2^{10}(x^3+Ax+B)^3x^3 - 2^{10}A(x^3+Ax+B)^3x - 2^{10}B(x^3+Ax+B)^3 = 0$$

Finally, using distributivity, associativity, commutativity and collecting equal monomials, everything cancels out.

To sum up, three ingredients are needed to automate the proof of lemmas like `add_lem1`: a procedure to normalise polynomial expressions (last step), a procedure to rewrite polynomial expressions (second step) and a procedure to normalise rational expressions (first step).

The first procedure to normalise polynomial expressions was already present in Coq and is described in [4]. Its main characteristic is to use an internal representation of polynomials in Horner form. This representation is unique up to variable ordering. Given a polynomial P and a variable x we write P as $P_1 + x^iQ_1$ where x does not occur in P_1 and is not a common factor in Q_1 and we proceed on P_1 and Q_1 recursively. The normal form is further simplified writing 0 for $m0$ and P for $0 + P$ and $P + 0$. For example

$$2^{10}y^8 - 2^{10}y^6x^3 - 2^{10}Ay^6x - 2^{10}By^6$$

is represented as

$$y^6((B(-2^{10}) + x(A(-2^{10}) + x^2(-2^{10}))) + y^2 2^{10})$$

with the order $y > x > A > B$. Once defined procedures to add and multiply polynomials in Horner form, the normal form of a polynomial P is obtained by a structural traversal of P. As described in [4], this leads to a very effective way of proving ring equalities by just checking that the normal form of the left side of the equality is structurally equal to the normal form of the right side of the equality.

Rewriting has been implemented in a very naive way on Horner representation. It uses a simple procedure that given a monomial m splits a polynomial P in a pair (P_1, Q_1) in such a way that $P = P_1 + mQ_1$. Now rewriting once with the equation $m = R$ is performed by forming the polynomial $P_1 + RQ_1$. For example, if we want to rewrite the previous polynomial with the equation $y^2x^2 = z + t$, we first split the polynomial

$$y^6((B(-2^{10}) + x(A(-2^{10}) + x^2(-2^{10}))) + y^2 2^{10})$$

into

$$(\ y^6((B(-2^{10}) + x(A(-2^{10}))) + y^2 2^{10})\ ,\ y^4 x(-2^{10})\ )$$

We now form

$$(y^6((B(-2^{10}) + x(A(-2^{10}))) + y^2 2^{10})) + (z + t)(y^4 x(-2^{10}))$$

and normalise it. For rewriting with respect to a list of equations, we just iterate the single rewriting operation for all the equations in a fair way. Note that for elliptic curve we are going to rewrite with equations of the type $y_i^2 = x_i^3 + Ax_i + B$, so we take a special care in choosing a variable ordering for the Horner representation such that the y_i are privileged. Doing so, the splitting procedure has only to visit a small part of the polynomial. We developed a first version of the rewriting procedure. It has been further improved and integrated to the Coq system by Benjamin Grégoire.

Normalising rational equalities is not as simple as for polynomial expressions. Since when reducing the initial expression to a common denominator the expressions for the numerator and the denominator can grow quickly. Some factorisation is needed. For example, when reducing $P_1/Q_1 + P_2/Q_2$ we can do much better than forming the naive fraction $(P_1Q_2 + P_2Q_1)/Q_1Q_2$. Unfortunately, gcd algorithms for multivariate polynomials are far more complex to implement than the simple euclidean algorithm for univariate polynomials. So, we have just implemented the heuristic that tries to factorise the common product in Q_1 and Q_2 by a simple traversal of the list of products that composed Q_1 and Q_2. In practise, this simple optimisation seems sufficient to get reasonable expression. For example, applying it to $1/x + 1/xy + 1/y = 0$ we get $x + y + 1 = 0$. Once again, we developed a first version of this normalising algorithm. It has then been greatly improved and integrated to the Coq system by Bruno Barras.

The three procedures (polynomial normaliser, rewriter and rational normaliser) are put together in a single tactic `field`. Calling `field[H₁ H₂]` attempts to solve the goal $F = 0$ using the rewriting rule $\mathtt{H}_1$ and $\mathtt{H}_2$. The three procedures have been defined inside the Coq logic using the two level approach [1]. Their correctness can then be stated inside the proof system and formally proved. This insures that the `field` tactic only performs valid simplifications. A key aspect of this tactic is its efficiency. Proving the group law properties requires non trivial computation. Having a relatively efficient procedure is mandatory to be able to complete the proof. In the following, every time this tactic is used, we indicate its running time on a 2 GHz Pentium M with 1 Gigabyte of RAM. For example, `opp_lem`, `add_lem1` and `add_lem2` are proved automatically with `field` in less than half a second.

4 Proving group law properties

Now that we have described the main tactic used to discard rational equations, we can sketch our proof of the properties of the group law. Two properties are very handy to shorten proofs. The first one is that, given two points of the curve $p_1 = (x_1, y_1)$ and $p_2 = (x_2, y_2)$, if we are capable of proving that $x_1 = x_2$ then it implies that $p_1 = p_2$ or $p_1 = -p_2$ (since $y_1^2 = y_2^2$). Note that often one of the two cases can actually be quickly discharged. The second property is that we can postpone the distinction between the tangent and the generic case. This is done by writing $p_3 = (x_3, y_3)$, the result of adding $p_1 = (x_1, y_1)$ and $p_2 = (x_2, y_2)$, as $x_3 = l^2 - x_1 - x_2$ and $y_3 = -y_1 - l(x_3 - x_1)$. Only the actual value of l differs, it is $(3x_1^2 + A)/2y_1$ in the tangent case and $(y_2 - y_1)/(x_2 - x_1)$ in the generic case. If the following, we will use the symbol $\oplus$ to denote this common case also called the non-degenerated case. More explicitly, we write $p_1 \oplus p_2$ to indicate $p_1 + p_2$ in the case where $p_1 \neq 0$, $p_2 \neq 0$, $p_1 \neq -p_2$. Also, we write $p_1 \oplus_g p_2$ for the addition in the generic case (we then add the extra assumption $p_1 \neq p_2$) and $p_1 \oplus_t p_2$ for the addition in the tangent case (we then add the extra assumption $p_1 = p_2$).

We can now start proving properties of the group law. First of all, 0 is a neutral element

Theorem *add0* : $\forall p,\ p + 0 = p$.

It is a consequence of the definition of the addition. Also, addition is commutative

Theorem *add_comm*: $\forall p_1\, p_2,\ p_1 + p_2 = p_2 + p_1$.

This is also straightforward since the definition of addition is symmetrical. The operation - acts as an opposite

Theorem *add_opp*: $\forall p,\ p + -p = 0$.

We are left with proving that addition is associative

Theorem *add_assoc*: $\forall p_1\, p_2\, p_3,\ p_1 + (p_2 + p_3) = (p_1 + p_2) + p_3$.

The structure of the proof is the following. First, we get three specific instances of the associativity by explicit computations. Then, we prove some simple properties like the cancellation rule (if $p_1 + p_2 = p_1 + p_3$ then $p_2 = p_3$). Finally, we show that we are able to cover all the cases of the general statement.

4.1 Specific instances

The first specific instance is the one where all the additions are done with the generic case

Theorem *spec*$_1$*_assoc*: $\forall p_1\, p_2\, p_3,\ p_1 \oplus_g (p_2 \oplus_g p_3) = (p_1 \oplus_g p_2) \oplus_g p_3$.

This is done by computation, taking the x component and the y component of both sides of the equation and showing that they are equal respectively. For the x component, this amounts in proving that

$$\begin{array}{lll} & x_1 - x_2 \neq 0 & \wedge \\ & x_4 - x_3 \neq 0 & \wedge \\ & x_2 - x_3 \neq 0 & \wedge \\ & x_5 - x_1 \neq 0 & \wedge \\ & y_1^2 = x_1^3 + Ax_1 + B & \wedge \\ & y_2^2 = x_2^3 + Ax_2 + B & \wedge \\ & y_3^2 = x_3^3 + Ax_3 + B & \wedge \\ & x_4 = (y_1 - y_2)^2/(x_1 - x_2)^2 - x_1 - x_2 & \wedge \\ & y_4 = -(y_1 - y_2)/(x_1 - x_2)(x_4 - x_1) - y_1 & \wedge \\ & x_6 = (y_4 - y_3)^2/(x_4 - x_3)^2 - x_4 - x_3 & \wedge \\ & y_6 = -(y_4 - y_3)/(x_4 - x_3)(x_6 - x_3) - y_3 & \wedge \\ & x_5 = (y_2 - y_3)^2/(x_2 - x_3)^2 - x_2 - x_3 & \wedge \\ & y_5 = -(y_2 - y_3)/(x_2 - x_3)(x_5 - x_2) - y2 & \wedge \\ & x_7 = (y_5 - y_1)^2/(x_5 - x_1)^2 - x_5 - x_1 & \wedge \\ & y_7 = -(y_5 - y_1)/(x_5 - x_1)(x_7 - x_1) - y_1 & \\ \Rightarrow & x_6 - x_7 = 0 & \end{array}$$

with the convention that $(x_4, y_4) = p_1 \oplus_g p_2$, $(x_5, y_5) = p_2 \oplus_g p_3$, $(x_6, y_6) = (p_1 \oplus_g p_2) \oplus_g p_3$ and $(x_7, y_7) = p_1 \oplus_g (p_2 \oplus_g p_3)$. The `field` tactic proves this goal automatically in 1.1 second. For the y component, it proves it automatically in 9.2 seconds.

Theorem *spec*$_2$*_assoc*: $\forall p_1\, p_2,\ p_1 \oplus_g (p_2 \oplus_t p_2) = (p_1 \oplus_g p_2) \oplus_g p_2$.

This is proved automatically by the `field` tactic in 3.9 seconds.

Theorem *spec*$_3$*_assoc*: $\forall p_1,\ p_1 \oplus_g (p_1 \oplus_g (p_1 \oplus_t p_1)) = (p_1 \oplus_t p_1) \oplus_t (p_1 \oplus_t p_1)$.

This is proved automatically by the `field` tactic in 13.8 seconds.

4.2 Basic Properties

The first basic property is that the opposite is unique

Theorem *uniq_opp*: $\forall p_1\, p_2,\ p_1 + p_2 = 0 \Rightarrow p_2 = -p1$.

If we look at the definition of $+$, there are only two cases when an addition can output a zero: if p_1 and p_2 are both zero or if $p_1 = -p_2$. So the theorem holds.
The zero is also unique.

Theorem *uniq_zero*: $\forall p_1\, p_2,\ p_1 + p_2 = p_2 \Rightarrow p_1 = 0$.

The proof is a bit more intricate, we need to examine all 5 possible cases for $p_1 + p_2$

1. If $p_1 = 0$, we have $p_1 = 0$

2. If $p_2 = 0$, we have $p_1 + p_2 = p_1 + 0 = 0$, so $p_1 = 0$.

3. If $p_1 = -p_2$, we have $p_1 + p_2 = p_1 - p_1 = 0 = -p_1$, so $p_1 = 0$.

4. If $p_1 = (x, y) = p_2$ with $y \neq 0$, we have $p_1 \oplus_t p_1 = p_1$,
taking the y component on both side, we get $-y - l(x - x) = y$, so $2y = 0$. As the characteristic is not 2, this contradict the fact that $y \neq 0$. This case is impossible.

5. If $p_1 = (x_1, y_1)$ and $p_2 = (x_2, y_2)$ with $x_1 \neq x_2$, we have $p_1 \oplus_g p_2 = p_2$.
Taking the y component on both sides, we get $-y_1 - l(x_2 - x_1) = y_2$ with $l = (y_2 - y_1)/(x_2 - x_1)$. We have $l(x_2 - x_1) = y_2 - y_1$, so $y_2 = -y_1 - l(x_2 - x_1) = -y_1 - (y_2 - y_1) = -y_2$, so $y_2 = 0$. As $p_2 = (x_2, 0)$ is on the curve, we have $x_2^3 = -(Ax_2 + B)$.
Taking the x component of $p_1 \oplus_g p_2 = p_2$, we get $l^2 - x_1 - 2x_2 = 0$. In particular, we have
$$x_2(l^2 - x_1 - 2x_2) = 0$$
Simplifying this rational expression with $l = (y_2 - y_1)/(x_2 - x_1)$, $y_2 = 0$ and $x_2^3 = -(Ax_2 + B)$, we get
$$(x_2 - x_1)(2Ax_2 + 3B) = 0$$
As we know that $x_2 \neq x_1$, we have $2Ax_2 + 3B = 0$.
We have $x_2^3 + Ax_2 + B = 0$ so in particular, we have $(2A)^3(x_2^3 + Ax_2 + B) = 0$. Simplifying this polynomial with $2Ax_2 + 3B = 0$ we get $B(4A^3 + 27B^2) = 0$.
As we have supposed that $4A^3 + 27B^2 \neq 0$, this implies that $B = 0$.
We know that $2Ax_2 + 3B = 0$, this implies that either A or x_2 is zero.
A cannot be zero, it would violate the assumption $4A^3 + 27B^2 \neq 0$, so we have $x_2 = 0$.
Simplifying
$$(l^2 - x_1 - 2x_2) = 0$$
with $l = (y_2 - y_1)/(x_2 - x_1)$, $y_1^2 = x_1^3 + Ax + B$, $y_2 = 0$, $x_2 = 0$ and $B = 0$ leads to
$$Ax_1 = 0$$
We already know that A cannot be zero and if $x_1 = 0$, as we know already that $x_2 = 0$, it would violate the assumption $x_1 \neq x_2$. This case is then impossible.

We then need to prove several properties of the opposite.

Theorem *opp_add*: $\forall p_1\, p_2,\ -(p_1 + p_2) = (-p_1) + (-p_2)$.

This follows from the definitions of addition and opposite.

Theorem *compat_opp_add*: $\forall p_1\, p_2,\ p_1 + p_2 = p_1 - p_2 \wedge p_1 \neq -p_1 \Rightarrow p_2 = -p_2$.

Again we enumerate all the cases for $p_1 + p_2$.

1. If $p_1 = 0$, this contradict the assumption $p_1 \neq -p_1$.

2. If $p_2 = 0$, we have $p_2 = -p_2$.

3. If $p_1 = -p_2$, we have $p_1 + p_2 = p_1 - p_1 = 0 = p_1 - p_2 = p_1 + p_1$, so $p_1 + p_1 = 0$. By the uniqueness of the opposite this contradict the fact that $p_1 \neq -p_1$.
4. If $p_1 = p_2$, we have $p_1 + p_2 = p_1 - p_2 = p_1 - p_1 = 0$. By the uniqueness of the opposite, we get $p_1 = -p_2$ which contradicts the fact that $p_1 \neq -p_1$.
5. If $p_1 = (x_1, y_1)$ and $p_2 = (x_2, y_2)$ with $x_1 \neq x_2$, taking the x component on both side of $p_1+p_2 = p_1-p_2$, we get $(y_2-y_1)^2/(x_2-x_1)^2-x_1-x_2 = (-y_2-y_1)^2/(x_2-x_1)^2-x_1-x_2$. So we have $4y_1y_2 = 0$. The characteristic is different from 2, so $4 \neq 0$, $p_1 \neq -p_1$ so $y_1 \neq 0$. This means that $y_2 = 0$, so $p_2 = -p_2$.

Theorem *compat_add_triple*: $\forall p, p \neq -p \wedge p + p \neq -p \Rightarrow (p + p) - p = p$.

We enumerate all the cases for $(p + p) + (-p)$.

1. If $p+p = 0$, by the uniqueness of the opposite this contradicts the assumption $p \neq -p$.
2. If $-p = 0$, this contradicts the assumption $p \neq -p$.
3. If $p + p = p$, by the uniqueness of the zero, we have $p = 0$ which contradicts $p \neq -p$.
4. If $p + p = -p$, this contradicts the assumption $p + p \neq -p$.
5. If $p = (x_1, y_1)$ and $p+p = (x_2, y_2)$ with $x_1 \neq x_2$ where $l = (3x_1^2+A)/2y_1$, $x_2 = l^2-2x_1$ and $y_2 = -y_1 - l(x_2 - x_1)$. Taking the x component of $(p + p) - p$ we get

$$((-y_1 - (-y_1 - l(l^2 - 2x_1 - x_1)))/(x_1 - (l^2 - 2x_1)))^2 - (l^2 - 2x_1) - x_1$$

which simplifies to x_1, so we have $(p+p)-p = p$ or $(p+p)-p = -p$. The second one is impossible, because of the uniqueness of the zero it would lead to $p + p = 0$. So we have $(p + p) - p = p$.

Theorem *add_opp_double_opp*: $\forall p_1\, p_2,\ p_1 + p_2 = -p_1 \Rightarrow p_2 = (-p_1) + (-p_1)$.

- If $p_1 = -p_1$, we have $-p_1 + p_2 = p_1 + p_2 = -p_1$, so by the uniqueness of the zero, we deduce $p_2 = 0$. So, we have $p_2 = 0 = p_1 + (-p_1) = (-p_1) + (-p_1)$.
- If $p_1 \neq -p_1$, we examine all the cases for $p_1 + p_2$.
  1. If $p_1 = 0$, this contradict $p_1 \neq -p_1$.
  2. If $p_2 = 0$, then we have $p_1 = p_1 + p_2 = -p_1$, so $p_1 = 0$ which we have shown in the previous case is impossible.
  3. If $p_1 = -p_2$, we have $0 = p_1 + p_2 = -p_1$, which is impossible.
  4. If $p_1 = p_2$, we have $p_1 + p_1 = p_1 + p_2 = -p_1 = -p_2$, so $p_2 = -(p_1 + p_1) = (-p_1) + (-p_1)$ by *opp_add*.

5. If $p_1 = (x_1, y_1)$ and $p_2 = (x_2, y_2)$ with $x_1 \neq x_2$, we have $-p_1 = p_1 + p_2$. Taking the x component on each side we get

$$x_1 = (y_2 - y_1)^2/(x_2 - x_1)^2 - x_1 - x_2$$

Simplifying it with $E_1 : y_1^2 = x_1^3 + Ax_1 + B$ and $E_2 : y_2^2 = x_2^3 + Ax_2 + B$ we get

$$2y_2y_1 = Ax_2 + 3x_2x_1^2 + Ax_1 - x_1^3 + 2B$$

Squaring this equality and simplifying by E_1 and E_2 we get

$$-x_1^6 + 6x_1^5x_2 + 2Ax_1^4 - 9x_1^4x_2^2 + 8Bx_1^3 + 4x_1^3x_2^3 - A^2x_1^2 - 6Ax_1^2x_2^2-$$

$$12Bx_1^2x_2 + 2A^2x_1x_2 + 4Ax_1x_2^3 - A^2x_2^2 + 4Bx_2^3 = 0$$

which is the same equation that the one we get when simplifying

$$(x_2 - (((3x_1^2 + A)/(2(-y_1)))^2 - 2x_1))(x_2 - x_1)^2 = 0.$$

by E_1 and E_2. As we know that $x_1 \neq x_2$, we can deduce that

$$x_2 = ((3x_1^2 + A)/(2(-y_1)))^2 - 2x_1.$$

But this indicates that the x component of p_2 is equal to the x component of $(-p_1)+(-p_1)$. So we have $p_2 = (-p_1)+(-p_1)$ or $p_2 = -((-p_1)+(-p_1))$. Suppose that $p_2 = -((-p_1) + (-p_1))$. Applying *opp_add*, gives that $p_2 = p_1 + p_1$.

- If $p_1 + p_1 = -p_1$, we have $p_2 = p_1 + p_1 = -p_1 = p_1 + p_2 = p_1 - p_1 = 0$ which is impossible because $p_1 \neq -p_1$.
- If $p_1 + p_1 \neq -p_1$, we have $p_1 - p_2 = p_1 - (p_1 + p_1) = -((p_1 + p_1) - p_1)$. Apply *comp_add_triple*, we get that $p_1 - p_2 = -p_1 = p_1 + p_2$. Using *compat_add_opp* this means that $p_2 = -p_2$. So $p_2 = -p_2 = -(p_1 + p_1) = (-p_1) + (-p_1)$.

Theorem *cancel*: $\forall p_1\, p_2\, p_3,\ p_1 + p_2 = p_1 + p_3 \Rightarrow p_2 = p_3$.

We first enumerate all the possible cases so to concentrate on the case for which $p_1 \oplus p_2$ and $p_1 \oplus p_3$ (case 6).

1. If $p_1 = 0$, then we have $p_2 = 0 + p_2 = 0 + p_3 = p_3$.
2. If $p_2 = 0$, then we have $p_1 = p_1 + 0 = p_1 + p_2 = p_1 + p_3$, so
3. If $p_3 = 0$, then we have $p_1 = p_1 + 0 = p_1 + p_3 = p_1 + p_2$, so by the uniqueness of the zero we have $p_2 = 0 = p_3$.
4. If $p_1 = -p_2$, we have $p_1 + p_3 = p_1 + p_2 = p_1 + (-p_1) = 0$, so by the uniqueness of the opposite we have $p_3 = -p_1 = p_2$.

5. If $p_1 = -p_3$, we have $p_1 + p_2 = p_1 + p_3 = p_1 + (-p_1) = 0$, so by the uniqueness of the opposite we have $p_2 = -p_1 = p_3$.

6. If $p_1 = (x_1, y_1)$ and $p_2 = (x_2, y_2)$, $p_3 = (x_3, y_3)$ and $(x_4, y_4) = p_1 \oplus p_2$, taking the y component of $p_1 \oplus p_2 = p_1 \oplus p_3$ we have

$$-y_1 - l_{12}(x_4 - x_1) = -y_1 - l_{13}(x_4 - x_1)$$

so we have $(l_{13} - l_{12})(x_4 - x_1) = 0$. We have two cases.

   - If $l_{13} = l_{12}$, taking the x component of $p_1 \oplus p_2 = p_1 \oplus p_3$ we have $l_{12}^2 - x_1 - x_3 = l_{13}^2 - x_1 - x_2$, so $x_2 = x_3$. So we have $p_2 = p_3$ or $p_2 = -p_3$. We are then left with the case $p_2 = -p_3$. If $p_2 = -p_3$, we have $p_1 + p_2 = p_1 + p_3 = p_1 - p_2$. Applying *compat_opp_add* gives us that $p_2 = -p_2$ as we already know that $p_2 = -p_3$, we get $p_2 = p_3$. The last application of the theorem *compat_opp_add* requires the side condition $p_1 \neq -p_1$. If we have $p_1 = -p_1$, as we have already $p_1 \neq -p_2$ and $p_1 \neq -p_3$, we get $p_1 \neq p_2$ and $p_1 \neq p_3$, so we have $p_1 \oplus_g p_2$ and $p_1 \oplus_g p_3$. So we can rewrite $l_{12} = l_{13}$ as

   $$(y_1 - y_2)/(x_1 - x_2) = (y_1 - y_3)/(x_1 - x_3)$$

   As we have $p_2 = -p_3$, we know that $x_2 = x_3$ which gives that $y_2 = y_3$ so we have $y_2 = 0$ and $p_2 = p_3$.
   - If $x_4 = x_1$, this means that either $p_1 + p_2 = p_1$ or $p_1 + p_2 = -p_1$
     - If $p_1 + p_2 = p_1 + p_3 = p_1$, by the uniqueness of the zero we have $p_2 = p_3 = 0$,
     - If $p_1 + p_2 = p_1 + p_3 = -p_1$ using twice *add_opp_double_opp* we get $p_2 = (-p_1) + (-p_1) = p_3$.

Theorem *add_minus_id*: $\forall p_1\, p_2,\ (p_1 + p_2) - p_2 = p_1$.

- If $p_1 + p_2 = -p_2$, we have $p_1 = (-p_2) + (-p_2)$ by theorem *add_opp_double_opp*. So we have $(p_1 + p_2) - p_2 = (-p_2) + (-p_2) = p_1$.
- If $p_1 + p_2 \neq -p_2$,
  - if $p_1 = 0$, then $(p_1 + p_2) - p_2 = p_2 - p_2 = 0 = p_1$.
  - if $p_2 = 0$, then $(p_1 + p_2) - p_2 = (p_1 + 0) - 0 = p_1$.
  - if $p_1 = -p_2$, then $(p_1 + p_2) - p_2 = (p_1 - p_1) - p_2 = -p_2 = p_1$.
  - if $p_1 = p_2$, then $(p_1 + p_2) - p_2 = (p_1 + p_1) - p_1 = p_1$ by theorem *add_minus_id*.
  - if $p_2 = p_1 + p_2$, then $p_1 = 0$ by the uniqueness of zero.
  - So we are left with proving $(p_1 \oplus_g p_2) \oplus_g (-p_2) = p_1$. This is done by computation. The `field` tactic proves it automatically in 1 second.

Theorem *add_shift_minus*: $\forall p_1\, p_2\, p_3,\ p_1 + p_2 = p_3 \Rightarrow p_1 = p_3 - p_2$.

We have $p_3 = (p_3 - p_2) + p_2$ by *add_minus_id*. Applying *cancel* to $p_1 + p_2 = (p_3 - p_2) + p_2$, we get $p_1 = p_3 - p_2$.
We are now proving the associativity for the degenerated cases.

Theorem *degen_assoc*: $\forall p_1\, p_2\, p_3$,
$(p_1 = 0 \vee p_2 = 0 \vee p_3 = 0) \vee (p_1 = -p_2 \vee p_2 = -p_3) \vee (-p_1 = p_2 + p_3 \vee -p_3 = p_1 + p_2) \Rightarrow$
$p_1 + (p_2 + p_3) = (p_1 + p_2) + p_3$.

We simply enumerate the cases:

- if $p_1 = 0$, then $p_1 + (p_2 + p_3) = p_2 + p_3 = (0 + p_2) + p_3 = (p_1 + p_2) + p_3$.
- if $p_2 = 0$, then $p_1 + (p_2 + p_3) = p_1 + p_3 = (p_1 + 0) + p_3 = (p_1 + p_2) + p_3$.
- if $p_3 = 0$, then $p_1 + (p_2 + p_3) = p_1 + p_2 = (p_1 + p_2) + 0 = (p_1 + p_2) + p_3$.
- if $p_1 = -p_2$, then by theorem *add_minus_id* we have $(p_3 + p_2) - p_2 = p_3$, so $(p_1 + p_2) + p_3 = p_3 = (p_3 + p_2) - p_2 = p_1 + (p_2 + p_3)$.
- if $p_2 = -p_3$, then by theorem *add_minus_id* we have $(p_1 + p_2) - p_2 = p_1$, so $p_1 + (p_2 + p_3) = p_1 = (p_1 + p_2) - p_2 = (p_1 + p_2) + p_3$.
- if $p_1 + p_2 = -p_3$, then by theorem *add_minus_id* we have $p_2 - (p_1 + p_2) = ((-p_1) + (-p_2)) - (-p_2) = -p_1$, so
  $p_1 + (p_2 + p_3) = p_1 + (p_2 - (p_1 + p_2)) = p_1 + -p_1 = 0 = (p_1 + p_2) - (p_1 + p_2) = (p_1 + p_2) + p_3$.
- if $p_2 + p_3 = -p_1$, then by theorem *add_minus_id* we have $-(p_2 + p_3) + p_2 = ((-p_3) + (-p_2)) - (-p_2) = -p_3$, so
  $p_1 + (p_2 + p_3) = 0 = -p_3 + p_3 = (-(p_2 + p_3) + p_2)) + p_3 = (p_1 + p_2) + p_3$.

The last step before proving the associativity is to prove it when there is at least one tangent operation.

Theorem *spec$_4$_assoc*: $\forall p_1\, p_2,\ p_1 + (p_2 + p_2) = (p_1 + p_2) + p_2$.

The previous theorem tells us that we can restrict ourself to $p_1 \oplus (p_2 \oplus_t p_2) = (p_1 \oplus p_2) \oplus p_2$

- if $p_1 = p_2$, then we have $p_1 \oplus (p_1 \oplus_t p_1) = (p_1 \oplus_t p_1) \oplus p_1$ by the commutativity of the addition.
- if $p_1 = p_2 \oplus_t p_2$, we have to prove

  $$(p_2 \oplus_t p_2) \oplus_t (p_2 \oplus_t p_2) = ((p_2 \oplus_t p_2) \oplus p_2) \oplus p_2$$

  We have $p_2 \neq p_2 \oplus_t p_2$, otherwise $p_2 = 0$ by the uniqueness of the zero and this would contradict $p_2 \oplus_t p_2$. We also have $p_2 \neq (p_2 \oplus_t p_2) \oplus p_2$, otherwise $p_2 \oplus p_2 = 0$ by the theorem *cancel* which is impossible. So we have to prove

  $$(p_2 \oplus_t p_2) \oplus_t (p_2 \oplus_t p_2) = ((p_2 \oplus_t p_2) \oplus_g p_2) \oplus_g p_2$$

which is exactly the theorem *spec*$_3$*_ assoc*.

- if $p_2 = p_1 \oplus p_2$, we have $p_1 = 0$ by the uniqueness of zero. As $p_1 \oplus_t p_1$, this case is impossible

- Otherwise the remaining case is

$$p_1 \oplus_g (p_2 \oplus_t p_2) = (p_1 \oplus_g p_2) \oplus_g p_2$$

which is exactly the theorem *spec*$_2$*_ assoc*.

4.3 Associativity

We are now ready to prove the associativity

Theorem *add_ assoc*: $\forall p_1\, p_2\, p_3,\ p_1 + (p_2 + p_3) = (p_1 + p_2) + p_3$.

Applying the theorem *degen_ assoc*, we can restrict ourself to the non-degenerated cases

$$p_1 \oplus (p_2 \oplus p_3) = (p_1 \oplus p_2) \oplus p_3$$

- If $p_2 = p_3$, we have $p_1 \oplus (p_2 \oplus_t p_2) = (p_1 \oplus p_2) \oplus p_2$ by the theorem *spec*$_4$*_ assoc*.

- If $p_1 = p_2$, we have $p_1 \oplus (p_1 \oplus_t p_3) = (p_1 \oplus_t p_1) \oplus p_3$ by the theorem *spec*$_4$*_ assoc*.

- if $p_3 = (p_1 \oplus_g p_2)$, we have
  $((p_1 \oplus_g p_2) \oplus_t (p_1 \oplus_g p_2)) \oplus (-p_2)$
  $= (p_1 \oplus_g p_2) \oplus_g ((p_1 \oplus_g p_2) \oplus_g (-p_2))$ by *spec*$_4$*_ assoc*
  $= (p_1 \oplus_g p_2) \oplus_g p_1$ by *add_ minus_ id*
  $= (((p_1 \oplus_g p_2) \oplus_g p_1) \oplus p_2) + (-p_2)$ by *add_ minus_ id*.
  Applying *cancel*, we have $(p_1 \oplus_g p_2) \oplus_t (p_1 \oplus_g p_2) = ((p_1 \oplus_g p_2) \oplus_g p_1) \oplus_g p_2$.

- if $p_1 = (p_2 \oplus_g p_3)$, similarly we have
  $((p_2 \oplus_g p_3) \oplus_t (p_2 \oplus_g p_3)) \oplus (-p_3)$
  $= (p_2 \oplus_g p_3) \oplus_g ((p_2 \oplus_g p_3) \oplus_g (-p_3))$ by *spec*$_4$*_ assoc*
  $= (p_2 \oplus_g p_3) \oplus_g p_2$ by *add_ minus_ id*
  $= (((p_2 \oplus_g p_3) \oplus_g p_2) \oplus p_3) + (-p_3)$ by *add_ minus_ id*.
  Applying *cancel*, we have $(p_2 \oplus_g p_3) \oplus_t (p_2 \oplus_g p_3) = ((p_2 \oplus_g p_3) \oplus_g p_2) \oplus_g p_3$.

- Otherwise, we are left with proving $p_1 \oplus_g (p_2 \oplus_g p_3) = (p_1 \oplus_g p_2) \oplus_g p_3$ which is exactly the theorem *spec*$_1$*_ assoc*.

5 Conclusion

This work has a long story. Joe Hurd was the first to draw our attention to the possibility of formalising elliptic curves inside a prover [7]. At that time, we had a go but were quickly convinced that a prover like Coq, which requires a formal justification of every step of a proof, could not cope with the necessary computations. Last september, we emailed to John Harrison the example of the x component of the generic case and to our great surprise he managed to prove it in less than 3 minutes inside HolLight [5] with his integrated version of Bucherger algorithm [6]. So the situation was not as hopeless as we thought. Indeed, the proof presented here is checked by Coq, computations included, in 1 minute 20 seconds. This is a striking example of how crucial it is in a prover to be able to mix proof and computation.

This proof is really tedious. We believe that actually translating the proof inside an interactive prover is the best way to assess its full correctness. In particular, most theorems presented here contain side-conditions. The prover makes sure that we do not forget to prove any of these. We have tried to keep our proof as elementary as possible. No elaborate technique is used to perform computation just a standard normaliser and a naive rewriter. Also thanks to Guillaume Hanrot's help, we manage to restrict the justifications in the proof to the basic properties of ring polynomials. For example, in the original paper, a proof was using the fact that a polynomial of degree 3 has at most 3 roots, we have found an alternative justification.

Finally, given its elementary nature, this proof is a good candidate for benchmarking proof systems. Also, having done elliptic curves is a first step towards further formalisations. The application of elliptic curves to primality is our next goal.

References

[1] Samuel Boutin. Using Reflection to Build Efficient and Certified Decision Procedures. In *TACS'97*, volume 1281 of *LNCS*, pages 515–529, Sendai, Japan, 1997.

[2] Antonio Capani, Gianfranco Niesi, and Lorenzo Robbiano. Cocoa: Computations in commutative algebra. Available at `http://cocoa.dima.unige.it/`.

[3] Stefan Friedl. An elementary proof of the group law for elliptic curves. extracted from undergraduate thesis, Regensburg (1998), Available at `http://www.labmath.uqam.ca/~friedl/papers/AAELLIPTIC.PDF`.

[4] Benjamin Grégoire and Assia Mahboubi. Proving ring equalities done right in Coq. In *TPHOLS'05*, volume 3603 of *LNCS*, pages 98–113.

[5] John Harrison. HOL light: A tutorial introduction. In *FMCAD'96*, volume 1166, pages 265–269, 1996.

[6] John Harrison. Complex quantifier elimination in HOL. In *TPHOLs 2001: Supplemental Proceedings*, pages 159–174. University of Edinburgh, 2001. Available on the Web at `http://www.informatics.ed.ac.uk/publications/report/0046.html`.

[7] Joe Hurd. Formalized elliptic curve cryptography. Available at `http://www.cl.cam.ac.uk/~jeh1004/research/talks/elliptic-talk.pdf`.

Unité de recherche INRIA Sophia Antipolis
2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex (France)

Unité de recherche INRIA Futurs : Parc Club Orsay Université - ZAC des Vignes
4, rue Jacques Monod - 91893 ORSAY Cedex (France)
Unité de recherche INRIA Lorraine : LORIA, Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - BP 101 - 54602 Villers-lès-Nancy Cedex (France)
Unité de recherche INRIA Rennes : IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex (France)
Unité de recherche INRIA Rhône-Alpes : 655, avenue de l'Europe - 38334 Montbonnot Saint-Ismier (France)
Unité de recherche INRIA Rocquencourt : Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex (France)

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-0803