The Group Law for Edwards Curves

Rodrigo Raya Castellano

Technical University of Munich

Contents

1 Introduction

The development of formal proofs of the group law of elliptic curves is very old. However, most of the works we are aware of focus on the group law for elliptic curves in Weierstrass form. Proofs varied largely on the theoretical complexity of the arguments used. For instance, in [3], the authors develop a formal library for elliptic curves and the group law is shown establishing an isomorphism between the elliptic curve and its Picard group of divisors. Simpler arguments exist. For instance, [18] provides a more practical approach based on computationally intensive reasoning techniques rather than in sophisticated mathematical concepts. Here the author already points out one of the main challenges that elliptic curve verification presents to a modern theorem prover:

> 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. The main difficulty has been to find an effective way to cope with these computations inside our proof system.

The author of this report conducted a similar case study two years ago [15] following closely [16]. This experience made him realise that the problem of verifying elliptic curves was not trivial at all and in fact, constitutes an ideal experiment to test the health of a theorem prover since it is both simple to understand and difficult to implement. Far are the times in which some textbooks could summarise the question as follows [17]:

> Of course, there are an awful lot of special cases to consider, such as when one of the points is the negative of the other or two of the points coincide. But in a few days you will be able to check associativity using these formulas. So we need say nothing more about the proof of the associative law!

There is already an entry on the Archive of Formal Proofs that formalizes elliptic curves in Weierstrass form [4]. Nevertheless, there is much less work done on elliptic curves in Edwards form. This can be explained by their relative novelty as they were first proposed in 2007 by Harold Edwards [9].

Despite this, we find a similar situation to that encountered with Weierstrass curves. On the one hand, there exist proofs whose mathematical content is quite deep. For instance, Edwards himself proposed an approach based on differential equations in connection with previous work by Euler and Gauss. Another approach is to build a so called rational equivalence between curves in Edwards form and curves in Weierstrass form [8] and then transfer the group properties from one to the other. A more practical approach has more recently been proposed and partially mechanized by Hales [9]. In the present work, this last approach was followed.

2 Edwards curves

2.1 Affine Edwards curves

Let k be an arbitrary field (for simplicity we assumed $k = \mathbb{R}$ in our proof). We define an Edward curve E as the set of zeros of the following multivariate polynomial:

$$e(x, y) = x^2 + cy^2 - 1 - dx^2y^2$$

where $c, d, x, y \in k$ and c, d are fixed. Then one defines the addition of two points on the curve as:

$$z_1 \oplus z_2 = (x_1, y_1) \oplus (x_2, y_2) = \Big(\frac{x_1x_2 - cy_1y_2}{1 - dx_1x_2y_1y_1}, \frac{x_1y_2 + y_1y_2}{1 + dx_1x_2y_1y_2}\Big)$$

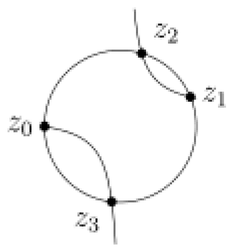
The motivation for this addition is explained in [10] in a very accessible manner without going into the details of algebraic geometry. Essentially, one reinterprets the normal complex multiplication:

$$(x_1, y_1) \cdot (x_2, y_2) = (x_1x_2 - y_1y_2, x_1y_2 + x_2y_1)$$

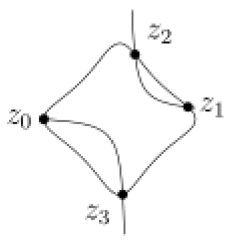
by considering the family of hyperbolas in the plane that pass through $z = (-1, 0)$ and whose asymptotes are parallel to the coordinate axes, these are given by the equation:

$$xy + p(x + 1) + qy = 0$$

Every two points z_1 and z_2 on the circle intersect some hyperbola within the family. The hyperbola meets the unit circle in one additional point z_3:



It turns out that $z_1z_2z_3 = 1$ and one may define complex multiplication as $z_1z_2 = \overline{z_3}$. The very same construction works for general elliptic curves and gives the addition formula above:



The goal of [10] is to show that $(E, \oplus)$ is an abelian group. This is only feasible for affine Edwards curves when d is not a non-null square.

2.2 Projective Edwards curves

Introduced to bypass the restriction that d should not be a non-null square.

Introduced by gluing two affine curves E_{aff} together.

The associative property for E is a direct consequence of the associative property of the affine pieces E_{aff}.

More formally, one introduces the set E_{aff} as the set of zeros of the equation:

$$e(x, y) = x^2 + y^2 - 1 - t^2x^2y^2$$

One also denotes by E_{circ} the subset E_{aff} whose coordinates x, y are non-zero. Finally, in the cartesian product $E_{\text{aff}} \times E_{\text{aff}}$ we use the notation (P, i) to refer to point P of E using the i-th copy of E_{aff} and we identify over the elements of E_{circ} the pair of points (P, i) and $(\tau P, i + 1)$. So elements of E are actually classes of points that we denote by $[P, i]$.

The addition on E requires first to define a second addition $\oplus_1$ (previous addition is denoted from now on by $\oplus_0$) given by:

$$z_1 \oplus_1 z_2 = (x_1, y_1) \oplus_1 (x_2, y_2) = \left(\frac{x_1y_1 - x_2y_2}{x_2y_1 - x_1y_2}, \frac{x_1y_1 + x_2y_2}{x_1x_2 + y_1y_2}\right)$$

Finally, addition on E is defined by:

$$[P, i] \oplus [Q, j] = [P \oplus_l Q, i + j]$$

if $\delta_l(P, Q) \neq 0, l \in \mathbb{F}_2 = \{0, 1\}$ where δ_l is the magnitude ensuring that the quotients are non-zero. Here, one still needs to prove that $\oplus$ is well-defined. Well-definition and an auxiliary lemma called dichotomy are actually the hardest properties to be established in the development.

2.3 Use of Edwards curves

Edwards curves present several advantages compared to traditional Weierstrass form. To start with, they lead to an explicit formula for addition while in Weierstrass curves addition needs to take care of the special *infinity point*. Also, Bernstein and Lange [5] showed that addition and multiplication by scalars could be done more efficiently over these curves. Another strong property noted in this study is that if d is not a square the addition was defined for all input pairs on the curve. This is the so-called completeness property of addition. Essentially, it tells that the Edwards addition law can carry out any sequence of group operations, without risk of failure [5].

Furthermore, there are important issues arising from the implementation of elliptic curves that should be kept in mind. Literature has given special attention to the so-called side-channel attacks. These cryptographic attacks are based in the measurement of the physical parameters of the system. For instance, one could measure the power consumption [12] or the time spent in executing the underlying algorithm [11] in order to gain information on the cryptographic parameters used. Edwards curves seem to be useful to prevent this kind of attacks [5].

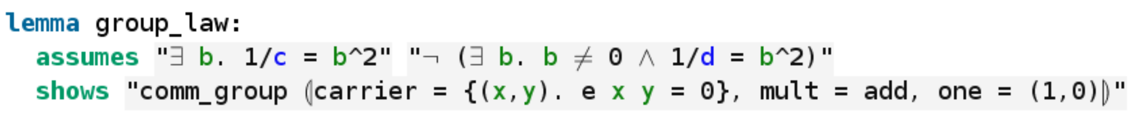
As for projective Edwards curves, which are known in the literature as twisted Edwards curves, they provide a generalization of affine Edwards curves covering a wider range of algebraic curves (in particular, the important class of Montgomery curves) which leads to faster arithmetic computations. The advantage of formalizing this curves lies in the fact that we do not special conditions on the defining parameters to get a group law with their addition [6].

All of these properties made Edwards curves and their extensions to be of interest in the field of elliptic curve cryptography.

3 Isabelle tools used

3.1 The use of locales

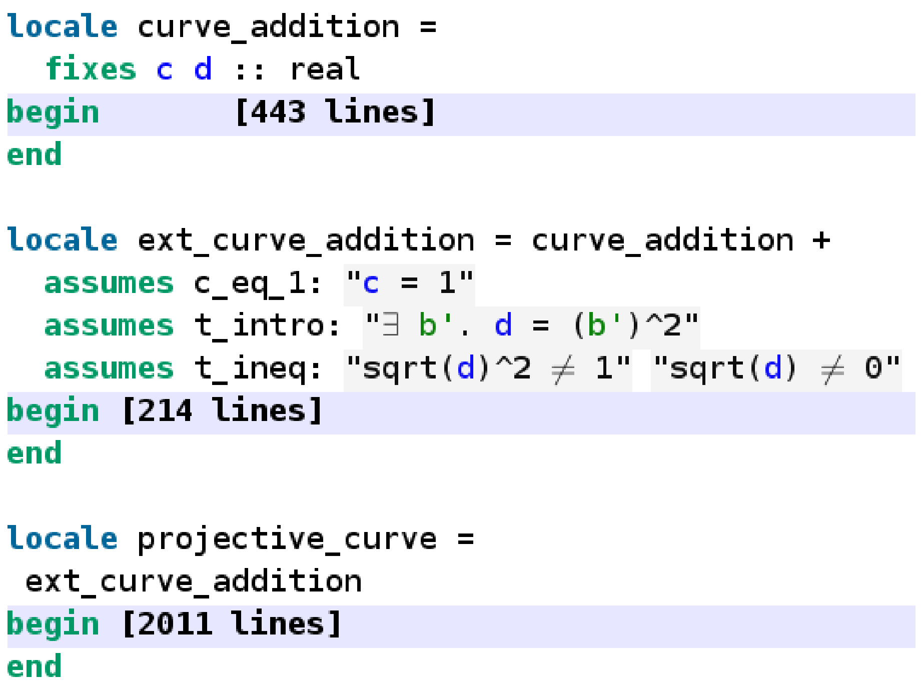
During this project, I had a first contact with the tools that Isabelle uses to manage theories. These are not normally introduced in a basic course on the proof assistant and we had to explore on our own using the existing documentation [2]. We worked both with external locales (such as comm_group):



```
lemma group_law:
  assumes "∃ b. 1/c = b^2" "¬ (∃ b. b ≠ 0 ∧ 1/d = b^2)"
  shows "comm_group ⦇carrier = {(x,y). e x y = 0}, mult = add, one = (1,0)⦈"
```

Figure 1: The group law theorem for affine elliptic curves using comm_group locale

and our own locales that served to structure and particularize the theories:



```
locale curve_addition =
  fixes c d :: real
begin       [443 lines]
end

locale ext_curve_addition = curve_addition +
  assumes c_eq_1: "c = 1"
  assumes t_intro: "∃ b'. d = (b')^2"
  assumes t_ineq: "sqrt(d)^2 ≠ 1" "sqrt(d) ≠ 0"
begin [214 lines]
end

locale projective_curve =
 ext_curve_addition
begin [2011 lines]
end
```

Figure 2: General structure of the theory organized in locales

3.2 The algebra method

A good understanding of the proving tools at our disposal can save a lot of work. The algebra method [19] has as one of its basic functionalities solving the following quantified formula:

$$
\begin{aligned}
&\forall x_1 \dots x_n. \\
&\qquad e_1(x_1,\dots,x_n) = 0 \wedge \dots \wedge e_m(x_1,\dots,x_n) = 0 \rightarrow \\
&\qquad (\exists y_1,\dots,y_k. \\
&\qquad p_{11}(x_1,\dots,x_n)y_1 + \dots + p_{1k}(x_1,\dots,x_n)y_k = 0 \wedge \\
&\qquad \dots \wedge \\
&\qquad p_{t1}(x_1,\dots,x_n)y_1 + \dots + p_{tk}(x_1,\dots,x_n)y_k = 0)
\end{aligned}
$$

where $e_1,\dots,e_n$ and p_{ij} are multivariate polynomials in the indicated variables. In our case, e_i could be the hypothesis that we have certain point in the curve and the exists fragment corresponds to the search of division quotients. As a direct consequence, one does not always need to copy the polynomial quotients obtained with Mathematica. Here is an example:

```
have gx_div: "∃ r1 r2 r3. gxpoly_expr = r1 * e1 + r2 * e2 + r3 * e3"
  unfolding gxpoly_expr_def e1_def e2_def e3_def e_def
  by algebra

have gy_div: "∃ r1 r2 r3. gypoly_expr = r1 * e1 + r2 * e2 + r3 * e3"
  unfolding gypoly_expr_def e1_def e2_def e3_def e_def
  by algebra
```

Figure 3: Example of polynomial division with the algebra method

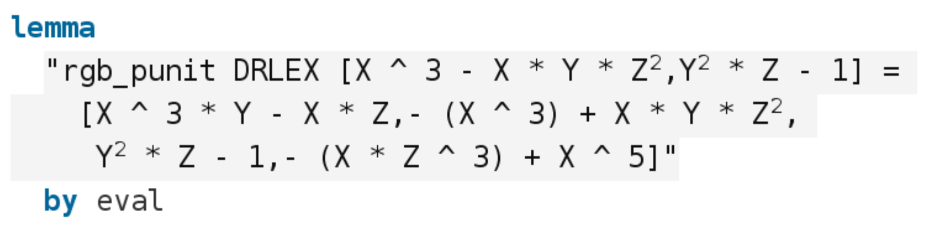
The variables gxpoly_expr, gypoly_expr correspond to given polynomials that are computed explicitly with Mathematica. However, in our case, there is no need to copy the expressions for $r1, r2, r3$ since all we need is their existence. This is in contrast to the situation found in [10]

3.3 The use of Gröbner basis

Hales explicitly uses Gröbner basis to go through the proof of dichotomy. This property is fundamental to establish the well-definition of the group law for projective Edwards curves. To be precise, it would be need in Isabelle whether a given set of polynomials corresponds to a Gröbner basis of a given polynomial ideal. However, the representation of this theory is quite different to ours:

Essentially, polynomials are represented as ordered lists of monomials, where monomials are represented as pairs consisting of a coefficient and a power-product (I.e., something like $x_0 * y_0$); power-products are represented similarly, as ordered lists of indeterminate-exponent-pairs. Indeterminates, finally, are typically represented by natural numbers (although this can be changed easily). See also [13].

While in practice the computation of Gröbner basis in Isabelle might look like in Figure 4.



```
lemma
  "rgb_punit DRLEX [X ^ 3 - X * Y * Z², Y² * Z - 1] =
    [X ^ 3 * Y - X * Z,- (X ^ 3) + X * Y * Z²,
     Y² * Z - 1,- (X * Z ^ 3) + X ^ 5]"
  by eval
```

Figure 4: Gröbner basis computation in Reduced_GB_Examples.thy

more expertise on this theory would be needed in order to connect it with our own. This can be considered as future work since having at our disposal an easy access to verified Gröbner basis computation would have speeded the proof process significantly.

On the other hand, the algebra method is not capable to solve this problem either. While it computes Gröbner basis for the purpose of proving whether a polynomial is in the generated ideal, this computation is done outside the logic. In particular, it is not proved that the computed basis is indeed a Gröbner basis.

As a last alternative, one could check by hand that the involved identities hold. This is what Hales suggests in the article when he says: *In particular, our approach does not require the use of Gröbner bases (except in Lemma 4.3.2 where they make an easily avoidable appearance).*

Here is one example of such a hand computation. We follow the notation in [10]. Namely, we focus on equation 19:

$$(x_0^2 - x_1^2, y_0^2 - x_1^2, x_0y_0 - x_1y_1) \equiv (0,0,0)\ mod\ S_\pm$$

where $S_\pm$ is the Gröbner basis associated with certain polynomials known to evaluate to zero. One then deduces one by one the corresponding equations:

To start, one deduces the third equality:

$$\delta' = x_0y_0\delta_{0x}x_1y_1\left(\frac{1}{tx_0}\right)\left(\frac{1}{ty_0}\right) = x_0y_0\left(1 - \frac{t^2x_1y_1}{t^2x_0y_0}\right) = x_0y_0 - x_1y_1$$

Since by assumption $\delta_- = 0$ we have the third equality and we note:

$$(1)\ x_0 = x_1\left(\frac{y_1}{y_0}\right)$$

where we have that $x_0, y_0, x_1, y_1 \neq 0$.

For the second, we have:

$$\delta_+ = tx_0t_0\delta_{1x}x_1y_1\left(\frac{1}{tx_0}\right)\left(\frac{1}{ty_0}\right) = tx_0y_0\left(\frac{y_1}{tx_0}-\frac{x_1}{ty_0}\right) = y_0y_1-x_0x_1 \overset{1}{=} y_0y_1-x_1^2\left(\frac{y_1}{y_0}\right) = \frac{y_1}{y_0}(y_0^2-x_1^2)$$

since $\delta', \delta_+ = 0$, we have the second equation. We also note:

$$(2)\ \frac{y_1}{y_0}(y_0^2 - x_1^2) = 0$$

Finally, the third equation is obtained as follows:

$$x_0^2 - y_1^2 \overset{1}{=} x_1^2\frac{y_1^2}{y_0^2} - y_1^2 = \left(\frac{y_1^2}{y_0^2}\right)(x_1^2 - y_0^2) \overset{2}{=} \frac{y_1^2}{y_0^2}(x_1^2 - y_0^2) \overset{2}{=} 0$$

One should note that not all the polynomials of $S_\pm$ were used in this deduction.

3.4 The representation of projective elliptic curves

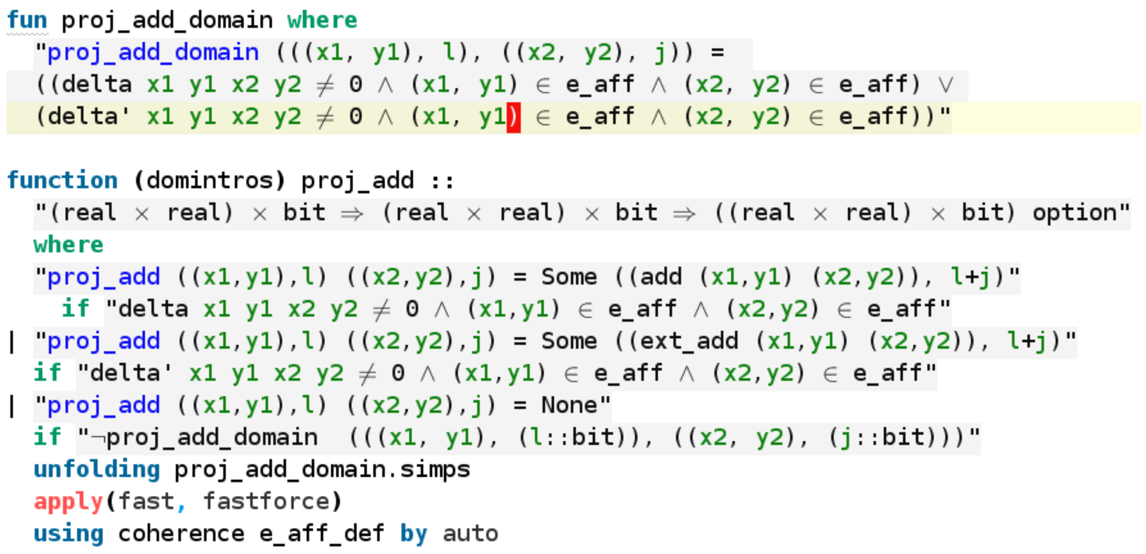
We have discussed the representation of projective Edwards curves as described in [10]. The challenge to represent this description is similar to the representation of equivalence classes in [14]. Here is, to start with, the definition of projective addition:

```
definition proj_add ::
  "(real × real) × bit ⇒ (real × real) × bit ⇒ ((real × real) × bit) option" where
  "
  proj_add p1 p2 =
    (
      if (p_delta p1 p2 ≠ 0 ∧ fst p1 ∈ e_aff ∧ fst p2 ∈ e_aff)
      then Some (add (fst p1) (fst p2), (snd p1) + (snd p2))
      else
        (
          if (p_delta' p1 p2 ≠ 0 ∧ fst p1 ∈ e_aff ∧ fst p2 ∈ e_aff)
          then Some (ext_add (fst p1) (fst p2), (snd p1) + (snd p2))
          else None
        )
    )
  "
```

Figure 5: Projective addition on points

A posteriori, one notes that it would be more convenient to have a more balanced definition without the if-else construct. The version in figure **??** would save some work. For instance, to show that we are in the second branch of the if, we would not need to show that the first branch has a false guard.

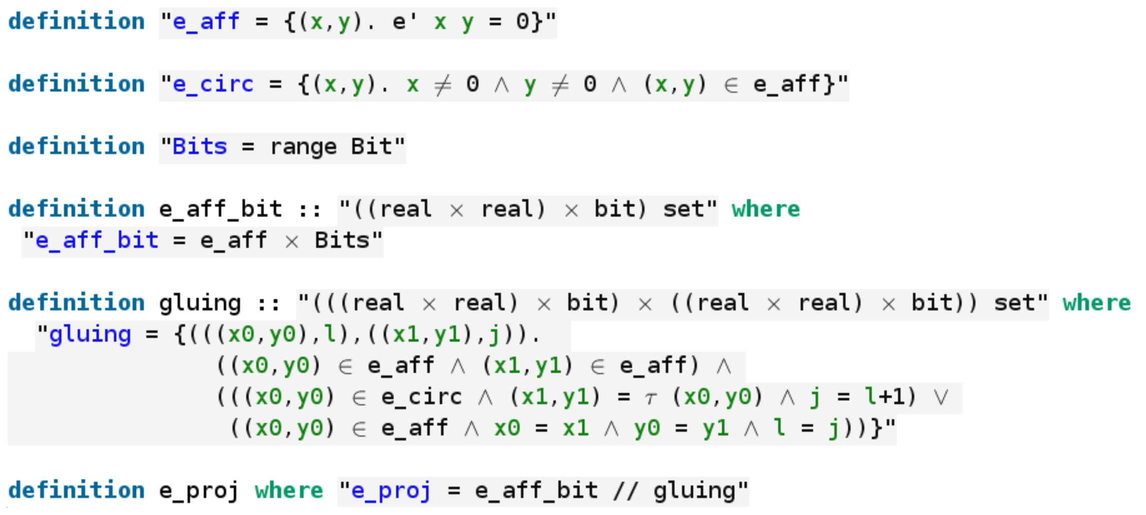
Before introducing proper projective addition, we introduce how are projective points represented:



```
fun proj_add_domain where
  "proj_add_domain (((x1, y1), l), ((x2, y2), j)) =
  ((delta x1 y1 x2 y2 ≠ 0 ∧ (x1, y1) ∈ e_aff ∧ (x2, y2) ∈ e_aff) ∨
  (delta' x1 y1 x2 y2 ≠ 0 ∧ (x1, y1) ∈ e_aff ∧ (x2, y2) ∈ e_aff))"

function (domintros) proj_add ::
  "(real × real) × bit ⇒ (real × real) × bit ⇒ ((real × real) × bit) option"
  where
  "proj_add ((x1,y1),l) ((x2,y2),j) = Some ((add (x1,y1) (x2,y2)), l+j)"
    if "delta x1 y1 x2 y2 ≠ 0 ∧ (x1,y1) ∈ e_aff ∧ (x2,y2) ∈ e_aff"
| "proj_add ((x1,y1),l) ((x2,y2),j) = Some ((ext_add (x1,y1) (x2,y2)), l+j)"
  if "delta' x1 y1 x2 y2 ≠ 0 ∧ (x1,y1) ∈ e_aff ∧ (x2,y2) ∈ e_aff"
| "proj_add ((x1,y1),l) ((x2,y2),j) = None"
  if "¬proj_add_domain  (((x1, y1), (l::bit)), ((x2, y2), (j::bit)))"
  unfolding proj_add_domain.simps
  apply(fast, fastforce)
  using coherence e_aff_def by auto
```

Figure 6: Balanced version of the projective addition on points



```
definition "e_aff = {(x,y). e' x y = 0}"

definition "e_circ = {(x,y). x ≠ 0 ∧ y ≠ 0 ∧ (x,y) ∈ e_aff}"

definition "Bits = range Bit"

definition e_aff_bit :: "((real × real) × bit) set" where
 "e_aff_bit = e_aff × Bits"

definition gluing :: "(((real × real) × bit) × ((real × real) × bit)) set" where
  "gluing = {(((x0,y0),l),((x1,y1),j)).
               ((x0,y0) ∈ e_aff ∧ (x1,y1) ∈ e_aff) ∧
               (((x0,y0) ∈ e_circ ∧ (x1,y1) = τ (x0,y0) ∧ j = l+1) ∨
                ((x0,y0) ∈ e_aff ∧ x0 = x1 ∧ y0 = y1 ∧ l = j))}"

definition e_proj where "e_proj = e_aff_bit // gluing"
```

Figure 7: Projective points representation

So points that satisfy the equation given by e' an which are non-zero are identified with their inversions modulo τ. The other conditions in the equivalence relation only ensure the reflexivity property. It is on these classes of points that projective addition should work. Figure 8 shows the adaption of proj_add to classes.

proj_add_class first selects those pairs that lead to some result, then it actually computes the result and finally identifies equivalent points. Much later, when we prove covering and well-definition we show that indeed the last identification provides only one class. This is the class that is selected with the_elem in proj_addition.

```
definition "proj_add_class c1 c2 =
  (((case_prod (λ x y. the (proj_add x y))) `
    (Map.dom (case_prod proj_add) ∩ (c1 × c2)))
    // gluing)"

definition "proj_addition c1 c2 = the_elem(proj_add_class c1 c2)"
```

Figure 8: Projective addition on classes of points

4 Notes on the proof

The proof is maintained on Github at this repository.

Once loaded in the editor, there are three sections. The first section formalises the plain affine elliptic curve group law. Second and third sections are dedicated to the projective group addition. The second section formalises the main preliminaries and the third formalises the group law properties of the addition for projective Edwards curves. The theorems and lemmas follow closely the nomenclature from [10] and in a definitive version the corresponding numbering will be added to make it clearer.

The current state of the proof is as follows: we formalised the paper until page 14 where we had to stop since we could not compute the required Gröbner basis. Instead, some of the hand computations that were described in previous sections were implemented, but there remain some to be completed. This would essentially finish dichotomy. Lemmas 4.3.3 and 4.3.4 were also fully formalised. Finally, from theorem 4.4.1, which establishes the group law of projective Edwards curves, it would remain to finish the proofs of associativity and closure. These follow similar techniques to the ones shown previously.

Pages 17-20, develop alternative proofs of previous theorems and formalise facts appearing in the motivation. There are no formalisations/Mathematica computations for this section in the original paper. Overall, we think that the full formalisation should contain the material presented until page 16 and that the last section is less important for practical purposes.

5 Conclusions

During the spring semester of 2017, I worked in developing a proof of the group law of Weierstrass elliptic curves. The system at hand was called Welder and allowed the user to reason with the basic rules of natural deduction. There were no tactics or any specialized tools to assist in proving. As it may be understood we did not go very far in our formalisation. We formalised part of the basics theorems about field theory and some of the basic proofs of the group law of the corresponding curve.

On the other hand, the pedagogical benefit of this experience can be hardly contested. We learnt that theorem proving is hard (too hard indeed). However, at the same time we did experiment the joy of problem solving. Nowadays, the mathematics student is surrounded with such a great quantity of information and materials that he can find solutions to his problems without thinking the solution by himself. He can also mistake wrong proofs for right ones without noticing. Theorem proving solves this by focusing the attention of the student. Otherwise work would not progress. I believe this is a valuable virtue of theorem proving.

A second benefit of this experience was that I was forced to do research on the methodologies that existed in order to improve the proving experience. I spent the summer holidays of 2017 reading the book Term rewriting and all that [1]. It was this experience which ultimately encouraged me to come to the Technical University of Munich and learn Isabelle.

If I learnt something during my experience with Welder was that proving correct the group law of an elliptic curve was a great benchmark to test the health of a proving engine. In much less time, Isabelle let me prove the group law of a large class of affine elliptic curves. We did actually prove more than what was made in the original paper by Hales, since we did not just verify the polynomials identities obtained through the proof but we did show the complete proving process.

On the section of projective curves, while we did find a lot more difficulties, these reduced to the use of particular theories and techniques of proving in the system. It should be noted that these proofs were not formalised at all in previous work and the only existing certificate was the Mathematica computations provided. I believe that with sufficient cooperation between the different researchers we have been in contact to, it should not take much development time to finish all the details.

Future work could tackle the implementation of algorithms for elliptic curve cryptography in Isabelle/HOL. Apparently, this was a student project proposal at the Chair of Software Reliability and Theoretical Computer Science which was never assigned. For this purpose, it could also be interesting to formalise the Montgomery family of elliptic curves since their combination with Edwards curves has been proposed to provide more robust implementations [7].

References

[1] Franz Baader and Tobias Nipkow. *Term rewriting and all that.* Cambridge university press, 1999.

[2] Clemens Ballarin. “Tutorial to locales and locale interpretation”. In: *Contribuciones científicas en honor de Mirian Andrés Gómez.* Universidad de La Rioja. 2010, pp. 123–140.

[3] Evmorfia-Iro Bartzia and Pierre-Yves Strub. “A formal library for elliptic curves in the Coq proof assistant”. In: *International Conference on Interactive Theorem Proving.* Springer. 2014, pp. 77–92.

[4] Stefan Berghofer. “The Group Law for Elliptic Curves”. In: *Archive of Formal Proofs* (Feb. 2017). `http://isa-afp.org/entries/Elliptic_Curves_Group_Law.html`, Formal proof development. ISSN: 2150-914x.

[5] Daniel J Bernstein and Tanja Lange. “Faster addition and doubling on elliptic curves”. In: *International Conference on the Theory and Application of Cryptology and Information Security.* Springer. 2007, pp. 29–50.

[6] Daniel J Bernstein et al. “Twisted edwards curves”. In: *International Conference on Cryptology in Africa.* Springer. 2008, pp. 389–405.

[7] Craig Costello and Benjamin Smith. “Montgomery curves and their arithmetic”. In: *Journal of Cryptographic Engineering* 8.3 (2018), pp. 227–240.

[8] M Prem Laxman Das and Palash Sarkar. “Pairing computation on twisted Edwards form elliptic curves”. In: *International Conference on Pairing-Based Cryptography.* Springer. 2008, pp. 192–210.

[9] Harold Edwards. “A normal form for elliptic curves”. In: *Bulletin of the American mathematical society* 44.3 (2007), pp. 393–422.

[10] Thomas Hales. “The group law for Edwards curves”. In: *arXiv preprint arXiv:1610.05278* (2016).

[11] Emilia Käsper. “Fast elliptic curve cryptography in OpenSSL”. In: *International Conference on Financial Cryptography and Data Security.* Springer. 2011, pp. 27–39.

[12] Paul Kocher, Joshua Jaffe, and Benjamin Jun. “Differential power analysis”. In: *Annual International Cryptology Conference.* Springer. 1999, pp. 388–397.

[13] Alexander Maletzky and Fabian Immler. “Gröbner Bases of Modules and Faugère’s F_4 Algorithm in Isabelle/HOL”. In: *International Conference on Intelligent Computer Mathematics.* Springer. 2018, pp. 178–193.

[14] Lawrence C Paulson. “Defining functions on equivalence classes”. In: *ACM Transactions on Computational Logic (TOCL)* 7.4 (2006), pp. 658–675.

[15] Rodrigo Raya. *Verifying Elliptic Curves using Welder.* Tech. rep. Semester project report. Laboratory of Automatic Reasoning and Analysis, EPFL, 2017. URL: `https://github.com/rjraya/WelderToolbox/blob/master/vec/documents/report.pdf` (visited on 07/14/2019).

[16] David M Russinoff. “A computationally surveyable proof of the group properties of an elliptic curve”. In: *arXiv preprint arXiv:1705.01226* (2017).

[17] Joseph H Silverman and John Torrence Tate. *Rational points on elliptic curves.* Vol. 9. Springer, 1992.

[18] Laurent Théry. “Proving the group law for elliptic curves formally”. In: (2007).

[19] Makarius Wenzel et al. *The isabelle/isar reference manual.* 2019.