

# Group Law of Edwards Elliptic Curves

Rodrigo Raya

February 6, 2026

## Abstract

This article gives an elementary computational proof of the group law for Edwards elliptic curves. The associative law is expressed as a polynomial identity over the integers that is directly checked by polynomial division. Unlike other proofs, no preliminaries such as intersection numbers, Bezouts theorem, projective geometry, divisors, or Riemann Roch are required. It supports the material of [1].

## Contents

<b>1</b>	<b>Affine Edwards curves</b>	<b>2</b>
<b>2</b>	<b>Extension</b>	<b>3</b>
2.1	Change of variables . . . . .	4
2.2	New points . . . . .	4
2.3	Group transformations and inversions . . . . .	4
2.4	Extended addition . . . . .	8
2.4.1	Inversion and rotation invariance . . . . .	9
2.4.2	Coherence and closure . . . . .	11
2.4.3	Useful lemmas in the extension . . . . .	12
2.5	Delta arithmetic . . . . .	12
<b>3</b>	<b>Projective Edwards curves</b>	<b>15</b>
3.1	No fixed-point lemma and dichotomies . . . . .	15
3.1.1	Meaning of dichotomy condition on deltas . . . . .	15
3.2	Gluing relation and projective points . . . . .	16
3.2.1	Point-class classification . . . . .	16
3.3	Projective addition on points . . . . .	18
3.4	Projective addition on classes . . . . .	18
3.4.1	Covering . . . . .	19
3.4.2	Independence of the representant . . . . .	19
3.4.3	Basic properties . . . . .	21

<b>4</b>	<b>Group law</b>	<b>22</b>
4.1	Class invariance on group operations . . . . .	22
4.2	Associativities . . . . .	25
4.3	Lemmas for associativity . . . . .	27
4.4	Group law . . . . .	29

```

theory Edwards-Elliptic-Curves-Group
  imports HOL-Algebra.Group HOL-Library.Rewrite
begin

```

## 1 Affine Edwards curves

```

class ell-field = field +
  assumes two-not-zero:  $2 \neq 0$ 

```

```

locale curve-addition =
  fixes c d :: 'a::ell-field'
begin

```

```

definition e :: 'a ⇒ 'a ⇒ 'a where
  e x y =  $x^2 + c * y^2 - 1 - d * x^2 * y^2$ 

```

```

definition delta-plus :: 'a ⇒ 'a ⇒ 'a ⇒ 'a ⇒ 'a where
  delta-plus x1 y1 x2 y2 =  $1 + d * x1 * y1 * x2 * y2$ 

```

```

definition delta-minus :: 'a ⇒ 'a ⇒ 'a ⇒ 'a ⇒ 'a where
  delta-minus x1 y1 x2 y2 =  $1 - d * x1 * y1 * x2 * y2$ 

```

```

definition delta :: 'a ⇒ 'a ⇒ 'a ⇒ 'a ⇒ 'a where
  delta x1 y1 x2 y2 = (delta-plus x1 y1 x2 y2) *
    (delta-minus x1 y1 x2 y2)

```

```

lemma delta-com:
  (delta x0 y0 x1 y1 = 0) = (delta x1 y1 x0 y0 = 0)
  <proof>

```

```

fun add :: 'a × 'a ⇒ 'a × 'a ⇒ 'a × 'a where
  add (x1,y1) (x2,y2) =
    ((x1*x2 - c*y1*y2) div ( $1 - d * x1 * y1 * x2 * y2$ ),
     (x1*y2 + y1*x2) div ( $1 + d * x1 * y1 * x2 * y2$ ))

```

```

lemma commutativity: add z1 z2 = add z2 z1
  <proof>

```

```

lemma add-closure:
  assumes add (x1,y1) (x2,y2) = (x3,y3)
  assumes delta-minus x1 y1 x2 y2 ≠ 0 delta-plus x1 y1 x2 y2 ≠ 0
  assumes e x1 y1 = 0 e x2 y2 = 0
  shows e x3 y3 = 0

```

*<proof>*

**lemma associativity:**

**assumes**  $z1' = (x1', y1')$   $z3' = (x3', y3')$

**assumes**  $z1' = \text{add } (x1, y1) (x2, y2)$   $z3' = \text{add } (x2, y2) (x3, y3)$

**assumes**  $\text{delta-minus } x1 \ y1 \ x2 \ y2 \neq 0$   $\text{delta-plus } x1 \ y1 \ x2 \ y2 \neq 0$   
 $\text{delta-minus } x2 \ y2 \ x3 \ y3 \neq 0$   $\text{delta-plus } x2 \ y2 \ x3 \ y3 \neq 0$   
 $\text{delta-minus } x1' \ y1' \ x3 \ y3 \neq 0$   $\text{delta-plus } x1' \ y1' \ x3 \ y3 \neq 0$   
 $\text{delta-minus } x1 \ y1 \ x3' \ y3' \neq 0$   $\text{delta-plus } x1 \ y1 \ x3' \ y3' \neq 0$

**assumes**  $e \ x1 \ y1 = 0$   $e \ x2 \ y2 = 0$   $e \ x3 \ y3 = 0$

**shows**  $\text{add } (\text{add } (x1, y1) (x2, y2)) (x3, y3) = \text{add } (x1, y1) (\text{add } (x2, y2) (x3, y3))$

*<proof>*

**lemma neutral:**  $\text{add } z \ (1, 0) = z$  *<proof>*

**lemma inverse:**

**assumes**  $e \ a \ b = 0$   $\text{delta-plus } a \ b \ a \ b \neq 0$

**shows**  $\text{add } (a, b) (a, -b) = (1, 0)$

*<proof>*

**lemma affine-closure:**

**assumes**  $\text{delta } x1 \ y1 \ x2 \ y2 = 0$   $e \ x1 \ y1 = 0$   $e \ x2 \ y2 = 0$

**shows**  $\exists b. (1/d = b^2 \wedge 1/d \neq 0) \vee (1/(c*d) = b^2 \wedge 1/(c*d) \neq 0)$

*<proof>*

**lemma delta-non-zero:**

**fixes**  $x1 \ y1 \ x2 \ y2$

**assumes**  $e \ x1 \ y1 = 0$   $e \ x2 \ y2 = 0$

**assumes**  $\exists b. 1/c = b^2 \neg (\exists b. b \neq 0 \wedge 1/d = b^2)$

**shows**  $\text{delta } x1 \ y1 \ x2 \ y2 \neq 0$

*<proof>*

**lemma group-law:**

**assumes**  $\exists b. 1/c = b^2 \neg (\exists b. b \neq 0 \wedge 1/d = b^2)$

**shows**  $\text{comm-group } (\text{carrier} = \{(x, y). e \ x \ y = 0\}, \text{mult} = \text{add}, \text{one} = (1, 0))$

(*is comm-group ?g*)

*<proof>*

**end**

## 2 Extension

**locale** *ext-curve-addition* = *curve-addition* +

**fixes**  $t' :: 'a :: \text{ell-field}$

**assumes** *c-eq-1*:  $c = 1$

**assumes** *t-intro*:  $d = t'^2$

**assumes** *t-ineq*:  $t'^2 \neq 1 \ t' \neq 0$

**begin**

## 2.1 Change of variables

**definition**  $t$  where  $t = t'$

**lemma**  $t\text{-nz}$ :  $t \neq 0$   $\langle$ proof $\rangle$

**lemma**  $d\text{-nz}$ :  $d \neq 0$   $\langle$ proof $\rangle$

**lemma**  $t\text{-expr}$ :  $t^2 = d t^4 = d^2$   $\langle$ proof $\rangle$

**lemma**  $t\text{-sq-n1}$ :  $t^2 \neq 1$   $\langle$ proof $\rangle$

**lemma**  $t\text{-nm1}$ :  $t \neq -1$   $\langle$ proof $\rangle$

**lemma**  $d\text{-n1}$ :  $d \neq 1$   $\langle$ proof $\rangle$

**lemma**  $t\text{-n1}$ :  $t \neq 1$   $\langle$ proof $\rangle$

**lemma**  $t\text{-dneq2}$ :  $2*t \neq -2$   
 $\langle$ proof $\rangle$

## 2.2 New points

**definition**  $e'$  where  $e' x y = x^2 + y^2 - 1 - t^2 * x^2 * y^2$

**definition**  $e'\text{-aff} = \{(x,y). e' x y = 0\}$

**definition**  $e\text{-circ} = \{(x,y). x \neq 0 \wedge y \neq 0 \wedge (x,y) \in e'\text{-aff}\}$

**lemma**  $e\text{-}e'\text{-iff}$ :  $e x y = 0 \iff e' x y = 0$   
 $\langle$ proof $\rangle$

**lemma**  $\text{circ-to-aff}$ :  $p \in e\text{-circ} \implies p \in e'\text{-aff}$   
 $\langle$ proof $\rangle$

The case  $t^2 = 1$  corresponds to a product of intersecting lines which cannot be a group

**lemma**  $t\text{-2-1-lines}$ :

$t^2 = 1 \implies e' x y = -(1 - x^2) * (1 - y^2)$   
 $\langle$ proof $\rangle$

The case  $t = 0$  corresponds to a circle which has been treated before

**lemma**  $t\text{-0-circle}$ :

$t = 0 \implies e' x y = x^2 + y^2 - 1$   
 $\langle$ proof $\rangle$

## 2.3 Group transformations and inversions

**fun**  $\rho :: 'a \times 'a \Rightarrow 'a \times 'a$  where

$\rho (x,y) = (-y,x)$

**fun**  $\tau :: 'a \times 'a \Rightarrow 'a \times 'a$  where

$$\tau(x, y) = (1/(t*x), 1/(t*y))$$

**definition**  $G$  where

$$G \equiv \{id, \varrho, \varrho \circ \varrho, \tau, \tau \circ \varrho, \tau \circ \varrho \circ \varrho, \tau \circ \varrho \circ \varrho \circ \varrho\}$$

**definition** *symmetries* where

$$symmetries = \{\tau, \tau \circ \varrho, \tau \circ \varrho \circ \varrho, \tau \circ \varrho \circ \varrho \circ \varrho\}$$

**definition** *rotations* where

$$rotations = \{id, \varrho, \varrho \circ \varrho, \varrho \circ \varrho \circ \varrho\}$$

**lemma**  $G$ -partition:  $G = rotations \cup symmetries$

*<proof>*

**lemma**  $\tau$ -sq:  $(\tau \circ \tau)(x, y) = (x, y)$  *<proof>*

**lemma**  $\tau$ -idemp:  $\tau \circ \tau = id$

*<proof>*

**lemma**  $\tau$ -idemp-explicit:  $\tau(\tau(x, y)) = (x, y)$

*<proof>*

**lemma**  $\tau$ -idemp-point:  $\tau(\tau p) = p$

*<proof>*

**fun**  $i :: 'a \times 'a \Rightarrow 'a \times 'a$  where

$$i(a, b) = (a, -b)$$

**lemma**  $i$ -idemp:  $i \circ i = id$

*<proof>*

**lemma**  $i$ -idemp-explicit:  $i(i(x, y)) = (x, y)$

*<proof>*

**lemma**  $\tau$ -rot-sym:

**assumes**  $r \in rotations$

**shows**  $\tau \circ r \in symmetries$

*<proof>*

**lemma**  $\tau$ -rho-com:

$$\tau \circ \varrho = \varrho \circ \tau$$
 *<proof>*

**lemma**  $\tau$ -rot-com:

**assumes**  $r \in rotations$

**shows**  $\tau \circ r = r \circ \tau$

*<proof>*

**lemma**  $\varrho$ -order-4:

$$\varrho \circ \varrho \circ \varrho \circ \varrho = id$$
 *<proof>*

**lemma** *rho-i-com-inverses*:

$i (id (x,y)) = id (i (x,y))$   
 $i (\varrho (x,y)) = (\varrho \circ \varrho \circ \varrho) (i (x,y))$   
 $i ((\varrho \circ \varrho) (x,y)) = (\varrho \circ \varrho) (i (x,y))$   
 $i ((\varrho \circ \varrho \circ \varrho) (x,y)) = \varrho (i (x,y))$   
(proof)

**lemma** *rotations-i-inverse*:

**assumes**  $tr \in rotations$   
**shows**  $\exists tr' \in rotations. (tr \circ i) (x,y) = (i \circ tr') (x,y) \wedge tr \circ tr' = id$   
(proof)

**lemma** *tau-i-com-inverses*:

$(i \circ \tau) (x,y) = (\tau \circ i) (x,y)$   
 $(i \circ \tau \circ \varrho) (x,y) = (\tau \circ \varrho \circ \varrho \circ \varrho \circ i) (x,y)$   
 $(i \circ \tau \circ \varrho \circ \varrho) (x,y) = (\tau \circ \varrho \circ \varrho \circ i) (x,y)$   
 $(i \circ \tau \circ \varrho \circ \varrho \circ \varrho) (x,y) = (\tau \circ \varrho \circ i) (x,y)$   
(proof)

**lemma** *rho-circ*:

**assumes**  $p \in e-circ$   
**shows**  $\varrho p \in e-circ$   
(proof)

**lemma** *i-aff*:

**assumes**  $p \in e'-aff$   
**shows**  $i p \in e'-aff$   
(proof)

**lemma** *i-circ*:

**assumes**  $(x,y) \in e-circ$   
**shows**  $i (x,y) \in e-circ$   
(proof)

**lemma** *i-circ-points*:

**assumes**  $p \in e-circ$   
**shows**  $i p \in e-circ$   
(proof)

**lemma** *rot-circ*:

**assumes**  $p \in e-circ$   $tr \in rotations$   
**shows**  $tr p \in e-circ$   
(proof)

**lemma** *tau-circ*:

**assumes**  $p \in e-circ$   
**shows**  $\tau p \in e-circ$   
(proof)

**lemma** *rot-comp*:

**assumes**  $t1 \in \text{rotations}$   $t2 \in \text{rotations}$

**shows**  $t1 \circ t2 \in \text{rotations}$

*<proof>*

**lemma** *rot-tau-com*:

**assumes**  $tr \in \text{rotations}$

**shows**  $tr \circ \tau = \tau \circ tr$

*<proof>*

**lemma** *tau-i-com*:

$\tau \circ i = i \circ \tau$  *<proof>*

**lemma** *rot-com*:

**assumes**  $r \in \text{rotations}$   $r' \in \text{rotations}$

**shows**  $r' \circ r = r \circ r'$

*<proof>*

**lemma** *rot-inv*:

**assumes**  $r \in \text{rotations}$

**shows**  $\exists r' \in \text{rotations}. r' \circ r = id$

*<proof>*

**lemma** *rot-aff*:

**assumes**  $r \in \text{rotations}$   $p \in e'\text{-aff}$

**shows**  $r p \in e'\text{-aff}$

*<proof>*

**lemma** *rot-delta*:

**assumes**  $r \in \text{rotations}$   $\text{delta } x1 \ y1 \ x2 \ y2 \neq 0$

**shows**  $\text{delta } (\text{fst } (r \ (x1,y1))) \ (\text{snd } (r \ (x1,y1))) \ x2 \ y2 \neq 0$

*<proof>*

**lemma** *tau-not-id*:  $\tau \neq id$

*<proof>*

**lemma** *sym-not-id*:

**assumes**  $r \in \text{rotations}$

**shows**  $\tau \circ r \neq id$

*<proof>*

**lemma** *sym-decomp*:

**assumes**  $g \in \text{symmetries}$

**shows**  $\exists r \in \text{rotations}. g = \tau \circ r$

*<proof>*

**lemma** *symmetries-i-inverse*:

**assumes**  $tr \in \text{symmetries}$   
**shows**  $\exists tr' \in \text{symmetries}. (tr \circ i) (x,y) = (i \circ tr') (x,y) \wedge tr \circ tr' = id$   
 <proof>

**lemma** *sym-to-rot*:  $g \in \text{symmetries} \implies \tau \circ g \in \text{rotations}$   
 <proof>

## 2.4 Extended addition

**fun** *ext-add* ::  $'a \times 'a \Rightarrow 'a \times 'a \Rightarrow 'a \times 'a$  **where**  
*ext-add* (x1,y1) (x2,y2) =  
 ((x1\*y1-x2\*y2) div (x2\*y1-x1\*y2),  
 (x1\*y1+x2\*y2) div (x1\*x2+y1\*y2))

**definition** *delta-x* ::  $'a \Rightarrow 'a \Rightarrow 'a \Rightarrow 'a \Rightarrow 'a$  **where**  
*delta-x* x1 y1 x2 y2 = x2\*y1 - x1\*y2

**definition** *delta-y* ::  $'a \Rightarrow 'a \Rightarrow 'a \Rightarrow 'a \Rightarrow 'a$  **where**  
*delta-y* x1 y1 x2 y2 = x1\*x2 + y1\*y2

**definition** *delta'* ::  $'a \Rightarrow 'a \Rightarrow 'a \Rightarrow 'a \Rightarrow 'a$  **where**  
*delta'* x1 y1 x2 y2 = *delta-x* x1 y1 x2 y2 \* *delta-y* x1 y1 x2 y2

**lemma** *delta'-com*:  $(\text{delta}' x0 y0 x1 y1 = 0) = (\text{delta}' x1 y1 x0 y0 = 0)$   
 <proof>

**definition** *e'-aff-0* **where**  
*e'-aff-0* = {((x1,y1),(x2,y2)). (x1,y1) ∈ *e'-aff* ∧  
 (x2,y2) ∈ *e'-aff* ∧  
*delta* x1 y1 x2 y2 ≠ 0 }

**definition** *e'-aff-1* **where**  
*e'-aff-1* = {((x1,y1),(x2,y2)). (x1,y1) ∈ *e'-aff* ∧  
 (x2,y2) ∈ *e'-aff* ∧  
*delta'* x1 y1 x2 y2 ≠ 0 }

**lemma** *ext-add-comm*:  
*ext-add* (x1,y1) (x2,y2) = *ext-add* (x2,y2) (x1,y1)  
 <proof>

**lemma** *ext-add-comm-points*:  
*ext-add* z1 z2 = *ext-add* z2 z1  
 <proof>

**lemma** *ext-add-inverse*:  
 $x \neq 0 \implies y \neq 0 \implies \text{ext-add} (x,y) (i (x,y)) = (1,0)$   
 <proof>

**lemma** *ext-add-deltas*:  
*ext-add* (x1,y1) (x2,y2) =  
 ((*delta-x* x2 y1 x1 y2) div (*delta-x* x1 y1 x2 y2),

(delta-y x1 x2 y1 y2) div (delta-y x1 y1 x2 y2))  
<proof>

### 2.4.1 Inversion and rotation invariance

**lemma** *inversion-invariance-1*:

**assumes**  $x1 \neq 0$   $y1 \neq 0$   $x2 \neq 0$   $y2 \neq 0$

**shows**  $add (\tau (x1,y1)) (x2,y2) = add (x1,y1) (\tau (x2,y2))$

<proof>

**lemma** *inversion-invariance-2*:

**assumes**  $x1 \neq 0$   $y1 \neq 0$   $x2 \neq 0$   $y2 \neq 0$

**shows**  $ext-add (\tau (x1,y1)) (x2,y2) = ext-add (x1,y1) (\tau (x2,y2))$

<proof>

**lemma** *rho-invariance-1*:

$add (\varrho (x1,y1)) (x2,y2) = \varrho (add (x1,y1) (x2,y2))$

<proof>

**lemma** *rho-invariance-1-points*:

$add (\varrho p1) p2 = \varrho (add p1 p2)$

<proof>

**lemma** *rho-invariance-2*:

$ext-add (\varrho (x1,y1)) (x2,y2) = \varrho (ext-add (x1,y1) (x2,y2))$

<proof>

**lemma** *rho-invariance-2-points*:

$ext-add (\varrho p1) p2 = \varrho (ext-add p1 p2)$

<proof>

**lemma** *rotation-invariance-1*:

**assumes**  $r \in rotations$

**shows**  $add (r (x1,y1)) (x2,y2) = r (add (x1,y1) (x2,y2))$

<proof>

**lemma** *rotation-invariance-1-points*:

**assumes**  $r \in rotations$

**shows**  $add (r p1) p2 = r (add p1 p2)$

<proof>

**lemma** *rotation-invariance-2*:

**assumes**  $r \in rotations$

**shows**  $ext-add (r (x1,y1)) (x2,y2) = r (ext-add (x1,y1) (x2,y2))$

<proof>

**lemma** *rotation-invariance-2-points*:

**assumes**  $r \in \text{rotations}$   
**shows**  $\text{ext-add } (r \ p1) \ p2 = r \ (\text{ext-add } p1 \ p2)$   
 $\langle \text{proof} \rangle$

**lemma** *rotation-invariance-3*:  
 $\text{delta } x1 \ y1 \ (\text{fst } (\varrho \ (x2,y2))) \ (\text{snd } (\varrho \ (x2,y2))) =$   
 $\text{delta } x1 \ y1 \ x2 \ y2$   
 $\langle \text{proof} \rangle$

**lemma** *rotation-invariance-4*:  
 $\text{delta}' \ x1 \ y1 \ (\text{fst } (\varrho \ (x2,y2))) \ (\text{snd } (\varrho \ (x2,y2))) = - \text{delta}' \ x1 \ y1 \ x2 \ y2$   
 $\langle \text{proof} \rangle$

**lemma** *rotation-invariance-5*:  
 $\text{delta}' \ (\text{fst } (\varrho \ (x1,y1))) \ (\text{snd } (\varrho \ (x1,y1))) \ x2 \ y2 = - \text{delta}' \ x1 \ y1 \ x2 \ y2$   
 $\langle \text{proof} \rangle$

**lemma** *rotation-invariance-6*:  
 $\text{delta} \ (\text{fst } (\varrho \ (x1,y1))) \ (\text{snd } (\varrho \ (x1,y1))) \ x2 \ y2 = \text{delta} \ x1 \ y1 \ x2 \ y2$   
 $\langle \text{proof} \rangle$

**lemma** *inverse-rule-1*:  
 $(\tau \circ i \circ \tau) \ (x,y) = i \ (x,y)$   
 $\langle \text{proof} \rangle$

**lemma** *inverse-rule-2*:  
 $(\varrho \circ i \circ \varrho) \ (x,y) = i \ (x,y)$   
 $\langle \text{proof} \rangle$

**lemma** *inverse-rule-3*:  
 $i \ (\text{add} \ (x1,y1) \ (x2,y2)) = \text{add} \ (i \ (x1,y1)) \ (i \ (x2,y2))$   
 $\langle \text{proof} \rangle$

**lemma** *inverse-rule-4*:  
 $i \ (\text{ext-add} \ (x1,y1) \ (x2,y2)) = \text{ext-add} \ (i \ (x1,y1)) \ (i \ (x2,y2))$   
 $\langle \text{proof} \rangle$

**lemma** *e'-aff-x0*:  
**assumes**  $x = 0 \ (x,y) \in e'\text{-aff}$   
**shows**  $y = 1 \vee y = -1$   
 $\langle \text{proof} \rangle$

**lemma** *e'-aff-y0*:  
**assumes**  $y = 0 \ (x,y) \in e'\text{-aff}$   
**shows**  $x = 1 \vee x = -1$   
 $\langle \text{proof} \rangle$

**lemma** *add-ext-add*:

**assumes**  $x1 \neq 0 \ y1 \neq 0$

**shows**  $\text{ext-add } (x1, y1) \ (x2, y2) = \tau \ (\text{add } (\tau \ (x1, y1)) \ (x2, y2))$

*<proof>*

**corollary** *add-ext-add-2*:

**assumes**  $x1 \neq 0 \ y1 \neq 0$

**shows**  $\text{add } (x1, y1) \ (x2, y2) = \tau \ (\text{ext-add } (\tau \ (x1, y1)) \ (x2, y2))$

*<proof>*

## 2.4.2 Coherence and closure

**lemma** *coherence-1*:

**assumes**  $\text{delta-x } x1 \ y1 \ x2 \ y2 \neq 0 \ \text{delta-minus } x1 \ y1 \ x2 \ y2 \neq 0$

**assumes**  $e' \ x1 \ y1 = 0 \ e' \ x2 \ y2 = 0$

**shows**  $\text{delta-x } x1 \ y1 \ x2 \ y2 \ * \ \text{delta-minus } x1 \ y1 \ x2 \ y2 \ * \\ (\text{fst } (\text{ext-add } (x1, y1) \ (x2, y2)) - \text{fst } (\text{add } (x1, y1) \ (x2, y2))) \\ = x2 \ * \ y2 \ * \ e' \ x1 \ y1 - x1 \ * \ y1 \ * \ e' \ x2 \ y2$

*<proof>*

**lemma** *coherence-2*:

**assumes**  $\text{delta-y } x1 \ y1 \ x2 \ y2 \neq 0 \ \text{delta-plus } x1 \ y1 \ x2 \ y2 \neq 0$

**assumes**  $e' \ x1 \ y1 = 0 \ e' \ x2 \ y2 = 0$

**shows**  $\text{delta-y } x1 \ y1 \ x2 \ y2 \ * \ \text{delta-plus } x1 \ y1 \ x2 \ y2 \ * \\ (\text{snd } (\text{ext-add } (x1, y1) \ (x2, y2)) - \text{snd } (\text{add } (x1, y1) \ (x2, y2))) \\ = -x2 \ * \ y2 \ * \ e' \ x1 \ y1 - x1 \ * \ y1 \ * \ e' \ x2 \ y2$

*<proof>*

**lemma** *coherence*:

**assumes**  $\text{delta } x1 \ y1 \ x2 \ y2 \neq 0 \ \text{delta}' \ x1 \ y1 \ x2 \ y2 \neq 0$

**assumes**  $e' \ x1 \ y1 = 0 \ e' \ x2 \ y2 = 0$

**shows**  $\text{ext-add } (x1, y1) \ (x2, y2) = \text{add } (x1, y1) \ (x2, y2)$

*<proof>*

**lemma** *ext-add-closure*:

**assumes**  $\text{delta}' \ x1 \ y1 \ x2 \ y2 \neq 0$

**assumes**  $e' \ x1 \ y1 = 0 \ e' \ x2 \ y2 = 0$

**assumes**  $(x3, y3) = \text{ext-add } (x1, y1) \ (x2, y2)$

**shows**  $e' \ x3 \ y3 = 0$

*<proof>*

**lemma** *ext-add-closure-points*:

**assumes**  $\text{delta}' \ x1 \ y1 \ x2 \ y2 \neq 0$

**assumes**  $(x1, y1) \in e'\text{-aff } (x2, y2) \in e'\text{-aff}$

**shows**  $\text{ext-add } (x1, y1) \ (x2, y2) \in e'\text{-aff}$

*<proof>*

### 2.4.3 Useful lemmas in the extension

**lemma** *inverse-generalized:*

**assumes**  $(a,b) \in e'\text{-aff delta-plus } a \ b \ a \ b \neq 0$

**shows**  $\text{add } (a,b) \ (a,-b) = (1,0)$

*<proof>*

**lemma** *inverse-generalized-points:*

**assumes**  $p \in e'\text{-aff delta-plus } (\text{fst } p) \ (\text{snd } p) \ (\text{fst } p) \ (\text{snd } p) \neq 0$

**shows**  $\text{add } p \ (i \ p) = (1,0)$

*<proof>*

**lemma** *add-closure-points:*

**assumes**  $\text{delta } x \ y \ x' \ y' \neq 0$

$(x,y) \in e'\text{-aff } (x',y') \in e'\text{-aff}$

**shows**  $\text{add } (x,y) \ (x',y') \in e'\text{-aff}$

*<proof>*

**lemma** *add-self:*

**assumes** *in-aff:*  $(x,y) \in e'\text{-aff}$

**shows**  $\text{delta } x \ y \ x \ (-y) \neq 0 \vee \text{delta}' \ x \ y \ x \ (-y) \neq 0$

*<proof>*

## 2.5 Delta arithmetic

**lemma** *mix-tau:*

**assumes**  $(x1,y1) \in e'\text{-aff } (x2,y2) \in e'\text{-aff } x2 \neq 0 \ y2 \neq 0$

**assumes**  $\text{delta}' \ x1 \ y1 \ x2 \ y2 \neq 0 \ \text{delta}' \ x1 \ y1 \ (\text{fst } (\tau \ (x2,y2))) \ (\text{snd } (\tau \ (x2,y2))) \neq 0$

**shows**  $\text{delta } x1 \ y1 \ x2 \ y2 \neq 0$

*<proof>*

**lemma** *mix-tau-0:*

**assumes**  $(x1,y1) \in e'\text{-aff } (x2,y2) \in e'\text{-aff } x2 \neq 0 \ y2 \neq 0$

**assumes**  $\text{delta } x1 \ y1 \ x2 \ y2 = 0$

**shows**  $\text{delta}' \ x1 \ y1 \ x2 \ y2 = 0 \vee \text{delta}' \ x1 \ y1 \ (\text{fst } (\tau \ (x2,y2))) \ (\text{snd } (\tau \ (x2,y2))) = 0$

*<proof>*

**lemma** *mix-tau-prime:*

**assumes**  $(x1,y1) \in e'\text{-aff } (x2,y2) \in e'\text{-aff } x2 \neq 0 \ y2 \neq 0$

**assumes**  $\text{delta } x1 \ y1 \ x2 \ y2 \neq 0 \ \text{delta } x1 \ y1 \ (\text{fst } (\tau \ (x2,y2))) \ (\text{snd } (\tau \ (x2,y2))) \neq 0$

**shows**  $\text{delta}' \ x1 \ y1 \ x2 \ y2 \neq 0$

*<proof>*

**lemma** *tau-tau-d:*

**assumes**  $(x1,y1) \in e'\text{-aff } (x2,y2) \in e'\text{-aff}$

**assumes**  $\text{delta } (\text{fst } (\tau \ (x1,y1))) \ (\text{snd } (\tau \ (x1,y1))) \ (\text{fst } (\tau \ (x2,y2))) \ (\text{snd } (\tau \ (x2,y2))) \neq 0$

**shows**  $\text{delta } x1 \ y1 \ x2 \ y2 \neq 0$   
 ⟨proof⟩

**lemma** *tau-tau-d'*:

**assumes**  $(x1,y1) \in e'\text{-aff } (x2,y2) \in e'\text{-aff}$   
**assumes**  $\text{delta}' (\text{fst } (\tau (x1,y1))) (\text{snd } (\tau (x1,y1))) (\text{fst } (\tau (x2,y2))) (\text{snd } (\tau (x2,y2))) \neq 0$   
**shows**  $\text{delta}' x1 \ y1 \ x2 \ y2 \neq 0$   
 ⟨proof⟩

**lemma** *delta-add-delta'-1*:

**assumes**  $1: x1 \neq 0 \ y1 \neq 0 \ x2 \neq 0 \ y2 \neq 0$   
**assumes**  $r\text{-expr}: rx = \text{fst } (\text{add } (x1,y1) (x2,y2)) \ ry = \text{snd } (\text{add } (x1,y1) (x2,y2))$   
  
**assumes**  $\text{in-aff}: (x1,y1) \in e'\text{-aff } (x2,y2) \in e'\text{-aff}$   
**assumes**  $\text{pd}: \text{delta } x1 \ y1 \ x2 \ y2 \neq 0$   
**assumes**  $\text{pd}': \text{delta } rx \ ry (\text{fst } (\tau (i (x2,y2)))) (\text{snd } (\tau (i (x2,y2)))) \neq 0$   
**shows**  $\text{delta}' rx \ ry (\text{fst } (i (x2,y2))) (\text{snd } (i (x2,y2))) \neq 0$   
 ⟨proof⟩

**lemma** *delta'-add-delta-1*:

**assumes**  $1: x1 \neq 0 \ y1 \neq 0 \ x2 \neq 0 \ y2 \neq 0$   
**assumes**  $r\text{-expr}: rx = \text{fst } (\text{ext-add } (x1,y1) (x2,y2)) \ ry = \text{snd } (\text{ext-add } (x1,y1) (x2,y2))$   
  
**assumes**  $\text{in-aff}: (x1,y1) \in e'\text{-aff } (x2,y2) \in e'\text{-aff}$   
**assumes**  $\text{pd}': \text{delta}' rx \ ry (\text{fst } (\tau (i (x2,y2)))) (\text{snd } (\tau (i (x2,y2)))) \neq 0$   
**shows**  $\text{delta } rx \ ry (\text{fst } (i (x2,y2))) (\text{snd } (i (x2,y2))) \neq 0$   
 ⟨proof⟩

**lemma** *funny-field-lemma-1*:

$((x1 * x2 - y1 * y2) * ((x1 * x2 - y1 * y2) * (x2 * (y2 * (1 + d * x1 * y1 * x2 * y2)))) +$   
 $(x1 * x2 - y1 * y2) * ((x1 * y2 + y1 * x2) * y2^2) * (1 - d * x1 * y1 * x2 * y2) *$   
 $(1 + d * x1 * y1 * x2 * y2) \neq$   
 $((x1 * y2 + y1 * x2) * ((x1 * y2 + y1 * x2) * (x2 * (y2 * (1 - d * x1 * y1 * x2 * y2)))) +$   
 $(x1 * x2 - y1 * y2) * ((x1 * y2 + y1 * x2) * x2^2) * (1 + d * x1 * y1 * x2 * y2) *$   
 $(1 - d * x1 * y1 * x2 * y2) \implies$   
 $(d * ((x1 * x2 - y1 * y2) * ((x1 * y2 + y1 * x2) * (x2 * y2))))^2 =$   
 $((1 - d * x1 * y1 * x2 * y2) * (1 + d * x1 * y1 * x2 * y2))^2 \implies$   
 $x1^2 + y1^2 - 1 = d * x1^2 * y1^2 \implies$   
 $x2^2 + y2^2 - 1 = d * x2^2 * y2^2 \implies \text{False}$   
 ⟨proof⟩

**lemma** *delta-add-delta'-2*:

**assumes**  $1: x1 \neq 0 \ y1 \neq 0 \ x2 \neq 0 \ y2 \neq 0$

**assumes** *r-expr*:  $rx = fst (add (x1,y1) (x2,y2))$   $ry = snd (add (x1,y1) (x2,y2))$

**assumes** *in-aff*:  $(x1,y1) \in e'\text{-aff}$   $(x2,y2) \in e'\text{-aff}$

**assumes** *pd*:  $delta\ x1\ y1\ x2\ y2 \neq 0$

**assumes** *pd'*:  $delta'\ rx\ ry\ (fst\ (\tau\ (i\ (x2,y2))))\ (snd\ (\tau\ (i\ (x2,y2)))) \neq 0$

**shows**  $delta\ rx\ ry\ (fst\ (i\ (x2,y2)))\ (snd\ (i\ (x2,y2))) \neq 0$

*<proof>*

**lemma** *funny-field-lemma-2*:  $(x2 * y2)^2 * ((x2 * y1 - x1 * y2) * (x1 * x2 + y1 * y2))^2 \neq ((x1 * y1 - x2 * y2) * (x1 * y1 + x2 * y2))^2 \implies$

$((x1 * y1 - x2 * y2) * ((x1 * y1 - x2 * y2) * (x2 * (y2 * (x1 * x2 + y1 * y2)))) +$

$(x1 * y1 - x2 * y2) * ((x1 * y1 + x2 * y2) * x2^2) * (x2 * y1 - x1 * y2) * (x1 * x2 + y1 * y2) =$

$((x1 * y1 + x2 * y2) * ((x1 * y1 + x2 * y2) * (x2 * (y2 * (x2 * y1 - x1 * y2)))) +$

$(x1 * y1 - x2 * y2) * ((x1 * y1 + x2 * y2) * y2^2) * (x1 * x2 + y1 * y2) * (x2 * y1 - x1 * y2) \implies$

$x1^2 + y1^2 - 1 = d * x1^2 * y1^2 \implies$

$x2^2 + y2^2 - 1 = d * x2^2 * y2^2 \implies False$

*<proof>*

**lemma** *delta'-add-delta-2*:

**assumes** *1*:  $x1 \neq 0$   $y1 \neq 0$   $x2 \neq 0$   $y2 \neq 0$

**assumes** *r-expr*:  $rx = fst (ext-add (x1,y1) (x2,y2))$   $ry = snd (ext-add (x1,y1) (x2,y2))$

**assumes** *in-aff*:  $(x1,y1) \in e'\text{-aff}$   $(x2,y2) \in e'\text{-aff}$

**assumes** *pd*:  $delta'\ x1\ y1\ x2\ y2 \neq 0$

**assumes** *pd'*:  $delta\ rx\ ry\ (fst\ (\tau\ (i\ (x2,y2))))\ (snd\ (\tau\ (i\ (x2,y2)))) \neq 0$

**shows**  $delta'\ rx\ ry\ (fst\ (i\ (x2,y2)))\ (snd\ (i\ (x2,y2))) \neq 0$

*<proof>*

**lemma** *delta'-add-delta-not-add*:

**assumes** *1*:  $x1 \neq 0$   $y1 \neq 0$   $x2 \neq 0$   $y2 \neq 0$

**assumes** *in-aff*:  $(x1,y1) \in e'\text{-aff}$   $(x2,y2) \in e'\text{-aff}$

**assumes** *pd*:  $delta'\ x1\ y1\ x2\ y2 \neq 0$

**assumes** *add-nz*:  $fst (ext-add (x1,y1) (x2,y2)) \neq 0$   $snd (ext-add (x1,y1) (x2,y2)) \neq 0$

**shows** *pd'*:  $delta\ (fst\ (\tau\ (x1,y1)))\ (snd\ (\tau\ (x1,y1)))\ x2\ y2 \neq 0$

*<proof>*

**lemma** *not-add-self*:

**assumes** *in-aff*:  $(x,y) \in e'\text{-aff}$   $x \neq 0$   $y \neq 0$

**shows**  $delta\ x\ y\ (fst\ (\tau\ (i\ (x,y))))\ (snd\ (\tau\ (i\ (x,y)))) = 0$

$delta'\ x\ y\ (fst\ (\tau\ (i\ (x,y))))\ (snd\ (\tau\ (i\ (x,y)))) = 0$

*<proof>*

## 3 Projective Edwards curves

### 3.1 No fixed-point lemma and dichotomies

**lemma** *g-no-fp*:

**assumes**  $g \in G$   $p \in e\text{-circ}$   $g p = p$

**shows**  $g = id$

*<proof>*

**lemma** *dichotomy-1*:

**assumes**  $p \in e'\text{-aff}$   $q \in e'\text{-aff}$

**shows**  $(p \in e\text{-circ} \wedge (\exists g \in \text{symmetries. } q = (g \circ i) p)) \vee$   
 $(p, q) \in e'\text{-aff-0} \vee (p, q) \in e'\text{-aff-1}$

*<proof>*

**lemma** *dichotomy-2*:

**assumes**  $add(x1, y1)(x2, y2) = (1, 0)$

$((x1, y1), (x2, y2)) \in e'\text{-aff-0}$

**shows**  $(x2, y2) = i(x1, y1)$

*<proof>*

**lemma** *dichotomy-3*:

**assumes**  $ext\text{-add}(x1, y1)(x2, y2) = (1, 0)$

$((x1, y1), (x2, y2)) \in e'\text{-aff-1}$

**shows**  $(x2, y2) = i(x1, y1)$

*<proof>*

#### 3.1.1 Meaning of dichotomy condition on deltas

**lemma** *wd-d-nz*:

**assumes**  $g \in \text{symmetries}$   $(x', y') = (g \circ i)(x, y)$   $(x, y) \in e\text{-circ}$

**shows**  $\text{delta } x y x' y' = 0$

*<proof>*

**lemma** *wd-d'-nz*:

**assumes**  $g \in \text{symmetries}$   $(x', y') = (g \circ i)(x, y)$   $(x, y) \in e\text{-circ}$

**shows**  $\text{delta}' x y x' y' = 0$

*<proof>*

**lemma** *meaning-of-dichotomy-1*:

**assumes**  $(\exists g \in \text{symmetries. } (x2, y2) = (g \circ i)(x1, y1))$

**shows**  $\text{fst}(add(x1, y1)(x2, y2)) = 0 \vee \text{snd}(add(x1, y1)(x2, y2)) = 0$

*<proof>*

**lemma** *meaning-of-dichotomy-2*:

**assumes**  $(\exists g \in \text{symmetries. } (x2, y2) = (g \circ i)(x1, y1))$

**shows**  $\text{fst}(ext\text{-add}(x1, y1)(x2, y2)) = 0 \vee \text{snd}(ext\text{-add}(x1, y1)(x2, y2)) = 0$

*<proof>*

## 3.2 Gluing relation and projective points

**definition** *gluing* ::  $((('a \times 'a) \times \text{bool}) \times ((('a \times 'a) \times \text{bool}))$  set where

$$\begin{aligned} \text{gluing} = \{ & ((x0, y0), l), ((x1, y1), j)) \cdot \\ & ((x0, y0) \in e'\text{-aff} \wedge (x1, y1) \in e'\text{-aff}) \wedge \\ & ((x0 \neq 0 \wedge y0 \neq 0 \wedge (x1, y1) = \tau(x0, y0) \wedge j = \text{Not } l) \vee \\ & (x0 = x1 \wedge y0 = y1 \wedge l = j)) \} \end{aligned}$$

**lemma** *gluing-char*:

assumes  $((x0, y0), l), ((x1, y1), j) \in \text{gluing}$   
 shows  $((x0, y0) = (x1, y1) \wedge l = j) \vee ((x1, y1) = \tau(x0, y0) \wedge l = \text{Not } j \wedge x0 \neq 0 \wedge y0 \neq 0)$   
 $\langle \text{proof} \rangle$

**lemma** *gluing-char-zero*:

assumes  $((x0, y0), l), ((x1, y1), j) \in \text{gluing}$   $x0 = 0 \vee y0 = 0$   
 shows  $(x0, y0) = (x1, y1) \wedge l = j$   
 $\langle \text{proof} \rangle$

**lemma** *gluing-aff*:

assumes  $((x0, y0), l), ((x1, y1), j) \in \text{gluing}$   
 shows  $(x0, y0) \in e'\text{-aff} \wedge (x1, y1) \in e'\text{-aff}$   
 $\langle \text{proof} \rangle$

**definition** *e'-aff-bit* ::  $((('a \times 'a) \times \text{bool})$  set where

$$e'\text{-aff-bit} = e'\text{-aff} \times \text{UNIV}$$

**lemma** *eq-rel*: equiv *e'-aff-bit* *gluing*

$\langle \text{proof} \rangle$

**lemma** *gluing-eq*:  $x = y \implies \text{gluing} \text{ `` } \{x\} = \text{gluing} \text{ `` } \{y\}$

$\langle \text{proof} \rangle$

**definition** *e-proj* where  $e\text{-proj} = e'\text{-aff-bit} // \text{gluing}$

### 3.2.1 Point-class classification

**lemma** *eq-class-simp*:

assumes  $X \in e\text{-proj}$   $X \neq \{\}$   
 shows  $X // \text{gluing} = \{X\}$   
 $\langle \text{proof} \rangle$

**lemma** *gluing-class-1*:

assumes  $x = 0 \vee y = 0$   $(x, y) \in e'\text{-aff}$   
 shows  $\text{gluing} \text{ `` } \{((x, y), l)\} = \{((x, y), l)\}$   
 $\langle \text{proof} \rangle$

**lemma** *gluing-class-2*:

assumes  $x \neq 0$   $y \neq 0$   $(x, y) \in e'\text{-aff}$   
 shows  $\text{gluing} \text{ `` } \{((x, y), l)\} = \{((x, y), l), (\tau(x, y), \text{Not } l)\}$

$\langle proof \rangle$

**lemma** *e-proj-elim-1*:

**assumes**  $(x,y) \in e'\text{-aff}$

**shows**  $\{((x,y),l)\} \in e\text{-proj} \longleftrightarrow x = 0 \vee y = 0$

$\langle proof \rangle$

**lemma** *e-proj-elim-2*:

**assumes**  $(x,y) \in e'\text{-aff}$

**shows**  $\{((x,y),l),(\tau(x,y),\text{Not } l)\} \in e\text{-proj} \longleftrightarrow x \neq 0 \wedge y \neq 0$

$\langle proof \rangle$

**lemma** *e-proj-eq*:

**assumes**  $p \in e\text{-proj}$

**shows**  $\exists x y l. (p = \{((x,y),l)\} \vee p = \{((x,y),l),(\tau(x,y),\text{Not } l)\}) \wedge (x,y) \in e'\text{-aff}$

$\langle proof \rangle$

**lemma** *e-proj-aff*:

*gluing* “  $\{((x,y),l)\} \in e\text{-proj} \longleftrightarrow (x,y) \in e'\text{-aff}$

$\langle proof \rangle$

**lemma** *gluing-cases*:

**assumes**  $x \in e\text{-proj}$

**obtains**  $x_0 y_0 l$  **where**  $x = \{((x_0,y_0),l)\} \vee x = \{((x_0,y_0),l),(\tau(x_0,y_0),\text{Not } l)\}$

$\langle proof \rangle$

**lemma** *gluing-cases-explicit*:

**assumes**  $x \in e\text{-proj}$   $x = \text{gluing} \text{ “ } \{((x_0,y_0),l)\}$

**shows**  $x = \{((x_0,y_0),l)\} \vee x = \{((x_0,y_0),l),(\tau(x_0,y_0),\text{Not } l)\}$

$\langle proof \rangle$

**lemma** *gluing-cases-points*:

**assumes**  $x \in e\text{-proj}$   $x = \text{gluing} \text{ “ } \{(p,l)\}$

**shows**  $x = \{(p,l)\} \vee x = \{(p,l),(\tau p,\text{Not } l)\}$

$\langle proof \rangle$

**lemma** *identity-equiv*:

*gluing* “  $\{((1,0),l)\} = \{((1,0),l)\}$

$\langle proof \rangle$

**lemma** *identity-proj*:

$\{((1,0),l)\} \in e\text{-proj}$

$\langle proof \rangle$

**lemma** *gluing-inv*:

**assumes**  $x \neq 0$   $y \neq 0$   $(x,y) \in e'\text{-aff}$

**shows** *gluing* “  $\{((x,y),j)\} = \text{gluing} \text{ “ } \{(\tau(x,y), \text{Not } j)\}$

*<proof>*

### 3.3 Projective addition on points

**definition** *xor* :: *bool* => *bool* => *bool*  
where *xor-def*: *xor P Q* ≡ (*P* ∧ ¬ *Q*) ∨ (¬ *P* ∧ *Q*)

**function** (*domintros*) *proj-add* :: (*'a* × *'a*) × *bool* => (*'a* × *'a*) × *bool* => (*'a* × *'a*) × *bool*

where

*proj-add* ((*x1*, *y1*), *l*) ((*x2*, *y2*), *j*) = (*add* (*x1*, *y1*) (*x2*, *y2*), *xor l j*)  
if *delta x1 y1 x2 y2* ≠ 0 and  
  (*x1*, *y1*) ∈ *e'-aff* and  
  (*x2*, *y2*) ∈ *e'-aff*  
| *proj-add* ((*x1*, *y1*), *l*) ((*x2*, *y2*), *j*) = (*ext-add* (*x1*, *y1*) (*x2*, *y2*), *xor l j*)  
if *delta' x1 y1 x2 y2* ≠ 0 and  
  (*x1*, *y1*) ∈ *e'-aff* and  
  (*x2*, *y2*) ∈ *e'-aff*  
| *proj-add* ((*x1*, *y1*), *l*) ((*x2*, *y2*), *j*) = *undefined*  
if (*x1*, *y1*) ∉ *e'-aff* ∨ (*x2*, *y2*) ∉ *e'-aff* ∨  
  (*delta x1 y1 x2 y2* = 0 ∧ *delta' x1 y1 x2 y2* = 0)  
*<proof>*

**termination** *proj-add* *<proof>*

**lemma** *proj-add-inv*:

assumes (*x0,y0*) ∈ *e'-aff*  
shows *proj-add* ((*x0,y0*),*l*) (*i* (*x0,y0*),*l'*) = ((1,0),*xor l l'*)  
*<proof>*

**lemma** *proj-add-comm*:

*proj-add* ((*x0,y0*),*l*) ((*x1,y1*),*j*) = *proj-add* ((*x1,y1*),*j*) ((*x0,y0*),*l*)  
*<proof>*

### 3.4 Projective addition on classes

**function** (*domintros*) *proj-add-class* :: ((*'a* × *'a*) × *bool*) *set* =>  
  ((*'a* × *'a*) × *bool*) *set* =>  
  (((*'a* × *'a*) × *bool*) *set*) *set*

where

*proj-add-class c1 c2* =  
(  
  {  
    *proj-add* ((*x1*, *y1*), *i*) ((*x2*, *y2*), *j*) |  
    *x1 y1 i x2 y2 j*.  
    ((*x1*, *y1*), *i*) ∈ *c1* ∧  
    ((*x2*, *y2*), *j*) ∈ *c2* ∧  
    ((*x1*, *y1*), (*x2*, *y2*)) ∈ *e'-aff-0* ∪ *e'-aff-1*  
  } // *gluing*  
)

**if**  $c1 \in e\text{-proj}$  **and**  $c2 \in e\text{-proj}$   
 |  $\text{proj-add-class } c1 \ c2 = \text{undefined}$   
**if**  $c1 \notin e\text{-proj} \vee c2 \notin e\text{-proj}$   
 $\langle \text{proof} \rangle$

**termination**  $\text{proj-add-class} \langle \text{proof} \rangle$

**definition**  $\text{proj-addition}$  **where**  
 $\text{proj-addition } c1 \ c2 = \text{the-elem } (\text{proj-add-class } c1 \ c2)$

### 3.4.1 Covering

**corollary**  $\text{no-fp-eq}$ :  
**assumes**  $p \in e\text{-circ}$   
**assumes**  $r' \in \text{rotations}$   $r \in \text{rotations}$   
**assumes**  $(r' \circ i) \ p = (\tau \circ r) \ (i \ p)$   
**shows**  $\text{False}$   
 $\langle \text{proof} \rangle$

**lemma**  $\text{covering}$ :  
**assumes**  $p \in e\text{-proj}$   $q \in e\text{-proj}$   
**shows**  $\text{proj-add-class } p \ q \neq \{\}$   
 $\langle \text{proof} \rangle$

**lemma**  $\text{covering-with-deltas}$ :  
**assumes**  $(\text{gluing } \{((x,y),l)\}) \in e\text{-proj}$   $(\text{gluing } \{((x',y'),l')\}) \in e\text{-proj}$   
**shows**  $\text{delta } x \ y \ x' \ y' \neq 0 \vee \text{delta}' \ x \ y \ x' \ y' \neq 0 \vee$   
 $\text{delta } x \ y \ (\text{fst } (\tau \ (x',y'))) \ (\text{snd } (\tau \ (x',y'))) \neq 0 \vee$   
 $\text{delta}' \ x \ y \ (\text{fst } (\tau \ (x',y'))) \ (\text{snd } (\tau \ (x',y'))) \neq 0$   
 $\langle \text{proof} \rangle$

### 3.4.2 Independence of the representant

**lemma**  $\text{proj-add-class-comm}$ :  
**assumes**  $c1 \in e\text{-proj}$   $c2 \in e\text{-proj}$   
**shows**  $\text{proj-add-class } c1 \ c2 = \text{proj-add-class } c2 \ c1$   
 $\langle \text{proof} \rangle$

**lemma**  $\text{gluing-add-1}$ :  
**assumes**  $\text{gluing } \{((x,y),l)\} = \{(x, y), l\}$   $\text{gluing } \{((x',y'),l')\} = \{(x', y'), l'\}$   
 $\text{gluing } \{((x,y),l)\} \in e\text{-proj}$   $\text{gluing } \{((x',y'),l')\} \in e\text{-proj}$   $\text{delta } x \ y \ x' \ y' \neq 0$   
**shows**  $\text{proj-addition } (\text{gluing } \{((x,y),l)\}) \ (\text{gluing } \{((x',y'),l')\}) = (\text{gluing } \{(\text{add } (x,y) \ (x',y'), \text{ xor } l \ l')\})$   
 $\langle \text{proof} \rangle$

**lemma**  $\text{gluing-add-2}$ :

**assumes** *gluing* “  $\{(x,y),l\} = \{(x, y), l\}$  *gluing* “  $\{(x',y'),l'\} = \{(x', y'), l'\}, (\tau (x', y'), \text{Not } l')$  }  
*gluing* “  $\{(x,y),l\} \in e\text{-proj}$  *gluing* “  $\{(x',y'),l'\} \in e\text{-proj}$   $\delta x y x' y' \neq 0$   
**shows** *proj-addition* (*gluing* “  $\{(x,y),l\}$ ) (*gluing* “  $\{(x',y'),l'\}$ ) = (*gluing* “  $\{(add (x,y) (x',y'), xor l l')\}$ )  
 ⟨*proof*⟩

**lemma** *gluing-add-4*:

**assumes** *gluing* “  $\{(x, y), l\} = \{(x, y), l, (\tau (x, y), \text{Not } l)\}$   
*gluing* “  $\{(x', y'), l'\} = \{(x', y'), l', (\tau (x', y'), \text{Not } l')\}$   
*gluing* “  $\{(x, y), l\} \in e\text{-proj}$  *gluing* “  $\{(x', y'), l'\} \in e\text{-proj}$   $\delta x y x' y' \neq 0$   
**shows** *proj-addition* (*gluing* “  $\{(x, y), l\}$ ) (*gluing* “  $\{(x', y'), l'\}$ ) =  
*gluing* “  $\{(add (x, y) (x',y'), xor l l')\}$   
 (is *proj-addition* ?p ?q = -)  
 ⟨*proof*⟩

**lemma** *gluing-add*:

**assumes** *gluing* “  $\{(x1,y1),l\} \in e\text{-proj}$  *gluing* “  $\{(x2,y2),j\} \in e\text{-proj}$   $\delta x1 y1 x2 y2 \neq 0$   
**shows** *proj-addition* (*gluing* “  $\{(x1,y1),l\}$ ) (*gluing* “  $\{(x2,y2),j\}$ ) =  
*gluing* “  $\{(add (x1,y1) (x2,y2), xor l j)\}$   
 ⟨*proof*⟩

**lemma** *gluing-ext-add-1*:

**assumes** *gluing* “  $\{(x,y),l\} = \{(x, y), l\}$  *gluing* “  $\{(x',y'),l'\} = \{(x', y'), l'\}$   
*gluing* “  $\{(x,y),l\} \in e\text{-proj}$  *gluing* “  $\{(x',y'),l'\} \in e\text{-proj}$   $\delta' x y x' y' \neq 0$   
**shows** *proj-addition* (*gluing* “  $\{(x,y),l\}$ ) (*gluing* “  $\{(x',y'),l'\}$ ) =  
*gluing* “  $\{(ext-add (x,y) (x',y'), xor l l')\}$   
 ⟨*proof*⟩

**lemma** *gluing-ext-add-2*:

**assumes** *gluing* “  $\{(x,y),l\} = \{(x, y), l\}$  *gluing* “  $\{(x',y'),l'\} = \{(x', y'), l'\}, (\tau (x', y'), \text{Not } l')$   
*gluing* “  $\{(x,y),l\} \in e\text{-proj}$  *gluing* “  $\{(x',y'),l'\} \in e\text{-proj}$   $\delta' x y x' y' \neq 0$   
**shows** *proj-addition* (*gluing* “  $\{(x,y),l\}$ ) (*gluing* “  $\{(x',y'),l'\}$ ) = (*gluing* “  $\{(ext-add (x,y) (x',y'), xor l l')\}$ )  
 ⟨*proof*⟩

**lemma** *gluing-ext-add-4*:

**assumes** *gluing* “  $\{(x,y),l\} = \{(x, y), l, (\tau (x, y), \text{Not } l)\}$   
*gluing* “  $\{(x',y'),l'\} = \{(x', y'), l', (\tau (x', y'), \text{Not } l')\}$   
*gluing* “  $\{(x,y),l\} \in e\text{-proj}$  *gluing* “  $\{(x',y'),l'\} \in e\text{-proj}$

$\text{delta}' x y x' y' \neq 0$   
**shows** *proj-addition* (*gluing* “  $\{(x,y),l\}$ ”) (*gluing* “  $\{(x',y'),l'\}$ ”) = (*gluing* “  $\{(ext\text{-}add (x,y) (x',y'),xor l l')\}$ ”)  
 (**is** *proj-addition* ?p ?q = -)  
 ⟨*proof*⟩

**lemma** *gluing-ext-add*:

**assumes** *gluing* “  $\{(x1,y1),l\} \in e\text{-proj}$ ” *gluing* “  $\{(x2,y2),j\} \in e\text{-proj}$ ”  $\text{delta}' x1 y1 x2 y2 \neq 0$   
**shows** *proj-addition* (*gluing* “  $\{(x1,y1),l\}$ ”) (*gluing* “  $\{(x2,y2),j\}$ ”) =  
 (*gluing* “  $\{(ext\text{-}add (x1,y1) (x2,y2),xor l j)\}$ ”)  
 ⟨*proof*⟩

**lemma** *gluing-ext-add-points*:

**assumes** *gluing* “  $\{(p1,l)\} \in e\text{-proj}$ ” *gluing* “  $\{(p2,j)\} \in e\text{-proj}$ ”  $\text{delta}' (fst p1) (snd p1) (fst p2) (snd p2) \neq 0$   
**shows** *proj-addition* (*gluing* “  $\{(p1,l)\}$ ”) (*gluing* “  $\{(p2,j)\}$ ”) =  
 (*gluing* “  $\{(ext\text{-}add p1 p2,xor l j)\}$ ”)  
 ⟨*proof*⟩

### 3.4.3 Basic properties

**theorem** *well-defined*:

**assumes**  $p \in e\text{-proj}$   $q \in e\text{-proj}$   
**shows** *proj-addition*  $p q \in e\text{-proj}$   
 ⟨*proof*⟩

**lemma** *proj-add-class-inv*:

**assumes** *gluing* “  $\{(x,y),l\} \in e\text{-proj}$ ”  
**shows** *proj-addition* (*gluing* “  $\{(x,y),l\}$ ”) (*gluing* “  $\{(i (x,y),l')\}$ ”) =  $\{(1, 0), xor l l'\}$   
*gluing* “  $\{(i (x,y),l')\} \in e\text{-proj}$ ”  
 ⟨*proof*⟩

**lemma** *proj-add-class-inv-point*:

**assumes** *gluing* “  $\{(p,l)\} \in e\text{-proj}$ ”  $ne = (1,0)$   
**shows** *proj-addition* (*gluing* “  $\{(p,l)\}$ ”) (*gluing* “  $\{(i p,l')\}$ ”) =  $\{(ne, xor l l')\}$   
*gluing* “  $\{(i p,l')\} \in e\text{-proj}$ ”  
 ⟨*proof*⟩

**lemma** *proj-add-class-identity*:

**assumes**  $x \in e\text{-proj}$   
**shows** *proj-addition*  $\{(1, 0), False\}$   $x = x$   
 ⟨*proof*⟩

**corollary** *proj-addition-comm*:

**assumes**  $c1 \in e\text{-proj}$   $c2 \in e\text{-proj}$   
**shows** *proj-addition*  $c1 c2 = \text{proj-addition } c2 c1$   
 ⟨*proof*⟩

## 4 Group law

### 4.1 Class invariance on group operations

**definition** *tf* where

$$tf\ g = image\ (\lambda\ p.\ (g\ (fst\ p),\ snd\ p))$$

**lemma** *tf-comp*:

$$tf\ g\ (tf\ f\ s) = tf\ (g\ \circ\ f)\ s$$

*<proof>*

**lemma** *tf-id*:

$$tf\ id\ s = s$$

*<proof>*

**lemma** *tf-cong*:

$$f = f' \implies s = s' \implies tf\ f\ s = tf\ f'\ s'$$

*<proof>*

**definition** *tf'* where

$$tf' = image\ (\lambda\ p.\ (fst\ p,\ Not\ (snd\ p)))$$

**lemma** *tf-tf'-commute*:

$$tf\ r\ (tf'\ p) = tf'\ (tf\ r\ p)$$

*<proof>*

**lemma** *rho-preserv-e-proj*:

**assumes** *gluing* “  $\{(x, y), l\} \in e\text{-proj}$   
**shows** *tf*  $\varrho$  (*gluing* “  $\{(x, y), l\} \in e\text{-proj}$ )  
*<proof>*

**lemma** *rho-preserv-e-proj-point*:

**assumes** *gluing* “  $\{p\} \in e\text{-proj}$   
**shows** *tf*  $\varrho$  (*gluing* “  $\{p\} \in e\text{-proj}$ )  
*<proof>*

**lemma** *insert-rho-gluing*:

**assumes** *gluing* “  $\{(x, y), l\} \in e\text{-proj}$   
**shows** *tf*  $\varrho$  (*gluing* “  $\{(x, y), l\}$ ) = *gluing* “  $\{\varrho(x, y), l\}$   
*<proof>*

**lemma** *insert-rho-gluing-point*:

**assumes** *gluing* “  $\{p, l\} \in e\text{-proj}$   
**shows** *tf*  $\varrho$  (*gluing* “  $\{p, l\}$ ) = *gluing* “  $\{\varrho p, l\}$   
*<proof>*

**lemma** *rotation-preserv-e-proj*:

**assumes** *gluing* “  $\{(x, y), l\} \in e\text{-proj}$  *r*  $\in$  *rotations*  
**shows** *tf* *r* (*gluing* “  $\{(x, y), l\} \in e\text{-proj}$ )  
**(is** *tf* ?*r* ?*g*  $\in$  -)

$\langle proof \rangle$

**lemma** *rotation-preserv-e-proj-point:*

**assumes** *gluing* “  $\{p\} \in e\text{-proj } r \in \text{rotations}$

**shows**  $tf\ r\ (\text{gluing}\ \{\{p\} \in e\text{-proj}$

$\langle proof \rangle$

**lemma** *insert-rotation-gluing:*

**assumes** *gluing* “  $\{(x, y), l\} \in e\text{-proj } r \in \text{rotations}$

**shows**  $tf\ r\ (\text{gluing}\ \{(x, y), l\}) = \text{gluing}\ \{(r\ (x, y), l)\}$

$\langle proof \rangle$

**lemma** *insert-rotation-gluing-point:*

**assumes** *gluing* “  $\{p, l\} \in e\text{-proj } r \in \text{rotations}$

**shows**  $tf\ r\ (\text{gluing}\ \{(p, l)\}) = \text{gluing}\ \{(r\ p, l)\}$

$\langle proof \rangle$

**lemma** *tf-tau:*

**assumes** *gluing* “  $\{(x, y), l\} \in e\text{-proj}$

**shows**  $\text{gluing}\ \{(x, y), \text{Not } l\} = tf'\ (\text{gluing}\ \{(x, y), l\})$

$\langle proof \rangle$

**lemma** *tf-preserv-e-proj:*

**assumes** *gluing* “  $\{(x, y), l\} \in e\text{-proj}$

**shows**  $tf'\ (\text{gluing}\ \{(x, y), l\}) \in e\text{-proj}$

$\langle proof \rangle$

**lemma** *tf-preserv-e-proj-point:*

**assumes** *gluing* “  $\{p\} \in e\text{-proj}$

**shows**  $tf'\ (\text{gluing}\ \{p\}) \in e\text{-proj}$

$\langle proof \rangle$

**lemma** *remove-rho:*

**assumes** *gluing* “  $\{(x, y), l\} \in e\text{-proj}$

**shows**  $\text{gluing}\ \{(\varrho\ (x, y), l)\} = tf\ \varrho\ (\text{gluing}\ \{(x, y), l\})$

$\langle proof \rangle$

**lemma** *remove-rotations:*

**assumes** *gluing* “  $\{(x, y), l\} \in e\text{-proj } r \in \text{rotations}$

**shows**  $\text{gluing}\ \{(r\ (x, y), l)\} = tf\ r\ (\text{gluing}\ \{(x, y), l\})$

$\langle proof \rangle$

**lemma** *remove-tau:*

**assumes** *gluing* “  $\{(x, y), l\} \in e\text{-proj}$  *gluing* “  $\{(\tau\ (x, y), l)\} \in e\text{-proj}$

**shows**  $\text{gluing}\ \{(\tau\ (x, y), l)\} = tf'\ (\text{gluing}\ \{(x, y), l\})$

(**is**  $?gt = tf'\ ?g$ )

$\langle proof \rangle$

**lemma** *remove-add-rho*:

**assumes**  $p \in e\text{-proj}$   $q \in e\text{-proj}$

**shows**  $\text{proj-addition } (tf \ \varrho \ p) \ q = tf \ \varrho \ (\text{proj-addition } p \ q)$

*<proof>*

**lemma** *remove-add-rotation*:

**assumes**  $p \in e\text{-proj}$   $q \in e\text{-proj}$   $r \in \text{rotations}$

**shows**  $\text{proj-addition } (tf \ r \ p) \ q = tf \ r \ (\text{proj-addition } p \ q)$

*<proof>*

**lemma** *remove-add-tau*:

**assumes**  $p \in e\text{-proj}$   $q \in e\text{-proj}$

**shows**  $\text{proj-addition } (tf' \ p) \ q = tf' \ (\text{proj-addition } p \ q)$

*<proof>*

**lemma** *remove-add-tau'*:

**assumes**  $p \in e\text{-proj}$   $q \in e\text{-proj}$

**shows**  $\text{proj-addition } p \ (tf' \ q) = tf' \ (\text{proj-addition } p \ q)$

*<proof>*

**lemma** *tf'-idemp*:

**assumes**  $s \in e\text{-proj}$

**shows**  $tf' \ (tf' \ s) = s$

*<proof>*

**definition** *tf'' where*

$tf'' \ g \ s = tf' \ (tf \ g \ s)$

**lemma** *remove-sym*:

**assumes**  $\text{gluing } \{((x, y), l)\} \in e\text{-proj}$   $\text{gluing } \{(g \ (x, y), l)\} \in e\text{-proj}$   $g \in \text{symmetries}$

**shows**  $\text{gluing } \{(g \ (x, y), l)\} = tf'' \ (\tau \circ g) \ (\text{gluing } \{((x, y), l)\})$

*<proof>*

**lemma** *remove-add-sym*:

**assumes**  $p \in e\text{-proj}$   $q \in e\text{-proj}$   $g \in \text{rotations}$

**shows**  $\text{proj-addition } (tf'' \ g \ p) \ q = tf'' \ g \ (\text{proj-addition } p \ q)$

*<proof>*

**lemma** *tf''-preserv-e-proj*:

**assumes**  $\text{gluing } \{((x, y), l)\} \in e\text{-proj}$   $r \in \text{rotations}$

**shows**  $tf'' \ r \ (\text{gluing } \{((x, y), l)\}) \in e\text{-proj}$

*<proof>*

**lemma** *tf'-injective*:

**assumes**  $c1 \in e\text{-proj}$   $c2 \in e\text{-proj}$

**assumes**  $tf' \ c1 = tf' \ c2$

**shows**  $c1 = c2$

*<proof>*

## 4.2 Associativities

**lemma** *add-add-add-add-assoc*:

**assumes**  $(x1,y1) \in e'\text{-aff}$   $(x2,y2) \in e'\text{-aff}$   $(x3,y3) \in e'\text{-aff}$   
**assumes**  $\text{delta } x1 \ y1 \ x2 \ y2 \neq 0$   $\text{delta } x2 \ y2 \ x3 \ y3 \neq 0$   
 $\text{delta } (\text{fst } (\text{add } (x1,y1) \ (x2,y2))) \ (\text{snd } (\text{add } (x1,y1) \ (x2,y2))) \ x3 \ y3 \neq 0$   
 $\text{delta } x1 \ y1 \ (\text{fst } (\text{add } (x2,y2) \ (x3,y3))) \ (\text{snd } (\text{add } (x2,y2) \ (x3,y3))) \neq 0$   
**shows**  $\text{add } (\text{add } (x1,y1) \ (x2,y2)) \ (x3,y3) = \text{add } (x1,y1) \ (\text{add } (x2,y2) \ (x3,y3))$   
 $\langle \text{proof} \rangle$

**lemma** *fstI*:  $x = (y, z) \implies y = \text{fst } x$   
 $\langle \text{proof} \rangle$

**lemma** *sndI*:  $x = (y, z) \implies z = \text{snd } x$   
 $\langle \text{proof} \rangle$

$\langle \text{ML} \rangle$

**lemma** *add-ext-add-ext-assoc-points*:

**assumes**  $(x1,y1) \in e'\text{-aff}$   $(x2,y2) \in e'\text{-aff}$   $(x3,y3) \in e'\text{-aff}$   
**assumes**  $\text{delta}' \ x1 \ y1 \ x2 \ y2 \neq 0$   $\text{delta}' \ x2 \ y2 \ x3 \ y3 \neq 0$   
 $\text{delta } (\text{fst } (\text{ext-add } (x1,y1) \ (x2,y2))) \ (\text{snd } (\text{ext-add } (x1,y1) \ (x2,y2))) \ x3 \ y3 \neq 0$   
 $\text{delta } x1 \ y1 \ (\text{fst } (\text{ext-add } (x2,y2) \ (x3,y3))) \ (\text{snd } (\text{ext-add } (x2,y2) \ (x3,y3))) \neq 0$   
**shows**  $\text{add } (\text{ext-add } (x1,y1) \ (x2,y2)) \ (x3,y3) = \text{add } (x1,y1) \ (\text{ext-add } (x2,y2) \ (x3,y3))$   
 $\langle \text{proof} \rangle$

$\langle \text{ML} \rangle$

**lemma** *add-ext-add-add-assoc-points*:

**assumes**  $(x1,y1) \in e'\text{-aff}$   $(x2,y2) \in e'\text{-aff}$   $(x3,y3) \in e'\text{-aff}$   
**assumes**  $\text{delta}' \ x1 \ y1 \ x2 \ y2 \neq 0$   $\text{delta } x2 \ y2 \ x3 \ y3 \neq 0$   
 $\text{delta } (\text{fst } (\text{ext-add } (x1,y1) \ (x2,y2))) \ (\text{snd } (\text{ext-add } (x1,y1) \ (x2,y2))) \ x3 \ y3 \neq 0$   
 $\text{delta } x1 \ y1 \ (\text{fst } (\text{add } (x2,y2) \ (x3,y3))) \ (\text{snd } (\text{add } (x2,y2) \ (x3,y3))) \neq 0$   
**shows**  $\text{add } (\text{ext-add } (x1,y1) \ (x2,y2)) \ (x3,y3) = \text{add } (x1,y1) \ (\text{add } (x2,y2) \ (x3,y3))$   
 $\langle \text{proof} \rangle$

$\langle \text{ML} \rangle$

**lemma** *add-add-ext-add-assoc-points*:

**assumes**  $(x1,y1) \in e'\text{-aff}$   $(x2,y2) \in e'\text{-aff}$   $(x3,y3) \in e'\text{-aff}$   
**assumes**  $\text{delta } x1 \ y1 \ x2 \ y2 \neq 0$   $\text{delta } x2 \ y2 \ x3 \ y3 \neq 0$   
 $\text{delta } (\text{fst } (\text{add } (x1,y1) \ (x2,y2))) \ (\text{snd } (\text{add } (x1,y1) \ (x2,y2))) \ x3 \ y3 \neq 0$

$\text{delta}' x1 y1 (\text{fst} (\text{add} (x2,y2) (x3,y3))) (\text{snd} (\text{add} (x2,y2) (x3,y3))) \neq 0$   
**shows**  $\text{add} (\text{add} (x1,y1) (x2,y2)) (x3,y3) = \text{ext-add} (x1,y1) (\text{add} (x2,y2) (x3,y3))$   
 <proof>

<ML>

**lemma** *add-add-add-ext-assoc-points*:

**assumes**  $(x1,y1) \in e'\text{-aff} (x2,y2) \in e'\text{-aff} (x3,y3) \in e'\text{-aff}$   
**assumes**  $\text{delta} x1 y1 x2 y2 \neq 0 \text{delta}' x2 y2 x3 y3 \neq 0$   
 $\text{delta} (\text{fst} (\text{add} (x1,y1) (x2,y2))) (\text{snd} (\text{add} (x1,y1) (x2,y2))) x3 y3 \neq 0$   
 $\text{delta} x1 y1 (\text{fst} (\text{ext-add} (x2,y2) (x3,y3))) (\text{snd} (\text{ext-add} (x2,y2) (x3,y3)))$   
 $\neq 0$   
**shows**  $\text{add} (\text{add} (x1,y1) (x2,y2)) (x3,y3) = \text{add} (x1,y1) (\text{ext-add} (x2,y2) (x3,y3))$   
 <proof>

<ML>

**lemma** *ext-add-add-ext-assoc-points*:

**assumes**  $(x1,y1) \in e'\text{-aff} (x2,y2) \in e'\text{-aff} (x3,y3) \in e'\text{-aff}$   
**assumes**  $\text{delta} x1 y1 x2 y2 \neq 0 \text{delta}' x2 y2 x3 y3 \neq 0$   
 $\text{delta}' (\text{fst} (\text{add} (x1,y1) (x2,y2))) (\text{snd} (\text{add} (x1,y1) (x2,y2))) x3 y3 \neq 0$   
 $\text{delta} x1 y1 (\text{fst} (\text{ext-add} (x2,y2) (x3,y3))) (\text{snd} (\text{ext-add} (x2,y2) (x3,y3)))$   
 $\neq 0$   
**shows**  $\text{ext-add} (\text{add} (x1,y1) (x2,y2)) (x3,y3) = \text{add} (x1,y1) (\text{ext-add} (x2,y2) (x3,y3))$   
 <proof>

<ML>

**lemma** *ext-add-add-add-assoc-points*:

**assumes**  $(x1,y1) \in e'\text{-aff} (x2,y2) \in e'\text{-aff} (x3,y3) \in e'\text{-aff}$   
**assumes**  $\text{delta} x1 y1 x2 y2 \neq 0 \text{delta} x2 y2 x3 y3 \neq 0$   
 $\text{delta}' (\text{fst} (\text{add} (x1,y1) (x2,y2))) (\text{snd} (\text{add} (x1,y1) (x2,y2))) x3 y3 \neq 0$   
 $\text{delta} x1 y1 (\text{fst} (\text{add} (x2,y2) (x3,y3))) (\text{snd} (\text{add} (x2,y2) (x3,y3))) \neq 0$   
**shows**  $\text{ext-add} (\text{add} (x1,y1) (x2,y2)) (x3,y3) = \text{add} (x1,y1) (\text{add} (x2,y2) (x3,y3))$   
 <proof>

<ML>

**lemma** *ext-ext-add-add-assoc-points*:

**assumes**  $(x1,y1) \in e'\text{-aff} (x2,y2) \in e'\text{-aff} (x3,y3) \in e'\text{-aff}$   
**assumes**  $\text{delta}' x1 y1 x2 y2 \neq 0 \text{delta} x2 y2 x3 y3 \neq 0$   
 $\text{delta}' (\text{fst} (\text{ext-add} (x1,y1) (x2,y2))) (\text{snd} (\text{ext-add} (x1,y1) (x2,y2))) x3$   
 $y3 \neq 0$   
 $\text{delta} x1 y1 (\text{fst} (\text{add} (x2,y2) (x3,y3))) (\text{snd} (\text{add} (x2,y2) (x3,y3))) \neq 0$   
**shows**  $\text{ext-add} (\text{ext-add} (x1,y1) (x2,y2)) (x3,y3) = \text{add} (x1,y1) (\text{add} (x2,y2) (x3,y3))$

$(x3, y3)$   
 $\langle proof \rangle$

$\langle ML \rangle$

**lemma** *ext-ext-add-ext-assoc-points*:

**assumes**  $(x1, y1) \in e'\text{-aff}$   $(x2, y2) \in e'\text{-aff}$   $(x3, y3) \in e'\text{-aff}$   
**assumes**  $\text{delta}' x1 y1 x2 y2 \neq 0$   $\text{delta}' x2 y2 x3 y3 \neq 0$   
 $\text{delta}' (\text{fst} (\text{ext-add} (x1, y1) (x2, y2))) (\text{snd} (\text{ext-add} (x1, y1) (x2, y2))) x3$   
 $y3 \neq 0$   
 $\text{delta} x1 y1 (\text{fst} (\text{ext-add} (x2, y2) (x3, y3))) (\text{snd} (\text{ext-add} (x2, y2) (x3, y3)))$   
 $\neq 0$   
**shows**  $\text{ext-add} (\text{ext-add} (x1, y1) (x2, y2)) (x3, y3) = \text{add} (x1, y1) (\text{ext-add} (x2, y2)$   
 $(x3, y3))$   
 $\langle proof \rangle$

$\langle ML \rangle$

### 4.3 Lemmas for associativity

**lemma** *cancellation-assoc*:

**assumes**  $\text{gluing} \text{ `` } \{(x1, y1), \text{False}\} \in e\text{-proj}$   
 $\text{gluing} \text{ `` } \{(x2, y2), \text{False}\} \in e\text{-proj}$   
 $\text{gluing} \text{ `` } \{(i (x2, y2), \text{False})\} \in e\text{-proj}$   
**shows**  $\text{proj-addition} (\text{proj-addition} (\text{gluing} \text{ `` } \{(x1, y1), \text{False}\}))$   
 $(\text{gluing} \text{ `` } \{(x2, y2), \text{False}\})) (\text{gluing} \text{ `` } \{(i (x2, y2),$   
 $\text{False}\})) =$   
 $\text{gluing} \text{ `` } \{(x1, y1), \text{False}\}$   
**(is**  $\text{proj-addition} (\text{proj-addition} ?g1 ?g2) ?g3 = ?g1)$   
 $\langle proof \rangle$

**lemma** *e'-aff-0-invariance*:

$((x, y), (x', y')) \in e'\text{-aff-0} \implies ((x', y'), (x, y)) \in e'\text{-aff-0}$   
 $\langle proof \rangle$

**lemma** *e'-aff-1-invariance*:

$((x, y), (x', y')) \in e'\text{-aff-1} \implies ((x', y'), (x, y)) \in e'\text{-aff-1}$   
 $\langle proof \rangle$

**lemma** *assoc-1*:

**assumes**  $\text{gluing} \text{ `` } \{(x1, y1), \text{False}\} \in e\text{-proj}$   
 $\text{gluing} \text{ `` } \{(x2, y2), \text{False}\} \in e\text{-proj}$   
 $\text{gluing} \text{ `` } \{(x3, y3), \text{False}\} \in e\text{-proj}$   
**assumes**  $a: g \in \text{symmetries} (x2, y2) = (g \circ i) (x1, y1)$   
**shows**  
 $\text{proj-addition} (\text{gluing} \text{ `` } \{(x1, y1), \text{False}\}) (\text{gluing} \text{ `` } \{(x2, y2), \text{False}\}) =$   
 $\text{tf''} (\tau \circ g) \{((1, 0), \text{False})\} (\text{is } \text{proj-addition} ?g1 ?g2 = -)$   
 $\text{proj-addition} (\text{proj-addition} (\text{gluing} \text{ `` } \{(x1, y1), \text{False}\})) (\text{gluing} \text{ `` } \{(x2, y2),$   
 $\text{False}\})) (\text{gluing} \text{ `` } \{(x3, y3), \text{False}\}) =$

$tf'' (\tau \circ g) (gluing \text{ `` } \{(x3, y3), False\})$   
 $proj\text{-}addition (gluing \text{ `` } \{(x1, y1), False\}) (proj\text{-}addition (gluing \text{ `` } \{(x2, y2),$   
 $False\})) (gluing \text{ `` } \{(x3, y3), False\})) =$   
 $tf'' (\tau \circ g) (gluing \text{ `` } \{(x3, y3), False\})$  (is proj-addition ?g1 (proj-addition  
?g2 ?g3) = -)  
<proof>

**lemma assoc-11:**

**assumes**  $gluing \text{ `` } \{(x1, y1), False\} \in e\text{-}proj$   
 $gluing \text{ `` } \{(x2, y2), False\} \in e\text{-}proj$   
 $gluing \text{ `` } \{(x3, y3), False\} \in e\text{-}proj$   
**assumes**  $a: g \in symmetries (x3, y3) = (g \circ i) (x2, y2)$   
**shows**  
 $proj\text{-}addition (proj\text{-}addition (gluing \text{ `` } \{(x1, y1), False\}) (gluing \text{ `` } \{(x2, y2),$   
 $False\})) (gluing \text{ `` } \{(x3, y3), False\}) =$   
 $proj\text{-}addition (gluing \text{ `` } \{(x1, y1), False\}) (proj\text{-}addition (gluing \text{ `` } \{(x2, y2),$   
 $False\})) (gluing \text{ `` } \{(x3, y3), False\}))$   
(is proj-addition (proj-addition ?g1 ?g2) ?g3 = -)  
<proof>

**lemma assoc-111-add:**

**assumes**  $gluing \text{ `` } \{(x1, y1), False\} \in e\text{-}proj$   
 $gluing \text{ `` } \{(x2, y2), False\} \in e\text{-}proj$   
 $gluing \text{ `` } \{(x3, y3), False\} \in e\text{-}proj$   
**assumes**  $22: g \in symmetries (x1, y1) = (g \circ i) (add (x2, y2) (x3, y3)) ((x2, y2),$   
 $x3, y3) \in e'\text{-}aff\text{-}0$   
**shows**  
 $proj\text{-}addition (proj\text{-}addition (gluing \text{ `` } \{(x1, y1), False\}) (gluing \text{ `` } \{(x2, y2),$   
 $False\})) (gluing \text{ `` } \{(x3, y3), False\}) =$   
 $proj\text{-}addition (gluing \text{ `` } \{(x1, y1), False\}) (proj\text{-}addition (gluing \text{ `` } \{(x2, y2),$   
 $False\})) (gluing \text{ `` } \{(x3, y3), False\}))$   
(is proj-addition (proj-addition ?g1 ?g2) ?g3 = -)  
<proof>

**lemma assoc-111-ext-add:**

**assumes**  $gluing \text{ `` } \{(x1, y1), False\} \in e\text{-}proj$   
 $gluing \text{ `` } \{(x2, y2), False\} \in e\text{-}proj$   
 $gluing \text{ `` } \{(x3, y3), False\} \in e\text{-}proj$   
**assumes**  $22: g \in symmetries (x1, y1) = (g \circ i) (ext\text{-}add (x2, y2) (x3, y3)) ((x2,$   
 $y2), x3, y3) \in e'\text{-}aff\text{-}1$   
**shows**  
 $proj\text{-}addition (proj\text{-}addition (gluing \text{ `` } \{(x1, y1), False\}) (gluing \text{ `` } \{(x2, y2),$   
 $False\})) (gluing \text{ `` } \{(x3, y3), False\}) =$   
 $proj\text{-}addition (gluing \text{ `` } \{(x1, y1), False\}) (proj\text{-}addition (gluing \text{ `` } \{(x2, y2),$   
 $False\})) (gluing \text{ `` } \{(x3, y3), False\}))$   
(is proj-addition (proj-addition ?g1 ?g2) ?g3 = -)  
<proof>

**lemma assoc-with-zeros:**

**assumes** *gluing* “  $\{(x1, y1), False\} \in e\text{-proj}$   
*gluing* “  $\{(x2, y2), False\} \in e\text{-proj}$   
*gluing* “  $\{(x3, y3), False\} \in e\text{-proj}$   
**shows** *proj-addition* (*proj-addition* (*gluing* “  $\{(x1, y1), False\}$ ) (*gluing* “  
 $\{(x2, y2), False\}$ ))  
 $(\text{gluing “ } \{(x3, y3), False\} =$   
*proj-addition* (*gluing* “  $\{(x1, y1), False\}$ )  
*proj-addition* (*gluing* “  $\{(x2, y2), False\}$ ) (*gluing* “  $\{(x3,$   
 $y3), False\}$ ))  
**(is** *proj-addition* (*proj-addition* ?g1 ?g2) ?g3 =  
*proj-addition* ?g1 (*proj-addition* ?g2 ?g3))  
 $\langle \text{proof} \rangle$

**lemma** *general-assoc*:

**assumes** *gluing* “  $\{(x1, y1), l\} \in e\text{-proj}$  *gluing* “  $\{(x2, y2), m\} \in e\text{-proj}$  *gluing*  
“  $\{(x3, y3), n\} \in e\text{-proj}$   
**shows** *proj-addition* (*proj-addition* (*gluing* “  $\{(x1, y1), l\}$ ) (*gluing* “  $\{(x2, y2),$   
 $m\}$ ))  
 $(\text{gluing “ } \{(x3, y3), n\} =$   
*proj-addition* (*gluing* “  $\{(x1, y1), l\}$ )  
*proj-addition* (*gluing* “  $\{(x2, y2), m\}$ ) (*gluing* “  $\{(x3, y3),$   
 $n\}$ ))  
 $\langle \text{proof} \rangle$

**lemma** *proj-assoc*:

**assumes**  $x \in e\text{-proj}$   $y \in e\text{-proj}$   $z \in e\text{-proj}$   
**shows** *proj-addition* (*proj-addition*  $x$   $y$ )  $z = \text{proj-addition } x$  (*proj-addition*  $y$   $z$ )  
 $\langle \text{proof} \rangle$

## 4.4 Group law

**theorem** *projective-group-law*:

**shows** *comm-group* ( $\text{carrier} = e\text{-proj}$ , *mult* = *proj-addition*, *one* =  $\{(1, 0), False\}$ )  
 $\langle \text{proof} \rangle$

**end**

**end**

## References

- [1] T. Hales and R. Raya. Formal proof of the group law for edwards elliptic curves. In N. Peltier and V. Sofronie-Stokkermans, editors, *Automated Reasoning*, pages 254–269, Cham, 2020. Springer International Publishing.