

# Poincaré Disc Model

Danijela Simić

Filip Marić

Pierre Boutry

June 17, 2024

## Abstract

We describe formalization of the Poincaré disc model of hyperbolic geometry within the Isabelle/HOL proof assistant. The model is defined within the extended complex plane (one dimensional complex projective space  $\mathbb{C}P^1$ ), formalized in the AFP entry “Complex Geometry” [6]. Points, lines, congruence of pairs of points, betweenness of triples of points, circles, and isometries are defined within the model. It is shown that the model satisfies all Tarski’s axioms except the Euclid’s axiom. It is shown that it satisfies its negation and the limiting parallels axiom (which proves it to be a model of hyperbolic geometry).

## Contents

<b>1</b>	<b>Introduction</b>	<b>3</b>
<b>2</b>	<b>Background theories</b>	<b>3</b>
2.1	Hyperbolic Functions . . . . .	3
<b>3</b>	<b>Tarski axioms</b>	<b>5</b>
<b>4</b>	<b>H-lines in the Poincaré model</b>	<b>6</b>
4.1	Definition and basic properties of h-lines . . . . .	6
4.1.1	Collinear points . . . . .	7
4.1.2	H-lines and inversion . . . . .	8
4.1.3	Classification of h-lines into Euclidean segments and circles . . . . .	8
4.1.4	Points on h-line . . . . .	9
4.1.5	H-line uniqueness . . . . .	12
4.1.6	H-isometries preserve h-lines . . . . .	15
4.1.7	Mapping h-lines to x-axis . . . . .	17
4.2	Construction of the h-line between the two given points . . . . .	18
4.2.1	Correctness of the construction . . . . .	21
4.2.2	Existence of h-lines . . . . .	29
4.2.3	Uniqueness of h-lines . . . . .	30
4.2.4	Some consequences of line uniqueness . . . . .	31
4.2.5	Transformations of constructed lines . . . . .	32
4.2.6	Collinear points and h-lines . . . . .	32
4.2.7	Points collinear with $\theta_h$ . . . . .	33
4.3	Ideal points of h-lines . . . . .	34
4.3.1	Calculation of ideal points . . . . .	35
4.3.2	Ideal points . . . . .	40
<b>5</b>	<b>H-distance in the Poincaré model</b>	<b>45</b>
5.1	Distance explicit formula . . . . .	49
5.2	Existence and uniqueness of points with a given distance . . . . .	55
5.3	Triangle inequality . . . . .	66
<b>6</b>	<b>H-circles in the Poincaré model</b>	<b>68</b>
6.1	Intersection of circles in special positions . . . . .	73
6.2	Congruent triangles . . . . .	75

<b>7</b>	<b>H-betweenness in the Poincaré model</b>	<b>78</b>
7.1	H-betweenness expressed by a cross-ratio . . . . .	78
7.1.1	H-betweenness is preserved by h-isometries . . . . .	78
7.1.2	Some elementary properties of h-betweenness . . . . .	79
7.1.3	H-betweenness and h-collinearity . . . . .	79
7.1.4	H-betweenness on Euclidean segments . . . . .	80
7.1.5	H-betweenness and h-collinearity . . . . .	87
7.2	Some properties of betweenness . . . . .	88
7.3	Poincare between - sum distances . . . . .	91
7.4	Some more properties of h-betweenness. . . . .	95
<b>8</b>	<b>Intersection of h-lines with the x-axis in the Poincaré model</b>	<b>97</b>
8.1	Betweenness of x-axis intersection . . . . .	97
8.2	Check if an h-line intersects the x-axis . . . . .	98
8.3	Check if a Poincaré line intersects the y-axis . . . . .	101
8.4	Intersection point of a Poincaré line with the x-axis in the unit disc . . . . .	102
8.5	Check if an h-line intersects the positive part of the x-axis . . . . .	106
8.6	Check if an h-line intersects the positive part of the y-axis . . . . .	110
8.7	Position of the intersection point in the unit disc . . . . .	111
8.8	Ideal points and x-axis intersection . . . . .	114
<b>9</b>	<b>H-perpendicular h-lines in the Poincaré model</b>	<b>115</b>
<b>10</b>	<b>Poincaré disc model types</b>	<b>125</b>
10.1	H-points . . . . .	125
10.2	H-lines . . . . .	126
10.3	H-collinearity . . . . .	127
10.4	H-isometries . . . . .	127
10.5	H-distance and h-congruence . . . . .	127
10.6	H-betweenness . . . . .	128
<b>11</b>	<b>Poincaré model satisfies Tarski axioms</b>	<b>128</b>
11.1	Pasch axiom . . . . .	128
11.2	Segment construction axiom . . . . .	140
11.3	Five segment axiom . . . . .	141
11.4	Upper dimension axiom . . . . .	142
11.5	Lower dimension axiom . . . . .	144
11.6	Negated Euclidean axiom . . . . .	144
11.7	Continuity axiom . . . . .	151
11.8	Limiting parallels axiom . . . . .	158
11.9	Interpretation of locales . . . . .	171

# 1 Introduction

Poincaré disc is a model of hyperbolic geometry. That fact has been a mathematical folklore for more than 100 years. However, up to the best of our knowledge, fully precise, formal proofs of this fact are lacking. In this paper we present a formalization of the Poincaré disc model in Isabelle/HOL, introduce its basic notions (h-points, h-lines, h-congruence, h-isometries, h-betweenness) and prove that it models Tarski's axioms except for Euclid's axiom. We show that it satisfies the negation of Euclid's axiom, and, moreover, the existence of limiting parallels axiom. The model is defined within the extended complex plane, which has been described quite precisely by Schwerdfeger [8] and formalized in the previous work of the first two authors [5].

**Related work.** In 1840 Lobachevsky [3] published developments about non-Euclidean geometry. Hyperbolic geometry is studied through many of its models. The concept of a projective disc model was introduced by Klein while Poincaré investigated the half-plane model proposed by Liouville and Beltrami and primarily studied the isometries of the hyperbolic plane that preserve orientation. In this paper, we focus on the formalization of the latter.

Regarding non-Euclidean geometry, Makarios showed the independence of Euclid's axiom [4]. He did so by formalizing that the Klein–Beltrami model is a model of Tarski's axioms at the exception of Euclid's axiom. Later Coghetto formalized the Klein–Beltrami model within Mizar [1, 2].

## 2 Background theories

### 2.1 Hyperbolic Functions

In this section hyperbolic cosine and hyperbolic sine functions are introduced and some of their properties needed for further development are proved.

```
theory Hyperbolic-Functions
  imports Complex-Main Complex-Geometry.More-Complex
begin

lemma cosh-arcosh [simp]:
  fixes x :: real
  assumes x ≥ 1
  shows cosh (arcosh x) = x
proof-
  from assms
  have **: x + sqrt(x2 - 1) ≥ 1
    by (smt one-le-power real-sqrt-ge-zero)
  hence *: x + sqrt(x2 - 1) ≠ 0
    by simp
  moreover
  have sqrt (x2 - 1) + 1 / (x + sqrt (x2 - 1)) = x (is ?lhs = x)
proof-
  have ?lhs = (x*sqrt(x2 - 1) + x2) / (x + sqrt(x2 - 1))
    using * ⟨x ≥ 1⟩
    by (subst add-divide-eq-iff, simp, simp add: field-simps)
  also have ... = x * (sqrt(x2 - 1) + x) / (x + sqrt(x2 - 1))
    by (simp add: field-simps power2-eq-square)
  finally
  show ?thesis
    using nonzero-mult-div-cancel-right[OF *, of x]
    by (simp add: field-simps)
qed
thus ?thesis
  using arcosh-real-def[OF assms(1)]
  unfolding cosh-def
  using ln-div[of 1, symmetric] **
  by auto
qed
```

```
lemma arcosh-ge-0 [simp]:
  fixes x::real
```

**assumes**  $x \geq 1$   
**shows**  $\text{arcosh } x \geq 0$   
**by** (*smt arcosh-def assms ln-ge-zero powr-ge-pzero*)

**lemma** *arcosh-eq-0-iff*:  
**fixes**  $x::\text{real}$   
**assumes**  $x \geq 1$   
**shows**  $\text{arcosh } x = 0 \iff x = 1$   
**using** *assms*  
**using** *cosh-arcosh* **by** *fastforce*

**lemma** *arcosh-eq-iff*:  
**fixes**  $x y::\text{real}$   
**assumes**  $x \geq 1 y \geq 1$   
**shows**  $\text{arcosh } x = \text{arcosh } y \iff x = y$   
**using** *assms*  
**using** *cosh-arcosh* **by** *fastforce*

**lemma** *cosh-gt-1 [simp]*:  
**fixes**  $x::\text{real}$   
**assumes**  $x > 0$   
**shows**  $\cosh x > 1$   
**using** *assms cosh-real-strict-mono* **by** *force*

**lemma** *cosh-eq-iff*:  
**fixes**  $x y::\text{real}$   
**assumes**  $x \geq 0 y \geq 0$   
**shows**  $\cosh x = \cosh y \iff x = y$   
**by** (*simp add: assms(1) assms(2)*)

**lemma** *arcosh-mono*:  
**fixes**  $x y::\text{real}$   
**assumes**  $x \geq 1 y \geq 1$   
**shows**  $\text{arcosh } x \geq \text{arcosh } y \iff x \geq y$   
**using** *assms*  
**by** (*smt arcosh-ge-0 cosh-arcosh cosh-real-nonneg-less-iff*)

**lemma** *arcosh-add*:  
**fixes**  $x y::\text{real}$   
**assumes**  $x \geq 1 y \geq 1$   
**shows**  $\text{arcosh } x + \text{arcosh } y = \text{arcosh } (x*y + \text{sqrt}((x^2 - 1)*(y^2 - 1)))$

**proof** –  
**have**  $\text{sqrt}((x^2 - 1) * (y^2 - 1)) \geq 0$   
**using** *assms*  
**by** *simp*  
**moreover**  
**have**  $x * y \geq 1$   
**using** *assms*  
**by** (*smt mult-le-cancel-left1*)  
**ultimately**  
**have**  $x * y + \text{sqrt}((x^2 - 1) * (y^2 - 1)) \geq 1$   
**by** *linarith*  
**hence**  $1: 0 \leq (x * y + \text{sqrt}((x^2 - 1) * (y^2 - 1)))^2 - 1$   
**by** *simp*

**have**  $2: x * \text{sqrt}(y^2 - 1) + y * \text{sqrt}(x^2 - 1) \geq 0$   
**using** *assms*  
**by** *simp*

**have**  $(x*\text{sqrt}(y^2 - 1)+y*\text{sqrt}(x^2 - 1))^2 = (\text{sqrt}((x*y+\text{sqrt}((x^2-1)*(y^2-1)))^2 - 1))^2$   
**using** *assms*

**proof** (*subst real-sqrt-pow2*)

```

show  $0 \leq (x * y + \text{sqrt}((x^2 - 1) * (y^2 - 1)))^2 - 1$ 
  by fact
next
have  $(x * \text{sqrt}(y^2 - 1))^2 = x^2 * (y^2 - 1)$ 
  using assms
  apply (subst power-mult-distrib)
  apply (subst real-sqrt-pow2, simp-all)
  done
moreover
have  $(y * \text{sqrt}(x^2 - 1))^2 = y^2 * (x^2 - 1)$ 
  using assms
  apply (subst power-mult-distrib)
  apply (subst real-sqrt-pow2, simp-all)
  done
ultimately show  $(x * \text{sqrt}(y^2 - 1) + y * \text{sqrt}(x^2 - 1))^2 = (x * y + \text{sqrt}((x^2 - 1) * (y^2 - 1)))^2 - 1$ 
  using assms
  unfolding power2-sum
  apply (simp add: real-sqrt-mult power-mult-distrib)
  apply (simp add: field-simps)
  done
qed
hence  $\text{sqrt}((x * y + \text{sqrt}((x^2 - 1) * (y^2 - 1)))^2 - 1) = x * \text{sqrt}(y^2 - 1) + y * \text{sqrt}(x^2 - 1)$ 
  using power2-eq-iff-nonneg[OF 2 real-sqrt-ge-zero[OF 1]]
  by simp
thus ?thesis
  using assms
  apply (subst arcosh-real-def[OF assms(1)])
  apply (subst arcosh-real-def[OF assms(2)])
  apply (subst arcosh-real-def[OF **])
  apply (subst ln-mult[symmetric])
  apply (smt one-le-power real-sqrt-ge-zero)
  apply (smt one-le-power real-sqrt-ge-zero)
  apply (simp add: real-sqrt-mult)
  apply (simp add: field-simps)
  done
qed

lemma arcosh-double:
  fixes  $x :: \text{real}$ 
  assumes  $x \geq 1$ 
  shows  $2 * \text{arcosh } x = \text{arcosh}(2 * x^2 - 1)$ 
  by (smt arcosh-add arcosh-mono assms one-power2 power2-eq-square real-sqrt-abs)

```

end

### 3 Tarski axioms

In this section we introduce axioms of Tarski [7] through a series of locales.

```

theory Tarski
imports Main
begin

```

The first locale assumes all Tarski axioms except for the Euclid's axiom and the continuity axiom and corresponds to absolute geometry.

```

locale TarskiAbsolute =
  fixes cong :: ' $p \Rightarrow 'p \Rightarrow 'p \Rightarrow 'p \Rightarrow \text{bool}$ '
  fixes betw :: ' $'p \Rightarrow 'p \Rightarrow 'p \Rightarrow \text{bool}$ '
  assumes cong-reflexive:  $\text{cong } x \ y \ y \ x$ 
  assumes cong-transitive:  $\text{cong } x \ y \ z \ u \wedge \text{cong } x \ y \ v \ w \longrightarrow \text{cong } z \ u \ v \ w$ 
  assumes cong-identity:  $\text{cong } x \ y \ z \ z \longrightarrow x = y$ 
  assumes segment-construction:  $\exists z. \text{betw } x \ y \ z \wedge \text{cong } y \ z \ a \ b$ 
  assumes five-segment:  $x \neq y \wedge \text{betw } x \ y \ z \wedge \text{betw } x' \ y' \ z' \wedge \text{cong } x \ y \ x' \ y' \wedge \text{cong } y \ z \ y' \ z' \wedge \text{cong } x \ u \ x' \ u' \wedge \text{cong } y \ u \ y' \ u' \longrightarrow \text{cong } z \ u \ z' \ u'$ 
  assumes betw-identity:  $\text{betw } x \ y \ x \longrightarrow x = y$ 

```

**assumes** *Pasch*:  $betw\ x\ u\ z \wedge betw\ y\ v\ z \longrightarrow (\exists\ a.\ betw\ u\ a\ y \wedge betw\ x\ a\ v)$   
**assumes** *lower-dimension*:  $\exists\ a.\ \exists\ b.\ \exists\ c.\ \neg\ betw\ a\ b\ c \wedge \neg\ betw\ b\ c\ a \wedge \neg\ betw\ c\ a\ b$   
**assumes** *upper-dimension*:  $cong\ x\ u\ x\ v \wedge cong\ y\ u\ y\ v \wedge cong\ z\ u\ z\ v \wedge u \neq v \longrightarrow betw\ x\ y\ z \vee betw\ y\ z\ x \vee betw\ z\ x\ y$   
**begin**

The following definitions are used to specify axioms in the following locales.

Point  $p$  is on line  $ab$ .

**definition** *on-line where*

*on-line*  $p\ a\ b \longleftrightarrow betw\ p\ a\ b \vee betw\ a\ p\ b \vee betw\ a\ b\ p$

Point  $p$  is on ray  $ab$ .

**definition** *on-ray where*

*on-ray*  $p\ a\ b \longleftrightarrow betw\ a\ p\ b \vee betw\ a\ b\ p$

Point  $p$  is inside angle  $abc$ .

**definition** *in-angle where*

*in-angle*  $p\ a\ b\ c \longleftrightarrow b \neq a \wedge b \neq c \wedge p \neq b \wedge (\exists\ x.\ betw\ a\ x\ c \wedge x \neq a \wedge x \neq c \wedge on-ray\ p\ b\ x)$

Ray  $ra$  meets the line  $l_a l_b$ .

**definition** *ray-meets-line where*

*ray-meets-line*  $ra\ rb\ la\ lb \longleftrightarrow (\exists\ x.\ on-ray\ x\ ra\ rb \wedge on-line\ x\ la\ lb)$

**end**

The second locale adds the negation of Euclid's axiom and limiting parallels and corresponds to hyperbolic geometry.

**locale** *TarskiHyperbolic* = *TarskiAbsolute* +

**assumes** *euclid-negation*:  $\exists\ a\ b\ c\ d\ t.\ betw\ a\ d\ t \wedge betw\ b\ d\ c \wedge a \neq d \wedge (\forall\ x\ y.\ betw\ a\ b\ x \wedge betw\ a\ c\ y \longrightarrow \neg\ betw\ x\ t\ y)$

**assumes** *limiting-parallels*:  $\neg\ on-line\ a\ x1\ x2 \implies$

$(\exists\ a1\ a2.\ \neg\ on-line\ a\ a1\ a2 \wedge$

$\neg\ ray-meets-line\ a\ a1\ x1\ x2 \wedge$

$\neg\ ray-meets-line\ a\ a2\ x1\ x2 \wedge$

$(\forall\ a'.\ in-angle\ a'\ a1\ a2 \longrightarrow ray-meets-line\ a\ a'\ x1\ x2))$

The third locale adds the continuity axiom and corresponds to elementary hyperbolic geometry.

**locale** *ElementaryTarskiHyperbolic* = *TarskiHyperbolic* +

**assumes** *continuity*:  $\llbracket \exists\ a.\ \forall\ x.\ \forall\ y.\ \varphi\ x \wedge \psi\ y \longrightarrow betw\ a\ x\ y \rrbracket \implies \exists\ b.\ \forall\ x.\ \forall\ y.\ \varphi\ x \wedge \psi\ y \longrightarrow betw\ x\ b\ y$

**end**

## 4 H-lines in the Poincaré model

**theory** *Poincare-Lines*

**imports** *Complex-Geometry.Unit-Circle-Preserving-Moebius* *Complex-Geometry.Circlines-Angle*

**begin**

### 4.1 Definition and basic properties of h-lines

H-lines in the Poincaré model are either line segments passing through the origin or segments (within the unit disc) of circles that are perpendicular to the unit circle. Algebraically these are circlines that are represented by Hermitean matrices of the form

$$H = \begin{pmatrix} A & B \\ \overline{B} & A \end{pmatrix},$$

for  $A \in \mathbb{R}$ , and  $B \in \mathbb{C}$ , and  $|B|^2 > A^2$ , where the circline equation is the usual one:  $z^* H z = 0$ , for homogenous coordinates  $z$ .

**definition** *is-poincare-line-cmat* :: *complex-mat*  $\Rightarrow$  *bool* **where**

[simp]: *is-poincare-line-cmat*  $H \longleftrightarrow$

$(let\ (A,\ B,\ C,\ D) = H$

$in\ hermitean\ (A,\ B,\ C,\ D) \wedge A = D \wedge (cmod\ B)^2 > (cmod\ A)^2)$

**lift-definition** *is-poincare-line-clmat* :: *circline-mat*  $\Rightarrow$  *bool* **is** *is-poincare-line-clmat*  
**done**

We introduce the predicate that checks if a given complex matrix is a matrix of a h-line in the Poincaré model, and then by means of the lifting package lift it to the type of non-zero Hermitean matrices, and then to circlines (that are equivalence classes of such matrices).

**lift-definition** *is-poincare-line* :: *circline*  $\Rightarrow$  *bool* **is** *is-poincare-line-clmat*  
**proof** (*transfer*, *transfer*)  
**fix** *H1 H2* :: *complex-mat*  
**assume** *hh*: *hermitean H1*  $\wedge$  *H1*  $\neq$  *mat-zero* *hermitean H2*  $\wedge$  *H2*  $\neq$  *mat-zero*  
**assume** *circline-eq-cmat H1 H2*  
**thus** *is-poincare-line-clmat H1*  $\longleftrightarrow$  *is-poincare-line-clmat H2*  
**using** *hh*  
**by** (*cases H1*, *cases H2*) (*auto simp: norm-mult power-mult-distrib*)  
**qed**

**lemma** *is-poincare-line-mk-circline*:  
**assumes** (*A*, *B*, *C*, *D*)  $\in$  *hermitean-nonzero*  
**shows** *is-poincare-line (mk-circline A B C D)*  $\longleftrightarrow$  (*cmod B*)<sup>2</sup> > (*cmod A*)<sup>2</sup>  $\wedge$  *A = D*  
**using** *assms*  
**by** (*transfer*, *transfer*, *auto simp add: Let-def*)

Abstract characterisation of *is-poincare-line* predicate: H-lines in the Poincaré model are real circlines (circlines with the negative determinant) perpendicular to the unit circle.

**lemma** *is-poincare-line-iff*:  
**shows** *is-poincare-line H*  $\longleftrightarrow$  *circline-type H = -1*  $\wedge$  *perpendicular H unit-circle*  
**unfolding** *perpendicular-def*  
**proof** (*simp*, *transfer*, *transfer*)  
**fix** *H*  
**assume** *hh*: *hermitean H*  $\wedge$  *H*  $\neq$  *mat-zero*  
**obtain** *A B C D* **where** \*: *H = (A, B, C, D)*  
**by** (*cases H*, *auto*)  
**have** \*\*: *is-real A is-real D C = cnj B*  
**using** *hh \* hermitean-elems*  
**by** *auto*  
**hence** (*Re A = Re D*  $\wedge$  *cmod A \* cmod A < cmod B \* cmod B*) =  
(*Re A \* Re D < Re B \* Re B + Im B \* Im B*  $\wedge$  (*Re D = Re A*  $\vee$  *Re A \* Re D = Re B \* Re B + Im B \* Im B*))  
**using** \*  
**by** (*smt cmod-power2 power2-eq-square zero-power2*)  
**thus** *is-poincare-line-clmat H*  $\longleftrightarrow$   
*circline-type-cmat H = -1*  $\wedge$  *cos-angle-cmat (of-circline-cmat H) unit-circle-cmat = 0*  
**using** \* \*\*  
**by** (*auto simp add: sgn-1-neg complex-eq-if-Re-eq cmod-square power2-eq-square simp del: pos-oriented-cmat-def*)  
**qed**

The *x-axis* is an h-line.

**lemma** *is-poincare-line-x-axis* [*simp*]:  
**shows** *is-poincare-line x-axis*  
**by** (*transfer*, *transfer*) (*auto simp add: hermitean-def mat-adj-def mat-cnj-def*)

The *unit-circle* is not an h-line.

**lemma** *not-is-poincare-line-unit-circle* [*simp*]:  
**shows**  $\neg$  *is-poincare-line unit-circle*  
**by** (*transfer*, *transfer*, *simp*)

#### 4.1.1 Collinear points

Points are collinear if they all belong to an h-line.

**definition** *poincare-collinear* :: *complex-homo set*  $\Rightarrow$  *bool* **where**  
*poincare-collinear S*  $\longleftrightarrow$  ( $\exists$  *p*. *is-poincare-line p*  $\wedge$  *S*  $\subseteq$  *circline-set p*)

### 4.1.2 H-lines and inversion

Every h-line in the Poincaré model contains the inverse (wrt. the unit circle) of each of its points (note that at most one of them belongs to the unit disc).

**lemma** *is-poincare-line-inverse-point*:

**assumes** *is-poincare-line*  $H$   $u \in \text{circline-set } H$

**shows** *inversion*  $u \in \text{circline-set } H$

**using** *assms*

**unfolding** *is-poincare-line-iff circline-set-def perpendicular-def inversion-def*

**apply** *simp*

**proof** (*transfer, transfer*)

**fix**  $u H$

**assume** *hh: hermitean*  $H \wedge H \neq \text{mat-zero } u \neq \text{vec-zero}$  **and**

*aa: circline-type-cmat*  $H = -1 \wedge \text{cos-angle-cmat (of-circline-cmat } H) \text{ unit-circle-cmat} = 0$  *on-circline-cmat-cvec*

$H$   $u$

**obtain**  $A B C D u1 u2$  **where**  $*$ :  $H = (A, B, C, D)$   $u = (u1, u2)$

**by** (*cases*  $H$ , *cases*  $u$ , *auto*)

**have** *is-real*  $A$  *is-real*  $D$   $C = \text{cnj } B$

**using**  $*$  *hh hermitean-elems*

**by** *auto*

**moreover**

**have**  $A = D$

**using** *aa(1) \* <is-real A> <is-real D>*

**by** (*auto simp del: pos-oriented-cmat-def simp add: complex.expand split: if-split-asm*)

**thus** *on-circline-cmat-cvec*  $H$  (*conjugate-cvec (reciprocal-cvec*  $u)$ )

**using** *aa(2) \**

**by** (*simp add: vec-cnj-def field-simps*)

**qed**

Every h-line in the Poincaré model and is invariant under unit circle inversion.

**lemma** *circline-inversion-poincare-line*:

**assumes** *is-poincare-line*  $H$

**shows** *circline-inversion*  $H = H$

**proof**–

**obtain**  $u v w$  **where**  $*$ :  $u \neq v$   $v \neq w$   $u \neq w$   $\{u, v, w\} \subseteq \text{circline-set } H$

**using** *assms is-poincare-line-iff[of H]*

**using** *circline-type-neg-card-gt3[of H]*

**by** *auto*

**hence**  $\{\text{inversion } u, \text{inversion } v, \text{inversion } w\} \subseteq \text{circline-set (circline-inversion } H)$

$\{\text{inversion } u, \text{inversion } v, \text{inversion } w\} \subseteq \text{circline-set } H$

**using** *is-poincare-line-inverse-point[OF assms]*

**by** *auto*

**thus** *?thesis*

**using**  $*$  *unique-circline-set[of inversion u inversion v inversion w]*

**by** (*metis insert-subset inversion-involution*)

**qed**

### 4.1.3 Classification of h-lines into Euclidean segments and circles

If an h-line contains zero, than it also contains infinity (the inverse point of zero) and is by definition an Euclidean line.

**lemma** *is-poincare-line-trough-zero-trough-infty [simp]*:

**assumes** *is-poincare-line*  $l$  **and**  $0_h \in \text{circline-set } l$

**shows**  $\infty_h \in \text{circline-set } l$

**using** *is-poincare-line-inverse-point[OF assms]*

**by** *simp*

**lemma** *is-poincare-line-trough-zero-is-line*:

**assumes** *is-poincare-line*  $l$  **and**  $0_h \in \text{circline-set } l$

**shows** *is-line*  $l$

**using** *assms*

**using** *inf-in-circline-set is-poincare-line-trough-zero-trough-infty*

**by** *blast*

If an h-line does not contain zero, than it also does not contain infinity (the inverse point of zero) and is by definition an Euclidean circle.



**lemma** *is-poincare-line-not-trough-zero-not-trough-infty* [simp]:  
**assumes** *is-poincare-line l*  
**assumes**  $0_h \notin \text{circline-set } l$   
**shows**  $\infty_h \notin \text{circline-set } l$   
**using** *assms*  
**using** *is-poincare-line-inverse-point[OF assms(1), of  $\infty_h$ ]*  
**by** *auto*

**lemma** *is-poincare-line-not-trough-zero-is-circle*:  
**assumes** *is-poincare-line l*  $0_h \notin \text{circline-set } l$   
**shows** *is-circle l*  
**using** *assms*  
**using** *inf-in-circline-set is-poincare-line-not-trough-zero-not-trough-infty*  
**by** *auto*

#### 4.1.4 Points on h-line

Each h-line in the Poincaré model contains at least two different points within the unit disc.

First we prove an auxiliary lemma.

**lemma** *ex-is-poincare-line-points'*:  
**assumes** *i12: i1 ∈ circline-set H ∩ unit-circle-set*  
 $i2 \in \text{circline-set } H \cap \text{unit-circle-set}$   
 $i1 \neq i2$   
**assumes** *a: a ∈ circline-set H a ∉ unit-circle-set*  
**shows**  $\exists b. b \neq i1 \wedge b \neq i2 \wedge b \neq a \wedge b \neq \text{inversion } a \wedge b \in \text{circline-set } H$   
**proof** –

**have** *inversion a ∉ unit-circle-set*  
**using**  $\langle a \notin \text{unit-circle-set} \rangle$   
**unfolding** *unit-circle-set-def circline-set-def*  
**by** (*metis inversion-id-iff-on-unit-circle inversion-involution mem-Collect-eq*)

**have**  $a \neq \text{inversion } a$   
**using**  $\langle a \notin \text{unit-circle-set} \rangle$  *inversion-id-iff-on-unit-circle[of a]*  
**unfolding** *unit-circle-set-def circline-set-def*  
**by** *auto*

**have**  $a \neq i1 \ a \neq i2 \ \text{inversion } a \neq i1 \ \text{inversion } a \neq i2$   
**using** *assms*  $\langle \text{inversion } a \notin \text{unit-circle-set} \rangle$   
**by** *auto*

**then obtain b where** *cr2: cross-ratio b i1 a i2 = of-complex 2*  
**using**  $\langle i1 \neq i2 \rangle$   
**using** *ex-cross-ratio[of i1 a i2]*  
**by** *blast*

**have** *distinct-b: b ≠ i1 b ≠ i2 b ≠ a*  
**using**  $\langle i1 \neq i2 \rangle \langle a \neq i1 \rangle \langle a \neq i2 \rangle$   
**using** *ex1-cross-ratio[of i1 a i2]*  
**using** *cross-ratio-0[of i1 a i2] cross-ratio-1[of i1 a i2] cross-ratio-inf[of i1 i2 a]*  
**using** *cr2*  
**by** *auto*

**hence**  $b \in \text{circline-set } H$   
**using** *assms* *four-points-on-circline-iff-cross-ratio-real[of b i1 a i2] cr2*  
**using** *unique-circline-set[of i1 i2 a]*  
**by** *auto*

**moreover**

**have**  $b \neq \text{inversion } a$   
**proof** (*rule ccontr*)  
**assume**  $*$ ;  $\neg$  *?thesis*  
**have** *inversion i1 = i1 inversion i2 = i2*  
**using** *i12*  
**unfolding** *unit-circle-set-def*

```

  by auto
  hence cross-ratio (inversion a) i1 a i2 = cross-ratio a i1 (inversion a) i2
  using * cross-ratio-inversion[of i1 a i2 b] ⟨a ≠ i1⟩ ⟨a ≠ i2⟩ ⟨i1 ≠ i2⟩ ⟨b ≠ i1⟩
  using four-points-on-circline-iff-cross-ratio-real[of b i1 a i2]
  using i12 distinct-b conjugate-id-iff[of cross-ratio b i1 a i2]
  using i12 a ⟨b ∈ circline-set H⟩
  by auto
  hence cross-ratio (inversion a) i1 a i2 ≠ of-complex 2
  using cross-ratio-commute-13[of inversion a i1 a i2]
  using reciprocal-id-iff
  using of-complex-inj
  by force
  thus False
  using * cr2
  by simp
qed

```

```

ultimately
show ?thesis
  using assms ⟨b ≠ i1⟩ ⟨b ≠ i2⟩ ⟨b ≠ a⟩
  by auto
qed

```

Now we can prove the statement.

```

lemma ex-is-poincare-line-points:
  assumes is-poincare-line H
  shows ∃ u v. u ∈ unit-disc ∧ v ∈ unit-disc ∧ u ≠ v ∧ {u, v} ⊆ circline-set H
proof -

```

```

  obtain u v w where *: u ≠ v v ≠ w u ≠ w {u, v, w} ⊆ circline-set H
  using assms is-poincare-line-iff[of H]
  using circline-type-neg-card-gt3[of H]
  by auto

```

```

  have ¬ {u, v, w} ⊆ unit-circle-set
  using unique-circline-set[of u v w] *
  by (metis assms insert-subset not-is-poincare-line-unit-circle unit-circle-set-def)

```

```

  hence H ≠ unit-circle
  unfolding unit-circle-set-def
  using *
  by auto

```

```

show ?thesis
proof (cases (u ∈ unit-disc ∧ v ∈ unit-disc) ∨
  (u ∈ unit-disc ∧ w ∈ unit-disc) ∨
  (v ∈ unit-disc ∧ w ∈ unit-disc))
  case True
  thus ?thesis
  using *
  by auto
next
  case False

```

```

  have ∃ a b. a ≠ b ∧ a ≠ inversion b ∧ a ∈ circline-set H ∧ b ∈ circline-set H ∧ a ∉ unit-circle-set ∧ b ∉
unit-circle-set

```

```

  proof (cases (u ∈ unit-circle-set ∧ v ∈ unit-circle-set) ∨
  (u ∈ unit-circle-set ∧ w ∈ unit-circle-set) ∨
  (v ∈ unit-circle-set ∧ w ∈ unit-circle-set))
  case True
  then obtain i1 i2 a where *:
    i1 ∈ unit-circle-set ∩ circline-set H i2 ∈ unit-circle-set ∩ circline-set H
    a ∈ circline-set H a ∉ unit-circle-set
    i1 ≠ i2 i1 ≠ a i2 ≠ a
  using * ⟨¬ {u, v, w} ⊆ unit-circle-set⟩
  by auto
  then obtain b where b ∈ circline-set H b ≠ i1 b ≠ i2 b ≠ a b ≠ inversion a

```

```

using ex-is-poincare-line-points'[of i1 H i2 a]
by blast

hence b ∉ unit-circle-set
using * ⟨H ≠ unit-circle⟩ unique-circline-set[of i1 i2 b]
unfolding unit-circle-set-def
by auto

thus ?thesis
using * ⟨b ∈ circline-set H⟩ ⟨b ≠ a⟩ ⟨b ≠ inversion a⟩
by auto
next
case False
then obtain f g h where
  *: f ≠ g f ∈ circline-set H f ∉ unit-circle-set
  g ∈ circline-set H g ∉ unit-circle-set
  h ∈ circline-set H h ≠ f h ≠ g
  using *
  by auto
show ?thesis
proof (cases f = inversion g)
  case False
  thus ?thesis
  using *
  by auto
next
case True
show ?thesis
proof (cases h ∈ unit-circle-set)
  case False
  thus ?thesis
  using * ⟨f = inversion g⟩
  by auto
next
case True
obtain m where cr2: cross-ratio m h f g = of-complex 2
  using ex-cross-ratio[of h f g] * ⟨f ≠ g⟩ ⟨h ≠ f⟩ ⟨h ≠ g⟩
  by auto
hence m ≠ h m ≠ f m ≠ g
  using ⟨h ≠ f⟩ ⟨h ≠ g⟩ ⟨f ≠ g⟩
  using ex1-cross-ratio[of h f g]
  using cross-ratio-0[of h f g] cross-ratio-1[of h f g] cross-ratio-inf[of h g f]
  using cr2
  by auto
hence m ∈ circline-set H
  using four-points-on-circline-iff-cross-ratio-real[of m h f g] cr2
  using ⟨h ≠ f⟩ ⟨h ≠ g⟩ ⟨f ≠ g⟩ *
  using unique-circline-set[of h f g]
  by auto

show ?thesis
proof (cases m ∈ unit-circle-set)
  case False
  thus ?thesis
  using ⟨m ≠ f⟩ ⟨m ≠ g⟩ ⟨f = inversion g⟩ * ⟨m ∈ circline-set H⟩
  by auto
next
case True
then obtain n where n ≠ h n ≠ m n ≠ f n ≠ inversion f n ∈ circline-set H
  using ex-is-poincare-line-points'[of h H m f] * ⟨m ∈ circline-set H⟩ ⟨h ∈ unit-circle-set⟩ ⟨m ≠ h⟩
  by auto
hence n ∉ unit-circle-set
  using * ⟨H ≠ unit-circle⟩ unique-circline-set[of m n h]
  using ⟨m ≠ h⟩ ⟨m ∈ unit-circle-set⟩ ⟨h ∈ unit-circle-set⟩ ⟨m ∈ circline-set H⟩
  unfolding unit-circle-set-def
  by auto

```

```

      thus ?thesis
      using * ⟨n ∈ circline-set H⟩ ⟨n ≠ f⟩ ⟨n ≠ inversion f⟩
      by auto
    qed
  qed
  qed
  then obtain a b where ab: a ≠ b a ≠ inversion b a ∈ circline-set H b ∈ circline-set H a ∉ unit-circle-set b ∉
unit-circle-set
  by blast
  have ∀ x. x ∈ circline-set H ∧ x ∉ unit-circle-set → (∃ x'. x' ∈ circline-set H ∩ unit-disc ∧ (x' = x ∨ x' =
inversion x))
  proof safe
    fix x
    assume x: x ∈ circline-set H x ∉ unit-circle-set
    show ∃ x'. x' ∈ circline-set H ∩ unit-disc ∧ (x' = x ∨ x' = inversion x)
    proof (cases x ∈ unit-disc)
      case True
      thus ?thesis
      using x
      by auto
    next
      case False
      hence x ∈ unit-disc-compl
      using x in-on-out-univ[of ounit-circle]
      unfolding unit-circle-set-def unit-disc-def unit-disc-compl-def
      by auto
      hence inversion x ∈ unit-disc
      using inversion-unit-disc-compl
      by blast
      thus ?thesis
      using is-poincare-line-inverse-point[OF assms, of x] x
      by auto
    qed
  qed
  then obtain a' b' where
  *: a' ∈ circline-set H a' ∈ unit-disc b' ∈ circline-set H b' ∈ unit-disc and
  **: a' = a ∨ a' = inversion a b' = b ∨ b' = inversion b
  using ab
  by blast
  have a' ≠ b'
  using ⟨a ≠ b⟩ ⟨a ≠ inversion b⟩ ** *
  by (metis inversion-involution)
  thus ?thesis
  using *
  by auto
  qed
  qed

```

#### 4.1.5 H-line uniqueness

There is no more than one h-line that contains two different h-points (in the disc).

lemma *unique-is-poincare-line*:

```

assumes in-disc: u ∈ unit-disc v ∈ unit-disc u ≠ v
assumes pl: is-poincare-line l1 is-poincare-line l2
assumes on-l: {u, v} ⊆ circline-set l1 ∩ circline-set l2
shows l1 = l2

```

proof –

```

have u ≠ inversion u v ≠ inversion u
  using in-disc
  using inversion-noteq-unit-disc[of u v]
  using inversion-noteq-unit-disc[of u u]
  by auto
thus ?thesis
using on-l

```

```

using unique-circline-set[of u inversion u v] ⟨u ≠ v⟩
using is-poincare-line-inverse-point[of l1 u]
using is-poincare-line-inverse-point[of l2 u]
using pl
by auto
qed

```

For the rest of our formalization it is often useful to consider points on h-lines that are not within the unit disc. Many lemmas in the rest of this section will have such generalizations.

There is no more than one h-line that contains two different and not mutually inverse points (not necessary in the unit disc).

**lemma** *unique-is-poincare-line-general*:

```

assumes different: u ≠ v u ≠ inversion v
assumes pl: is-poincare-line l1 is-poincare-line l2
assumes on-l: {u, v} ⊆ circline-set l1 ∩ circline-set l2
shows l1 = l2
proof (cases u ≠ inversion u)
case True
thus ?thesis
using unique-circline-set[of u inversion u v]
using assms
using is-poincare-line-inverse-point by force

```

**next**

```

case False
show ?thesis
proof (cases v ≠ inversion v)
case True
thus ?thesis
using unique-circline-set[of u inversion v v]
using assms
using is-poincare-line-inverse-point by force

```

**next**

**case** False

**have** on-circline unit-circle u on-circline unit-circle v

```

using ⟨¬ u ≠ inversion u⟩ ⟨¬ v ≠ inversion v⟩
using inversion-id-iff-on-unit-circle
by fastforce+

```

**thus** ?thesis

```

using pl on-l ⟨u ≠ v⟩
unfolding circline-set-def
apply simp

```

**proof** (transfer, transfer, safe)

**fix** u1 u2 v1 v2 A1 B1 C1 D1 A2 B2 C2 D2 :: complex

**let** ?u = (u1, u2) **and** ?v = (v1, v2) **and** ?H1 = (A1, B1, C1, D1) **and** ?H2 = (A2, B2, C2, D2)

**assume** \*: ?u ≠ vec-zero ?v ≠ vec-zero

```

on-circline-cmat-cvec unit-circle-cmat ?u on-circline-cmat-cvec unit-circle-cmat ?v
is-poincare-line-cmat ?H1 is-poincare-line-cmat ?H2
hermitean ?H1 ?H1 ≠ mat-zero hermitean ?H2 ?H2 ≠ mat-zero
on-circline-cmat-cvec ?H1 ?u on-circline-cmat-cvec ?H1 ?v
on-circline-cmat-cvec ?H2 ?u on-circline-cmat-cvec ?H2 ?v
¬ (u1, u2) ≈v (v1, v2)

```

**have** \*\*: A1 = D1 A2 = D2 C1 = cnj B1 C2 = cnj B2 is-real A1 is-real A2

**using** ⟨is-poincare-line-cmat ?H1⟩ ⟨is-poincare-line-cmat ?H2⟩

**using** ⟨hermitean ?H1⟩ ⟨?H1 ≠ mat-zero⟩ ⟨hermitean ?H2⟩ ⟨?H2 ≠ mat-zero⟩

**using** hermitean-elems

**by** auto

**have** uv: u1 ≠ 0 u2 ≠ 0 v1 ≠ 0 v2 ≠ 0

**using** \*(1-4)

**by** (auto simp add: vec-cnj-def)

**have** u: cor ((Re (u1/u2))<sup>2</sup>) + cor ((Im (u1/u2))<sup>2</sup>) = 1

**using** ⟨on-circline-cmat-cvec unit-circle-cmat ?u⟩ uv

**apply** (subst of-real-add[symmetric])

**apply** (subst complex-mult-cnj[symmetric])  
**apply** (simp add: vec-cnj-def mult.commute)  
**done**

**have**  $v$ : cor ((Re (v1/v2))<sup>2</sup>) + cor ((Im (v1/v2))<sup>2</sup>) = 1  
**using** ‹on-circline-cmat-cvec unit-circle-cmat ?v› uv  
**apply** (subst of-real-add[symmetric])  
**apply** (subst complex-mult-cnj[symmetric])  
**apply** (simp add: vec-cnj-def mult.commute)  
**done**

**have**

$A1 * (cor ((Re (u1/u2))<sup>2</sup>) + cor ((Im (u1/u2))<sup>2</sup>) + 1) + cor (Re B1) * cor(2 * Re (u1/u2)) + cor (Im B1) * cor(2 * Im (u1/u2)) = 0$   
 $A2 * (cor ((Re (u1/u2))<sup>2</sup>) + cor ((Im (u1/u2))<sup>2</sup>) + 1) + cor (Re B2) * cor(2 * Re (u1/u2)) + cor (Im B2) * cor(2 * Im (u1/u2)) = 0$   
 $A1 * (cor ((Re (v1/v2))<sup>2</sup>) + cor ((Im (v1/v2))<sup>2</sup>) + 1) + cor (Re B1) * cor(2 * Re (v1/v2)) + cor (Im B1) * cor(2 * Im (v1/v2)) = 0$   
 $A2 * (cor ((Re (v1/v2))<sup>2</sup>) + cor ((Im (v1/v2))<sup>2</sup>) + 1) + cor (Re B2) * cor(2 * Re (v1/v2)) + cor (Im B2) * cor(2 * Im (v1/v2)) = 0$   
**using** circline-equation-quadratic-equation[of A1 u1/u2 B1 D1 Re (u1/u2) Im (u1 / u2) Re B1 Im B1]  
**using** circline-equation-quadratic-equation[of A2 u1/u2 B2 D2 Re (u1/u2) Im (u1 / u2) Re B2 Im B2]  
**using** circline-equation-quadratic-equation[of A1 v1/v2 B1 D1 Re (v1/v2) Im (v1 / v2) Re B1 Im B1]  
**using** circline-equation-quadratic-equation[of A2 v1/v2 B2 D2 Re (v1/v2) Im (v1 / v2) Re B2 Im B2]  
**using** ‹on-circline-cmat-cvec ?H1 ?u› ‹on-circline-cmat-cvec ?H2 ?u›  
**using** ‹on-circline-cmat-cvec ?H1 ?v› ‹on-circline-cmat-cvec ?H2 ?v›  
**using** \*\* uv  
**by** (simp-all add: vec-cnj-def field-simps)

**hence**

$A1 + cor (Re B1) * cor(Re (u1/u2)) + cor (Im B1) * cor(Im (u1/u2)) = 0$   
 $A1 + cor (Re B1) * cor(Re (v1/v2)) + cor (Im B1) * cor(Im (v1/v2)) = 0$   
 $A2 + cor (Re B2) * cor(Re (u1/u2)) + cor (Im B2) * cor(Im (u1/u2)) = 0$   
 $A2 + cor (Re B2) * cor(Re (v1/v2)) + cor (Im B2) * cor(Im (v1/v2)) = 0$   
**using** u v  
**by** simp-all algebra+

**hence**

$cor (Re A1 + Re B1 * Re (u1/u2) + Im B1 * Im (u1/u2)) = 0$   
 $cor (Re A2 + Re B2 * Re (u1/u2) + Im B2 * Im (u1/u2)) = 0$   
 $cor (Re A1 + Re B1 * Re (v1/v2) + Im B1 * Im (v1/v2)) = 0$   
 $cor (Re A2 + Re B2 * Re (v1/v2) + Im B2 * Im (v1/v2)) = 0$   
**using** ‹is-real A1› ‹is-real A2›  
**by** simp-all

**hence**

$Re A1 + Re B1 * Re (u1/u2) + Im B1 * Im (u1/u2) = 0$   
 $Re A1 + Re B1 * Re (v1/v2) + Im B1 * Im (v1/v2) = 0$   
 $Re A2 + Re B2 * Re (u1/u2) + Im B2 * Im (u1/u2) = 0$   
 $Re A2 + Re B2 * Re (v1/v2) + Im B2 * Im (v1/v2) = 0$   
**using** of-real-eq-0-iff  
**by** blast+

**moreover**

**have**  $Re(u1/u2) \neq Re(v1/v2) \vee Im(u1/u2) \neq Im(v1/v2)$   
**proof** (rule ccontr)  
**assume**  $\neg$  ?thesis  
**hence**  $u1/u2 = v1/v2$   
**using** complex-eqI **by** blast  
**thus** False  
**using** uv ‹ $\neg (u1, u2) \approx_v (v1, v2)$ ›  
**using** \*(1) \*(2) complex-cvec-eq-mix[OF \*(1) \*(2)]  
**by** (auto simp add: field-simps)  
**qed**

moreover

have  $Re A1 \neq 0 \vee Re B1 \neq 0 \vee Im B1 \neq 0$   
using  $\langle ?H1 \neq mat-zero \rangle **$   
by  $(metis\ complex-cnj-zero\ complex-of-real-Re\ mat-zero-def\ of-real-0)$

ultimately

obtain  $k$  where

$k: Re A2 = k * Re A1\ Re B2 = k * Re B1\ Im B2 = k * Im B1$

using  $linear-system-homogenous-3-2[of\ \lambda x\ y\ z.\ 1 * x + Re (u1 / u2) * y + Im (u1 / u2) * z\ 1\ Re (u1/u2)\ Im (u1/u2)]$

$\lambda x\ y\ z.\ 1 * x + Re (v1 / v2) * y + Im (v1 / v2) * z\ 1\ Re (v1/v2)\ Im (v1/v2)$   
 $Re A2\ Re B2\ Im B2\ Re A1\ Re B1\ Im B1]$

by  $(auto\ simp\ add:\ field-simps)$

have  $Re A2 \neq 0 \vee Re B2 \neq 0 \vee Im B2 \neq 0$

using  $\langle ?H2 \neq mat-zero \rangle **$

by  $(metis\ complex-cnj-zero\ complex-of-real-Re\ mat-zero-def\ of-real-0)$

hence  $k \neq 0$

using  $k$

by  $auto$

show  $circline-eq-cmat\ ?H1\ ?H2$

using  $**\ k\ \langle k \neq 0 \rangle$

by  $(auto\ simp\ add:\ vec-cnj-def)\ (rule-tac\ x=k\ in\ exI,\ auto\ simp\ add:\ complex.expand)$

qed

qed

qed

The only h-line that goes trough zero and a non-zero point on the x-axis is the x-axis.

**lemma**  $is-poincare-line-0-real-is-x-axis$ :

assumes  $is-poincare-line\ l\ 0_h \in\ circline-set\ l$

$x \in\ circline-set\ l \cap\ circline-set\ x-axis\ x \neq\ 0_h\ x \neq\ \infty_h$

shows  $l = x-axis$

using  $assms$

using  $is-poincare-line-trough-zero-trough-infty[OF\ assms(1-2)]$

using  $unique-circline-set[of\ x\ 0_h\ \infty_h]$

by  $auto$

The only h-line that goes trough zero and a non-zero point on the y-axis is the y-axis.

**lemma**  $is-poincare-line-0-imag-is-y-axis$ :

assumes  $is-poincare-line\ l\ 0_h \in\ circline-set\ l$

$y \in\ circline-set\ l \cap\ circline-set\ y-axis\ y \neq\ 0_h\ y \neq\ \infty_h$

shows  $l = y-axis$

using  $assms$

using  $is-poincare-line-trough-zero-trough-infty[OF\ assms(1-2)]$

using  $unique-circline-set[of\ y\ 0_h\ \infty_h]$

by  $auto$

#### 4.1.6 H-isometries preserve h-lines

$H$ -isometries are defined as homographies (actions of Möbius transformations) and antihomographies (compositions of actions of Möbius transformations with conjugation) that fix the unit disc (map it onto itself). They also map h-lines onto h-lines

We prove a bit more general lemma that states that all Möbius transformations that fix the unit circle (not necessary the unit disc) map h-lines onto h-lines

**lemma**  $unit-circle-fix-preserve-is-poincare-line$   $[simp]$ :

assumes  $unit-circle-fix\ M\ is-poincare-line\ H$

shows  $is-poincare-line\ (moebius-circline\ M\ H)$

using  $assms$

unfolding  $is-poincare-line-iff$

proof  $(safe)$

let  $?H' = moebius-ocircline\ M\ (of-circline\ H)$

```

let ?U' = moebius-ocircline M ounit-circle
assume ++: unit-circle-fix M perpendicular H unit-circle
have ounit: ounit-circle = moebius-ocircline M ounit-circle ∨
      ounit-circle = moebius-ocircline M (opposite-ocircline ounit-circle)
using ++(1) unit-circle-fix-iff[of M]
by (simp add: inj-of-ocircline moebius-circline-ocircline)

show perpendicular (moebius-circline M H) unit-circle
proof (cases pos-oriented ?H')
case True
hence *: of-circline (of-ocircline ?H') = ?H'
using of-circline-of-ocircline-pos-oriented
by blast
from ounit show ?thesis
proof
assume **: ounit-circle = moebius-ocircline M ounit-circle
show ?thesis
using ++
unfolding perpendicular-def
by (simp, subst moebius-circline-ocircline, subst *, subst **) simp
next
assume **: ounit-circle = moebius-ocircline M (opposite-ocircline ounit-circle)
show ?thesis
using ++
unfolding perpendicular-def
by (simp, subst moebius-circline-ocircline, subst *, subst **) simp
qed
next
case False
hence *: of-circline (of-ocircline ?H') = opposite-ocircline ?H'
by (metis of-circline-of-ocircline pos-oriented-of-circline)
from ounit show ?thesis
proof
assume **: ounit-circle = moebius-ocircline M ounit-circle
show ?thesis
using ++
unfolding perpendicular-def
by (simp, subst moebius-circline-ocircline, subst *, subst **) simp
next
assume **: ounit-circle = moebius-ocircline M (opposite-ocircline ounit-circle)
show ?thesis
using ++
unfolding perpendicular-def
by (simp, subst moebius-circline-ocircline, subst *, subst **) simp
qed
qed
qed simp

```

```

lemma unit-circle-fix-preserve-is-poincare-line-iff [simp]:
  assumes unit-circle-fix M
  shows is-poincare-line (moebius-circline M H)  $\longleftrightarrow$  is-poincare-line H
  using assms
  using unit-circle-fix-preserve-is-poincare-line[of M H]
  using unit-circle-fix-preserve-is-poincare-line[of moebius-inv M moebius-circline M H]
  by (auto simp del: unit-circle-fix-preserve-is-poincare-line)

```

Since h-lines are preserved by transformations that fix the unit circle, so is collinearity.

```

lemma unit-disc-fix-preserve-poincare-collinear [simp]:
  assumes unit-circle-fix M poincare-collinear A
  shows poincare-collinear (moebius-pt M ' A)
  using assms
  unfolding poincare-collinear-def
  by (auto, rule-tac x=moebius-circline M p in exI, auto)

```

```

lemma unit-disc-fix-preserve-poincare-collinear-iff [simp]:
  assumes unit-circle-fix M

```



shows *poincare-collinear* (*moebius-pt*  $M \text{ ' } A$ )  $\longleftrightarrow$  *poincare-collinear*  $A$   
 using *assms*  
 using *unit-disc-fix-preserve-poincare-collinear*[*of*  $M A$ ]  
 using *unit-disc-fix-preserve-poincare-collinear*[*of* *moebius-inv*  $M$  *moebius-pt*  $M \text{ ' } A$ ]  
 by (*auto simp del: unit-disc-fix-preserve-poincare-collinear*)

**lemma** *unit-disc-fix-preserve-poincare-collinear3* [*simp*]:  
 assumes *unit-disc-fix*  $M$   
 shows *poincare-collinear* {*moebius-pt*  $M u$ , *moebius-pt*  $M v$ , *moebius-pt*  $M w$ }  $\longleftrightarrow$   
*poincare-collinear* { $u$ ,  $v$ ,  $w$ }  
 using *assms unit-disc-fix-preserve-poincare-collinear-iff*[*of*  $M \{u, v, w\}$ ]  
 by *simp*

Conjugation is also an h-isometry and it preserves h-lines.

**lemma** *is-poincare-line-conjugate-circline* [*simp*]:  
 assumes *is-poincare-line*  $H$   
 shows *is-poincare-line* (*conjugate-circline*  $H$ )  
 using *assms*  
 by (*transfer, transfer, auto simp add: mat-cnj-def hermitean-def mat-adj-def*)

**lemma** *is-poincare-line-conjugate-circline-iff* [*simp*]:  
 shows *is-poincare-line* (*conjugate-circline*  $H$ )  $\longleftrightarrow$  *is-poincare-line*  $H$   
 using *is-poincare-line-conjugate-circline*[*of* *conjugate-circline*  $H$ ]  
 by *auto*

Since h-lines are preserved by conjugation, so is collinearity.

**lemma** *conjugate-preserve-poincare-collinear* [*simp*]:  
 assumes *poincare-collinear*  $A$   
 shows *poincare-collinear* (*conjugate*  $\text{ ' } A$ )  
 using *assms*  
 unfolding *poincare-collinear-def*  
 by *auto* (*rule-tac x=conjugate-circline p in exI, auto*)

**lemma** *conjugate-conjugate* [*simp*]: *conjugate*  $\text{ ' } \text{conjugate} \text{ ' } A = A$   
 by (*auto simp add: image-iff*)

**lemma** *conjugate-preserve-poincare-collinear-iff* [*simp*]:  
 shows *poincare-collinear* (*conjugate*  $\text{ ' } A$ )  $\longleftrightarrow$  *poincare-collinear*  $A$   
 using *conjugate-preserve-poincare-collinear*[*of*  $A$ ]  
 using *conjugate-preserve-poincare-collinear*[*of* *conjugate*  $\text{ ' } A$ ]  
 by (*auto simp del: conjugate-preserve-poincare-collinear*)

#### 4.1.7 Mapping h-lines to x-axis

Each h-line in the Poincaré model can be mapped onto the x-axis (by a unit-disc preserving Möbius transformation).

**lemma** *ex-unit-disc-fix-is-poincare-line-to-x-axis*:  
 assumes *is-poincare-line*  $l$   
 shows  $\exists M. \text{unit-disc-fix } M \wedge \text{moebius-circline } M l = \text{x-axis}$   
**proof** –  
 from *assms* obtain  $u v$  where  $u \neq v$   $u \in \text{unit-disc}$   $v \in \text{unit-disc}$  and  $\{u, v\} \subseteq \text{circline-set } l$   
 using *ex-is-poincare-line-points*  
 by *blast*  
 then obtain  $M$  where  $*$ : *unit-disc-fix*  $M$  *moebius-pt*  $M u = 0_h$  *moebius-pt*  $M v \in \text{positive-x-axis}$   
 using *ex-unit-disc-fix-to-zero-positive-x-axis*[*of*  $u v$ ]  
 by *auto*  
**moreover**  
 hence  $\{0_h, \text{moebius-pt } M v\} \subseteq \text{circline-set } \text{x-axis}$   
 unfolding *positive-x-axis-def*  
 by *auto*  
**moreover**  
 have *moebius-pt*  $M v \neq 0_h$   
 using  $\langle u \neq v \rangle *$   
 by (*metis moebius-pt-neq-I*)  
**moreover**

```

have moebius-pt  $M v \neq \infty_h$ 
  using ⟨unit-disc-fix  $M$ ⟩ ⟨ $v \in$  unit-disc⟩
  using unit-disc-fix-discI
  by fastforce
ultimately
show ?thesis
  using ⟨is-poincare-line  $l$ ⟩ ⟨ $\{u, v\} \subseteq$  circline-set  $l$ ⟩ ⟨unit-disc-fix  $M$ ⟩
  using is-poincare-line-0-real-is-x-axis[of moebius-circline  $M l$  moebius-pt  $M v$ ]
  by (rule-tac  $x=M$  in  $exI$ , force)
qed

```

When proving facts about h-lines, without loss of generality it can be assumed that h-line is the x-axis (if the property being proved is invariant under Möbius transformations that fix the unit disc).

```

lemma wlog-line-x-axis:
  assumes is-line: is-poincare-line  $H$ 
  assumes x-axis:  $P$  x-axis
  assumes preserves:  $\bigwedge M. \llbracket$ unit-disc-fix  $M; P$  (moebius-circline  $M H$ ) $\rrbracket \implies P H$ 
  shows  $P H$ 
  using assms
  using ex-unit-disc-fix-is-poincare-line-to-x-axis[of  $H$ ]
  by auto

```

## 4.2 Construction of the h-line between the two given points

Next we show how to construct the (unique) h-line between the two given points in the Poincaré model

Geometrically, h-line can be constructed by finding the inverse point of one of the two points and by constructing the circle (or line) through it and the two given points.

Algebraically, for two given points  $u$  and  $v$  in  $\mathbb{C}$ , the h-line matrix coefficients can be  $A = i \cdot (u\bar{v} - v\bar{u})$  and  $B = i \cdot (v(|u|^2 + 1) - u(|v|^2 + 1))$ .

We need to extend this to homogenous coordinates. There are several degenerate cases.

- If  $\{z, w\} = \{0_h, \infty_h\}$  then there is no unique h-line (any line through zero is an h-line).
- If  $z$  and  $w$  are mutually inverse, then the construction fails (both geometric and algebraic).
- If  $z$  and  $w$  are different points on the unit circle, then the standard construction fails (only geometric).
- None of this problematic cases occur when  $z$  and  $w$  are inside the unit disc.

We express the construction algebraically, and construct the Hermitean circline matrix for the two points given in homogenous coordinates. It works correctly in all cases except when the two points are the same or are mutually inverse.

```

definition mk-poincare-line-cmat :: real  $\Rightarrow$  complex  $\Rightarrow$  complex-mat where
  [simp]: mk-poincare-line-cmat  $A B =$  (cor  $A, B, \text{cnj } B, \text{cor } A$ )

```

```

lemma mk-poincare-line-cmat-zero-iff:
  mk-poincare-line-cmat  $A B =$  mat-zero  $\longleftrightarrow A = 0 \wedge B = 0$ 
  by auto

```

```

lemma mk-poincare-line-cmat-hermitean
  [simp]: hermitean (mk-poincare-line-cmat  $A B$ )
  by simp

```

```

lemma mk-poincare-line-cmat-scale:
  cor  $k *_{sm}$  mk-poincare-line-cmat  $A B =$  mk-poincare-line-cmat ( $k * A$ ) ( $k * B$ )
  by simp

```

```

definition poincare-line-cvec-cmat :: complex-vec  $\Rightarrow$  complex-vec  $\Rightarrow$  complex-mat where
  [simp]: poincare-line-cvec-cmat  $z w =$ 
    (let ( $z1, z2$ ) =  $z$ ;
        ( $w1, w2$ ) =  $w$ ;
        nom =  $w1 * \text{cnj } w2 * (z1 * \text{cnj } z1 + z2 * \text{cnj } z2) - z1 * \text{cnj } z2 * (w1 * \text{cnj } w1 + w2 * \text{cnj } w2)$ ;
        den =  $z1 * \text{cnj } z2 * \text{cnj } w1 * w2 - w1 * \text{cnj } w2 * \text{cnj } z1 * z2$ ;
        in if den  $\neq 0$  then
          mk-poincare-line-cmat (Re( $i * \text{den}$ )) ( $i * \text{nom}$ )
        else if  $z1 * \text{cnj } z2 \neq 0$  then
          mk-poincare-line-cmat 0 ( $i * z1 * \text{cnj } z2$ )

```

```

else if w1*cnj w2 ≠ 0 then
  mk-poincare-line-cmat 0 (i*w1*cnj w2)
else
  mk-poincare-line-cmat 0 i)

```

**lemma** *poincare-line-cvec-cmat-AeqD*:  
**assumes** *poincare-line-cvec-cmat* z w = (A, B, C, D)  
**shows** A = D  
**using** *assms*  
**by** (*cases* z, *cases* w) (*auto split: if-split-asm*)

**lemma** *poincare-line-cvec-cmat-hermitean* [*simp*]:  
**shows** *hermitean* (*poincare-line-cvec-cmat* z w)  
**by** (*cases* z, *cases* w) (*auto split: if-split-asm simp del: mk-poincare-line-cmat-def*)

**lemma** *poincare-line-cvec-cmat-nonzero* [*simp*]:  
**assumes** z ≠ *vec-zero* w ≠ *vec-zero*  
**shows** *poincare-line-cvec-cmat* z w ≠ *mat-zero*  
**proof** –

```

obtain z1 z2 w1 w2 where *: z = (z1, z2) w = (w1, w2)
by (cases z, cases w, auto)

```

```

let ?den = z1*cnj z2*cnj w1*w2 - w1*cnj w2*cnj z1*z2
show ?thesis

```

```

proof (cases ?den ≠ 0)

```

```

  case True

```

```

    have is-real (i * ?den)

```

```

      using eq-cnj-iff-real[of i * ?den]

```

```

      by (simp add: field-simps)

```

```

    hence Re (i * ?den) ≠ 0

```

```

      using ⟨?den ≠ 0⟩

```

```

      by (metis complex-i-not-zero complex-surj mult-eq-0-iff zero-complex.code)

```

```

    thus ?thesis

```

```

      using * ⟨?den ≠ 0⟩

```

```

      by (simp del: mk-poincare-line-cmat-def mat-zero-def add: mk-poincare-line-cmat-zero-iff)

```

```

next

```

```

  case False

```

```

    thus ?thesis

```

```

      using *

```

```

      by (simp del: mk-poincare-line-cmat-def mat-zero-def add: mk-poincare-line-cmat-zero-iff)

```

```

qed

```

```

qed

```

**lift-definition** *poincare-line-hcoords-clmat* :: *complex-homo-coords* ⇒ *complex-homo-coords* ⇒ *circline-mat* **is** *poincare-line-cvec-cmat*  
**using** *poincare-line-cvec-cmat-hermitean* *poincare-line-cvec-cmat-nonzero*  
**by** *simp*

**lift-definition** *poincare-line* :: *complex-homo* ⇒ *complex-homo* ⇒ *circline* **is** *poincare-line-hcoords-clmat*  
**proof** *transfer*

```

fix za zb wa wb

```

```

assume za ≠ vec-zero zb ≠ vec-zero wa ≠ vec-zero wb ≠ vec-zero

```

```

assume za ≈v zb wa ≈v wb

```

```

obtain za1 za2 zb1 zb2 wa1 wa2 wb1 wb2 where

```

```

  *: (za1, za2) = za (zb1, zb2) = zb

```

```

    (wa1, wa2) = wa (wb1, wb2) = wb

```

```

by (cases za, cases zb, cases wa, cases wb, auto)

```

```

obtain kz kw where

```

```

  **: kz ≠ 0 kw ≠ 0 zb1 = kz * za1 zb2 = kz * za2 wb1 = kw * wa1 wb2 = kw * wa2

```

```

using ⟨za ≈v zb⟩ ⟨wa ≈v wb⟩ *[symmetric]

```

```

by auto

```

```

let ?nom = λ z1 z2 w1 w2. w1*cnj w2*(z1*cnj z1 + z2*cnj z2) - z1*cnj z2*(w1*cnj w1 + w2*cnj w2)

```

```

let ?den = λ z1 z2 w1 w2. z1*cnj z2*cnj w1*w2 - w1*cnj w2*cnj z1*z2

```

```

show circline-eq-cmat (poincare-line-cvec-cmat za wa)

```

```

      (poincare-line-cvec-cmat zb wb)
proof-
  have  $\exists k. k \neq 0 \wedge$ 
    poincare-line-cvec-cmat (zb1, zb2) (wb1, wb2) = cor k *sm poincare-line-cvec-cmat (za1, za2) (wa1, wa2)
proof (cases ?den za1 za2 wa1 wa2  $\neq 0$ )
  case True
  hence ?den zb1 zb2 wb1 wb2  $\neq 0$ 
    using **
    by (simp add: field-simps)

  let ?k = kz * cnj kz * kw * cnj kw

  have ?k  $\neq 0$ 
    using **
    by simp

  have is-real ?k
    using eq-cnj-iff-real[of ?k]
    by auto

  have cor (Re ?k) = ?k
    using <is-real ?k>
    using complex-of-real-Re
    by blast

  have Re ?k  $\neq 0$ 
    using <?k  $\neq 0$ > <cor (Re ?k) = ?k>
    by (metis of-real-0)

  have arg1: Re (i * ?den zb1 zb2 wb1 wb2) = Re ?k * Re (i * ?den za1 za2 wa1 wa2)
    apply (subst **)+
    apply (subst Re-mult-real[symmetric, OF <is-real ?k>])
    apply (rule arg-cong[where f=Re])
    apply (simp add: field-simps)
    done

  have arg2: i * ?nom zb1 zb2 wb1 wb2 = ?k * i * ?nom za1 za2 wa1 wa2
    using **
    by (simp add: field-simps)

  have mk-poincare-line-cmat (Re (i*?den zb1 zb2 wb1 wb2)) (i*?nom zb1 zb2 wb1 wb2) =
    cor (Re ?k) *sm mk-poincare-line-cmat (Re (i*?den za1 za2 wa1 wa2)) (i*?nom za1 za2 wa1 wa2)
    using <cor (Re ?k) = ?k> <is-real ?k>
    apply (subst mk-poincare-line-cmat-scale)
    apply (subst arg1, subst arg2)
    apply (subst <cor (Re ?k) = ?k>)+
    apply simp
    done

  thus ?thesis
    using <?den za1 za2 wa1 wa2  $\neq 0$ > <?den zb1 zb2 wb1 wb2  $\neq 0$ >
    using <Re ?k  $\neq 0$ > <cor (Re ?k) = ?k>
    by (rule-tac x=Re ?k in exI, simp)
next
  case False
  hence ?den zb1 zb2 wb1 wb2 = 0
    using **
    by (simp add: field-simps)
  show ?thesis
proof (cases za1*cnj za2  $\neq 0$ )
  case True
  hence zb1*cnj zb2  $\neq 0$ 
    using **
    by (simp add: field-simps)

  let ?k = kz * cnj kz

  have ?k  $\neq 0$  is-real ?k
    using **

```

```

    using eq-cnj-iff-real[of ?k]
  by auto
thus ?thesis
  using ⟨za1 * cnj za2 ≠ 0⟩ ⟨zb1 * cnj zb2 ≠ 0⟩
  using ⟨¬ (?den za1 za2 wa1 wa2 ≠ 0)⟩ ⟨?den zb1 zb2 wb1 wb2 = 0⟩ **
  by (rule-tac x=Re (kz * cnj kz) in exI, auto simp add: complex.expand)
next
case False
hence zb1 * cnj zb2 = 0
  using **
  by (simp add: field-simps)
show ?thesis
proof (cases wa1 * cnj wa2 ≠ 0)
case True
hence wb1 * cnj wb2 ≠ 0
  using **
  by (simp add: field-simps)

let ?k = kw * cnj kw

have ?k ≠ 0 is-real ?k
  using **
  using eq-cnj-iff-real[of ?k]
  by auto

thus ?thesis
  using ⟨¬ (za1 * cnj za2 ≠ 0)⟩
  using ⟨wa1 * cnj wa2 ≠ 0⟩ ⟨wb1 * cnj wb2 ≠ 0⟩
  using ⟨¬ (?den za1 za2 wa1 wa2 ≠ 0)⟩ ⟨?den zb1 zb2 wb1 wb2 = 0⟩ **
  by (rule-tac x=Re (kw * cnj kw) in exI)
  (auto simp add: complex.expand)
next
case False
hence wb1 * cnj wb2 = 0
  using **
  by (simp add: field-simps)
thus ?thesis
  using ⟨¬ (za1 * cnj za2 ≠ 0)⟩ ⟨zb1 * cnj zb2 = 0⟩
  using ⟨¬ (wa1 * cnj wa2 ≠ 0)⟩ ⟨wb1 * cnj wb2 = 0⟩
  using ⟨¬ (?den za1 za2 wa1 wa2 ≠ 0)⟩ ⟨?den zb1 zb2 wb1 wb2 = 0⟩ **
  by simp
qed
qed
qed
thus ?thesis
  using *[symmetric]
  by simp
qed
qed

```

#### 4.2.1 Correctness of the construction

For finite points, our definition matches the classic algebraic definition for points in  $\mathbb{C}$  (given in ordinary, not homogenous coordinates).

**lemma** *poincare-line-non-homogenous:*

**assumes**  $u \neq \infty_h$   $v \neq \infty_h$   $u \neq v$   $u \neq \text{inversion } v$

**shows** let  $u' = \text{to-complex } u$ ;  $v' = \text{to-complex } v$ ;

$$A = i * (u' * cnj v' - v' * cnj u');$$

$$B = i * (v' * ((cmod u')^2 + 1) - u' * ((cmod v')^2 + 1))$$

in *poincare-line*  $u v = \text{mk-circline } A B (\text{cnj } B) A$

**using** *assms*

**unfolding** *unit-disc-def disc-def inversion-def*

**apply** (*simp add: Let-def*)

**proof** (*transfer, transfer, safe*)

**fix**  $u1 u2 v1 v2$

**assume**  $uv: (u1, u2) \neq \text{vec-zero}$   $(v1, v2) \neq \text{vec-zero}$

```

       $\neg (u1, u2) \approx_v \infty_v \neg (v1, v2) \approx_v \infty_v$ 
       $\neg (u1, u2) \approx_v (v1, v2) \neg (u1, u2) \approx_v \text{conjugate-cvec (reciprocal-cvec (v1, v2))}$ 
let ?u = to-complex-cvec (u1, u2) and ?v = to-complex-cvec (v1, v2)
let ?A = i * (?u * cnj ?v - ?v * cnj ?u)
let ?B = i * (?v * ((cor (cmod ?u))2 + 1) - ?u * ((cor (cmod ?v))2 + 1))
let ?C = - (i * (cnj ?v * ((cor (cmod ?u))2 + 1) - cnj ?u * ((cor (cmod ?v))2 + 1)))
let ?D = ?A
let ?H = (?A, ?B, ?C, ?D)

let ?den = u1 * cnj u2 * cnj v1 * v2 - v1 * cnj v2 * cnj u1 * u2

have u2 ≠ 0 v2 ≠ 0
  using wv
  using inf-cvec-z2-zero-iff
  by blast+

have  $\neg (u1, u2) \approx_v (cnj v2, cnj v1)$ 
  using wv(6)
  by (simp add: vec-cnj-def)
moreover
have (cnj v2, cnj v1) ≠ vec-zero
  using wv(2)
  by auto
ultimately
have *: u1 * cnj v1 ≠ u2 * cnj v2 u1 * v2 ≠ u2 * v1
  using wv(5) wv(1) wv(2) ⟨u2 ≠ 0⟩ ⟨v2 ≠ 0⟩
  using complex-cvec-eq-mix
  by blast+

show circline-eq-cmat (poincare-line-cvec-cmat (u1, u2) (v1, v2))
  (mk-circline-cmat ?A ?B ?C ?D)
proof (cases ?den ≠ 0)
  case True

  let ?nom = v1 * cnj v2 * (u1 * cnj u1 + u2 * cnj u2) - u1 * cnj u2 * (v1 * cnj v1 + v2 * cnj v2)
  let ?H' = mk-poincare-line-cmat (Re (i * ?den)) (i * ?nom)

  have circline-eq-cmat ?H ?H'
  proof–
    let ?k = (u2 * cnj v2) * (v2 * cnj u2)
    have is-real ?k
      using eq-cnj-iff-real
      by fastforce
    hence cor (Re ?k) = ?k
      using complex-of-real-Re
      by blast

  have Re (i * ?den) = Re ?k * ?A
  proof–
    have ?A = cnj ?A
      by (simp add: field-simps)
    hence is-real ?A
      using eq-cnj-iff-real
      by fastforce
    moreover
    have i * ?den = cnj (i * ?den)
      by (simp add: field-simps)
    hence is-real (i * ?den)
      using eq-cnj-iff-real
      by fastforce
    hence cor (Re (i * ?den)) = i * ?den
      using complex-of-real-Re
      by blast
    ultimately
    show ?thesis

```

using  $\langle \text{cor } (Re \ ?k) = ?k \rangle$   
 by (simp add: field-simps)  
 qed

moreover

have  $i * ?nom = Re \ ?k * ?B$   
 using  $\langle \text{cor } (Re \ ?k) = ?k \rangle \langle u2 \neq 0 \rangle \langle v2 \neq 0 \rangle \text{complex-mult-cnj-cmod[symmetric]}$   
 by (auto simp add: field-simps)

moreover

have  $?k \neq 0$   
 using  $\langle u2 \neq 0 \rangle \langle v2 \neq 0 \rangle$   
 by simp  
 hence  $Re \ ?k \neq 0$   
 using  $\langle \text{is-real } ?k \rangle$   
 by (metis  $\langle \text{cor } (Re \ ?k) = ?k \rangle$  of-real-0)

ultimately

show  $?thesis$   
 by simp (rule-tac  $x=Re \ ?k$  in  $exI$ , simp add: mult commute)  
 qed

moreover

have  $\text{poincare-line-cvec-cmat } (u1, u2) (v1, v2) = ?H'$   
 using  $\langle ?den \neq 0 \rangle$   
 unfolding  $\text{poincare-line-cvec-cmat-def}$   
 by (simp add: Let-def)

moreover

hence  $\text{hermitean } ?H' \wedge ?H' \neq \text{mat-zero}$   
 by (metis  $\text{mk-poincare-line-cmat-hermitean } \text{poincare-line-cvec-cmat-nonzero } uw(1) uw(2)$ )

hence  $\text{hermitean } ?H \wedge ?H \neq \text{mat-zero}$   
 using  $\langle \text{circline-eq-cmat } ?H \ ?H' \rangle$   
 using  $\text{circline-eq-cmat-hermitean-nonzero[of } ?H' \ ?H] \text{symp-circline-eq-cmat}$   
 unfolding  $\text{symp-def}$   
 by metis

hence  $\text{mk-circline-cmat } ?A \ ?B \ ?C \ ?D = ?H$   
 by simp

ultimately

have  $\text{circline-eq-cmat } (\text{mk-circline-cmat } ?A \ ?B \ ?C \ ?D)$   
 $(\text{poincare-line-cvec-cmat } (u1, u2) (v1, v2))$   
 by simp  
 thus  $?thesis$   
 using  $\text{symp-circline-eq-cmat}$   
 unfolding  $\text{symp-def}$   
 by blast

next

case  $False$

let  $?d = v1 * (u1 * \text{cnj } u1 / (u2 * \text{cnj } u2) + 1) / v2 - u1 * (v1 * \text{cnj } v1 / (v2 * \text{cnj } v2) + 1) / u2$   
 let  $?cd = \text{cnj } v1 * (u1 * \text{cnj } u1 / (u2 * \text{cnj } u2) + 1) / \text{cnj } v2 - \text{cnj } u1 * (v1 * \text{cnj } v1 / (v2 * \text{cnj } v2) + 1) / \text{cnj}$

$u2$

have  $\text{cnj } ?d = ?cd$   
 by (simp add: mult commute)

let  $?d1 = (v1 / v2) * (\text{cnj } u1 / \text{cnj } u2) - 1$   
 let  $?d2 = u1 / u2 - v1 / v2$

have \*\*:  $?d = ?d1 * ?d2$

```

using ⟨¬ ?den ≠ 0⟩ ⟨u2 ≠ 0⟩ ⟨v2 ≠ 0⟩
by (simp add: field-simps)

hence ?d ≠ 0
using ⟨¬ ?den ≠ 0⟩ ⟨u2 ≠ 0⟩ ⟨v2 ≠ 0⟩ *
by auto (simp add: field-simps)+

have is-real ?d1
proof-
  have conj ?d1 = ?d1
  using ⟨¬ ?den ≠ 0⟩ ⟨u2 ≠ 0⟩ ⟨v2 ≠ 0⟩ *
  by (simp add: field-simps)
  thus ?thesis
  using eq-conj-iff-real
  by blast
qed

show ?thesis
proof (cases u1 * conj u2 ≠ 0)
  case True
  let ?nom = u1 * conj u2
  let ?H' = mk-poincare-line-cmat 0 (i * ?nom)

  have circline-eq-cmat ?H ?H'
  proof-

    let ?k = (u1 * conj u2) / ?d

    have is-real ?k
    proof-
      have is-real ((u1 * conj u2) / ?d2)
      proof-
        let ?rhs = (u2 * conj u2) / (1 - (v1*u2)/(u1*v2))

        have 1: (u1 * conj u2) / ?d2 = ?rhs
        using ⟨¬ ?den ≠ 0⟩ ⟨u2 ≠ 0⟩ ⟨v2 ≠ 0⟩ * ⟨u1 * conj u2 ≠ 0⟩
        by (simp add: field-simps)
        moreover
        have conj ?rhs = ?rhs
        proof-
          have conj (1 - v1 * u2 / (u1 * v2)) = 1 - v1 * u2 / (u1 * v2)
          using ⟨¬ ?den ≠ 0⟩ ⟨u2 ≠ 0⟩ ⟨v2 ≠ 0⟩ * ⟨u1 * conj u2 ≠ 0⟩
          by (simp add: field-simps)
          moreover
          have conj (u2 * conj u2) = u2 * conj u2
          by simp
          ultimately
          show ?thesis
          by simp
        qed
      qed

      ultimately

      show ?thesis
      using eq-conj-iff-real
      by fastforce
    qed

    thus ?thesis
    using ** ⟨is-real ?d1⟩
    by (metis complex-conj-divide divide-divide-eq-left' eq-conj-iff-real)
  qed

  have ?k ≠ 0
  using ⟨?d ≠ 0⟩ ⟨u1 * conj u2 ≠ 0⟩
  by simp

```



```

have cnj ?k = ?k
  using ⟨is-real ?k⟩
  using eq-cnj-iff-real by blast

have Re ?k ≠ 0
  using ⟨?k ≠ 0⟩ ⟨is-real ?k⟩
  by (metis complex.expand zero-complex.simps(1) zero-complex.simps(2))

have u1 * cnj u2 = ?k * ?d
  using ⟨?d ≠ 0⟩
  by simp

moreover

hence cnj u1 * u2 = cnj ?k * cnj ?d
  by (metis complex-cnj-cnj complex-cnj-mult)
hence cnj u1 * u2 = ?k * ?cd
  using ⟨cnj ?k = ?k⟩ ⟨cnj ?d = ?cd⟩
  by metis

ultimately

show ?thesis
  using <~ ?den ≠ 0⟩ ⟨u1 * cnj u2 ≠ 0⟩ ⟨u2 ≠ 0⟩ ⟨v2 ≠ 0⟩ ⟨Re ?k ≠ 0⟩ ⟨is-real ?k⟩ ⟨?d ≠ 0⟩
  using complex-mult-cnj-cmod[symmetric, of u1]
  using complex-mult-cnj-cmod[symmetric, of v1]
  using complex-mult-cnj-cmod[symmetric, of u2]
  using complex-mult-cnj-cmod[symmetric, of v2]
  apply (simp add: power-divide norm-mult norm-divide)
  apply (rule-tac x=Re ?k in exI)
  apply simp
  apply (simp add: field-simps)
  done
qed

moreover

have poincare-line-cvec-cmat (u1, u2) (v1, v2) = ?H'
  using ⟨¬ ?den ≠ 0⟩ ⟨u1 * cnj u2 ≠ 0⟩
  unfolding poincare-line-cvec-cmat-def
  by (simp add: Let-def)

moreover

hence hermitean ?H' ∧ ?H' ≠ mat-zero
  by (metis mk-poincare-line-cmat-hermitean poincare-line-cvec-cmat-nonzero uw(1) uw(2))

hence hermitean ?H ∧ ?H ≠ mat-zero
  using ⟨circline-eq-cmat ?H ?H'⟩
  using circline-eq-cmat-hermitean-nonzero[of ?H' ?H] symp-circline-eq-cmat
  unfolding symp-def
  by metis

hence mk-circline-cmat ?A ?B ?C ?D = ?H
  by simp

ultimately

have circline-eq-cmat (mk-circline-cmat ?A ?B ?C ?D)
  (poincare-line-cvec-cmat (u1, u2) (v1, v2))
  by simp
thus ?thesis
  using symp-circline-eq-cmat
  unfolding symp-def
  by blast

```

```

next
case False
show ?thesis
proof (cases v1 * cnj v2 ≠ 0)
case True
let ?nom = v1 * cnj v2
let ?H' = mk-poincare-line-cmat 0 (i * ?nom)

have circline-eq-cmat ?H ?H'
proof-
let ?k = (v1 * cnj v2) / ?d

have is-real ?k
proof-
have is-real ((v1 * cnj v2) / ?d2)
proof-
let ?rhs = (v2 * cnj v2) / ((u1*v2)/(u2*v1) - 1)

have 1: (v1 * cnj v2) / ?d2 = ?rhs
using ⟨¬ ?den ≠ 0⟩ ⟨u2 ≠ 0⟩ ⟨v2 ≠ 0⟩ * ⟨v1 * cnj v2 ≠ 0⟩
by (simp add: field-simps)
moreover
have cnj ?rhs = ?rhs
proof-
have cnj (u1 * v2 / (u2 * v1) - 1) = u1 * v2 / (u2 * v1) - 1
using ⟨¬ ?den ≠ 0⟩ ⟨u2 ≠ 0⟩ ⟨v2 ≠ 0⟩ * ⟨v1 * cnj v2 ≠ 0⟩
by (simp add: field-simps)
moreover
have cnj (v2 * cnj v2) = v2 * cnj v2
by simp
ultimately
show ?thesis
by simp
qed

ultimately

show ?thesis
using eq-cnj-iff-real
by fastforce
qed

thus ?thesis
using ** ⟨is-real ?d1⟩
by (metis complex-cnj-divide divide-divide-eq-left' eq-cnj-iff-real)
qed

have ?k ≠ 0
using ⟨?d ≠ 0⟩ ⟨v1 * cnj v2 ≠ 0⟩
by simp

have cnj ?k = ?k
using ⟨is-real ?k⟩
using eq-cnj-iff-real by blast

have Re ?k ≠ 0
using ⟨?k ≠ 0⟩ ⟨is-real ?k⟩
by (metis complex.expand zero-complex.simps(1) zero-complex.simps(2))

have v1 * cnj v2 = ?k * ?d
using ⟨?d ≠ 0⟩
by simp

moreover

hence cnj v1 * v2 = cnj ?k * cnj ?d

```

by (*metis complex-cnj-cnj complex-cnj-mult*)  
**hence**  $\text{cnj } v1 * v2 = ?k * ?cd$   
**using**  $\langle \text{cnj } ?k = ?k \rangle \langle \text{cnj } ?d = ?cd \rangle$   
**by** *metis*

**ultimately**

**show** *?thesis*  
**using**  $\langle \sim ?den \neq 0 \rangle \langle v1 * \text{cnj } v2 \neq 0 \rangle \langle u2 \neq 0 \rangle \langle v2 \neq 0 \rangle \langle \text{Re } ?k \neq 0 \rangle \langle \text{is-real } ?k \rangle \langle ?d \neq 0 \rangle$   
**using** *complex-mult-cnj-cmod[symmetric, of u1]*  
**using** *complex-mult-cnj-cmod[symmetric, of v1]*  
**using** *complex-mult-cnj-cmod[symmetric, of u2]*  
**using** *complex-mult-cnj-cmod[symmetric, of v2]*  
**apply** (*simp add: power-divide norm-mult norm-divide*)  
**apply** (*rule-tac x=Re ?k in exI*)  
**apply** *simp*  
**apply** (*simp add: field-simps*)  
**done**  
**qed**

**moreover**

**have** *poincare-line-cvec-cmat (u1, u2) (v1, v2) = ?H'*  
**using**  $\langle \neg ?den \neq 0 \rangle \langle \neg u1 * \text{cnj } u2 \neq 0 \rangle \langle v1 * \text{cnj } v2 \neq 0 \rangle$   
**unfolding** *poincare-line-cvec-cmat-def*  
**by** (*simp add: Let-def*)

**moreover**

**hence** *hermitean ?H'  $\wedge$  ?H'  $\neq$  mat-zero*  
**by** (*metis mk-poincare-line-cmat-hermitean poincare-line-cvec-cmat-nonzero w(1) w(2)*)

**hence** *hermitean ?H  $\wedge$  ?H  $\neq$  mat-zero*  
**using**  $\langle \text{circline-eq-cmat } ?H ?H' \rangle$   
**using** *circline-eq-cmat-hermitean-nonzero[of ?H' ?H] symp-circline-eq-cmat*  
**unfolding** *symp-def*  
**by** *metis*

**hence** *mk-circline-cmat ?A ?B ?C ?D = ?H*  
**by** *simp*

**ultimately**

**have** *circline-eq-cmat (mk-circline-cmat ?A ?B ?C ?D)*  
*(poincare-line-cvec-cmat (u1, u2) (v1, v2))*

**by** *simp*

**thus** *?thesis*

**using** *symp-circline-eq-cmat*

**unfolding** *symp-def*

**by** *blast*

**next**

**case** *False*

**hence** *False*

**using**  $\langle \neg ?den \neq 0 \rangle \langle \neg u1 * \text{cnj } u2 \neq 0 \rangle w$

**by** (*simp add:  $\langle u2 \neq 0 \rangle \langle v2 \neq 0 \rangle$* )

**thus** *?thesis*

**by** *simp*

**qed**

**qed**

**qed**

**qed**

Our construction (in homogenous coordinates) always yields an h-line that contain two starting points (this also holds for all degenerate cases except when points are the same).

**lemma** *poincare-line [simp]:*

**assumes**  $z \neq w$

```

shows on-circline (poincare-line z w) z
      on-circline (poincare-line z w) w
proof–
have on-circline (poincare-line z w) z  $\wedge$  on-circline (poincare-line z w) w
  using assms
proof (transfer, transfer)
  fix z w
  assume vz: z  $\neq$  vec-zero w  $\neq$  vec-zero
  obtain z1 z2 w1 w2 where
    zw: (z1, z2) = z (w1, w2) = w
    by (cases z, cases w, auto)

let ?den = z1*cnj z2*cnj w1*w2 - w1*cnj w2*cnj z1*z2
have *: cor (Re (i * ?den)) = i * ?den
proof–
  have cnj ?den = -?den
  by auto
  hence is-imag ?den
  using eq-minus-cnj-iff-imag[of ?den]
  by simp
  thus ?thesis
  using complex-of-real-Re[of i * ?den]
  by simp
qed
show on-circline-cmat-cvec (poincare-line-cvec-cmat z w) z  $\wedge$ 
      on-circline-cmat-cvec (poincare-line-cvec-cmat z w) w
  unfolding poincare-line-cvec-cmat-def mk-poincare-line-cmat-def
  apply (subst zw[symmetric])+
  unfolding Let-def prod.case
  apply (subst *)+
  by (auto simp add: vec-cnj-def field-simps)
qed
thus on-circline (poincare-line z w) z on-circline (poincare-line z w) w
  by auto
qed

```

```

lemma poincare-line-circline-set [simp]:
  assumes z  $\neq$  w
  shows z  $\in$  circline-set (poincare-line z w)
        w  $\in$  circline-set (poincare-line z w)
  using assms
  by (auto simp add: circline-set-def)

```

When the points are different, the constructed line matrix always has a negative determinant

```

lemma poincare-line-type:
  assumes z  $\neq$  w
  shows circline-type (poincare-line z w) = -1
proof–
have  $\exists$  a b. a  $\neq$  b  $\wedge$  {a, b}  $\subseteq$  circline-set (poincare-line z w)
  using poincare-line[of z w] assms
  unfolding circline-set-def
  by (rule-tac x=z in exI, rule-tac x=w in exI, simp)
thus ?thesis
  using circline-type[of poincare-line z w]
  using circline-type-pos-card-eq0[of poincare-line z w]
  using circline-type-zero-card-eq1[of poincare-line z w]
  by auto
qed

```

The constructed line is an h-line in the Poincaré model (in all cases when the two points are different)

```

lemma is-poincare-line-poincare-line [simp]:
  assumes z  $\neq$  w
  shows is-poincare-line (poincare-line z w)
  using poincare-line-type[of z w, OF assms]
proof (transfer, transfer)
  fix z w

```

```

assume vz:  $z \neq \text{vec-zero } w \neq \text{vec-zero}$ 
obtain  $A B C D$  where  $*$ : poincare-line-cvec-cmat  $z w = (A, B, C, D)$ 
  by (cases poincare-line-cvec-cmat  $z w$ ) auto
assume circline-type-cmat (poincare-line-cvec-cmat  $z w$ ) = - 1
thus is-poincare-line-cmat (poincare-line-cvec-cmat  $z w$ )
  using vz *
  using poincare-line-cvec-cmat-hermitean[of z w]
  using poincare-line-cvec-cmat-nonzero[of z w]
  using poincare-line-cvec-cmat-AeqD[of z w A B C D]
  using hermitean-elems[of A B C D]
  using cmod-power2[of D] cmod-power2[of C]
  unfolding is-poincare-line-cmat-def
  by (simp del: poincare-line-cvec-cmat-def add: sgn-1-neg power2-eq-square)
qed

```

When the points are different, the constructed h-line between two points also contains their inverses

```

lemma poincare-line-inversion:
  assumes  $z \neq w$ 
  shows on-circline (poincare-line  $z w$ ) (inversion  $z$ )
    on-circline (poincare-line  $z w$ ) (inversion  $w$ )
  using assms
  using is-poincare-line-poincare-line[OF  $\langle z \neq w \rangle$ ]
  using is-poincare-line-inverse-point
  unfolding circline-set-def
  by auto

```

When the points are different, the onstructed h-line between two points contains the inverse of its every point

```

lemma poincare-line-inversion-full:
  assumes  $u \neq v$ 
  assumes on-circline (poincare-line  $u v$ )  $x$ 
  shows on-circline (poincare-line  $u v$ ) (inversion  $x$ )
  using is-poincare-line-inverse-point[of poincare-line u v x]
  using is-poincare-line-poincare-line[OF  $\langle u \neq v \rangle$ ] assms
  unfolding circline-set-def
  by simp

```

#### 4.2.2 Existence of h-lines

There is an h-line trough every point in the Poincaré model

```

lemma ex-poincare-line-one-point:
  shows  $\exists l. \text{is-poincare-line } l \wedge z \in \text{circline-set } l$ 
proof (cases  $z = 0_h$ )
  case True
  thus ?thesis
    by (rule-tac  $x=x\text{-axis}$  in exI) simp
next
  case False
  thus ?thesis
    by (rule-tac  $x=\text{poincare-line } 0_h$   $z$  in exI) auto
qed

```

```

lemma poincare-collinear-singleton [simp]:
  assumes  $u \in \text{unit-disc}$ 
  shows poincare-collinear  $\{u\}$ 
  using assms
  using ex-poincare-line-one-point[of u]
  by (auto simp add: poincare-collinear-def)

```

There is an h-line trough every two points in the Poincaré model

```

lemma ex-poincare-line-two-points:
  assumes  $z \neq w$ 
  shows  $\exists l. \text{is-poincare-line } l \wedge z \in \text{circline-set } l \wedge w \in \text{circline-set } l$ 
  using assms
  by (rule-tac  $x=\text{poincare-line } z$   $w$  in exI, simp)

```

```

lemma poincare-collinear-doubleton [simp]:
  assumes  $u \in \text{unit-disc } v \in \text{unit-disc}$ 
  shows poincare-collinear  $\{u, v\}$ 
  using assms
  using ex-poincare-line-one-point[of u]
  using ex-poincare-line-two-points[of u v]
  by (cases u = v) (simp-all add: poincare-collinear-def)

```

### 4.2.3 Uniqueness of h-lines

The only h-line between two points is the one obtained by the line-construction.

First we show this only for two different points inside the disc.

```

lemma unique-poincare-line:
  assumes in-disc:  $u \neq v \ u \in \text{unit-disc } v \in \text{unit-disc}$ 
  assumes on-l:  $u \in \text{circline-set } l \ v \in \text{circline-set } l \ \text{is-poincare-line } l$ 
  shows  $l = \text{poincare-line } u \ v$ 
  using assms
  using unique-is-poincare-line[of u v l poincare-line u v]
  unfolding circline-set-def
  by auto

```

The assumption that the points are inside the disc can be relaxed.

```

lemma unique-poincare-line-general:
  assumes in-disc:  $u \neq v \ u \neq \text{inversion } v$ 
  assumes on-l:  $u \in \text{circline-set } l \ v \in \text{circline-set } l \ \text{is-poincare-line } l$ 
  shows  $l = \text{poincare-line } u \ v$ 
  using assms
  using unique-is-poincare-line-general[of u v l poincare-line u v]
  unfolding circline-set-def
  by auto

```

The explicit line construction enables us to prove that there exists a unique h-line through any given two h-points (uniqueness part was already shown earlier).

First we show this only for two different points inside the disc.

```

lemma ex1-poincare-line:
  assumes  $u \neq v \ u \in \text{unit-disc } v \in \text{unit-disc}$ 
  shows  $\exists! l. \text{is-poincare-line } l \wedge u \in \text{circline-set } l \wedge v \in \text{circline-set } l$ 
proof (rule ex1I)
  let  $?l = \text{poincare-line } u \ v$ 
  show is-poincare-line  $?l \wedge u \in \text{circline-set } ?l \wedge v \in \text{circline-set } ?l$ 
    using assms
    unfolding circline-set-def
    by auto
next
  fix  $l$ 
  assume is-poincare-line  $l \wedge u \in \text{circline-set } l \wedge v \in \text{circline-set } l$ 
  thus  $l = \text{poincare-line } u \ v$ 
    using unique-poincare-line assms
    by auto
qed

```

The assumption that the points are in the disc can be relaxed.

```

lemma ex1-poincare-line-general:
  assumes  $u \neq v \ u \neq \text{inversion } v$ 
  shows  $\exists! l. \text{is-poincare-line } l \wedge u \in \text{circline-set } l \wedge v \in \text{circline-set } l$ 
proof (rule ex1I)
  let  $?l = \text{poincare-line } u \ v$ 
  show is-poincare-line  $?l \wedge u \in \text{circline-set } ?l \wedge v \in \text{circline-set } ?l$ 
    using assms
    unfolding circline-set-def
    by auto
next
  fix  $l$ 

```

```

assume is-poincare-line  $l \wedge u \in \text{circline-set } l \wedge v \in \text{circline-set } l$ 
thus  $l = \text{poincare-line } u \ v$ 
  using unique-poincare-line-general assms
  by auto
qed

```

#### 4.2.4 Some consequences of line uniqueness

H-line  $uv$  is the same as the h-line  $vu$ .

```

lemma poincare-line-sym:
assumes  $u \in \text{unit-disc } v \in \text{unit-disc } u \neq v$ 
shows  $\text{poincare-line } u \ v = \text{poincare-line } v \ u$ 
using assms
using unique-poincare-line[of  $u \ v$  poincare-line  $v \ u$ ]
by simp

```

```

lemma poincare-line-sym-general:
assumes  $u \neq v \ u \neq \text{inversion } v$ 
shows  $\text{poincare-line } u \ v = \text{poincare-line } v \ u$ 
using assms
using unique-poincare-line-general[of  $u \ v$  poincare-line  $v \ u$ ]
by simp

```

Each h-line is the h-line constructed out of its two arbitrary different points.

```

lemma ex-poincare-line-points:
assumes is-poincare-line  $H$ 
shows  $\exists u \ v. u \in \text{unit-disc } \wedge v \in \text{unit-disc } \wedge u \neq v \wedge H = \text{poincare-line } u \ v$ 
using assms
using ex-is-poincare-line-points
using unique-poincare-line[where  $l=H$ ]
by fastforce

```

If an h-line contains two different points on x-axis/y-axis then it is the x-axis/y-axis.

```

lemma poincare-line-0-real-is-x-axis:
assumes  $x \in \text{circline-set } x\text{-axis } x \neq 0_h \ x \neq \infty_h$ 
shows  $\text{poincare-line } 0_h \ x = x\text{-axis}$ 
using assms
using is-poincare-line-0-real-is-x-axis[of  $\text{poincare-line } 0_h \ x \ x$ ]
by auto

```

```

lemma poincare-line-0-imag-is-y-axis:
assumes  $y \in \text{circline-set } y\text{-axis } y \neq 0_h \ y \neq \infty_h$ 
shows  $\text{poincare-line } 0_h \ y = y\text{-axis}$ 
using assms
using is-poincare-line-0-imag-is-y-axis[of  $\text{poincare-line } 0_h \ y \ y$ ]
by auto

```

```

lemma poincare-line-x-axis:
assumes  $x \in \text{unit-disc } y \in \text{unit-disc } x \in \text{circline-set } x\text{-axis } y \in \text{circline-set } x\text{-axis } x \neq y$ 
shows  $\text{poincare-line } x \ y = x\text{-axis}$ 
using assms
using unique-poincare-line
by auto

```

```

lemma poincare-line-minus-one-one [simp]:
shows  $\text{poincare-line } (\text{of-complex } (-1)) (\text{of-complex } 1) = x\text{-axis}$ 

```

**proof** –

```

have  $0_h \in \text{circline-set } (\text{poincare-line } (\text{of-complex } (-1)) (\text{of-complex } 1))$ 
  unfolding circline-set-def
  by simp (transfer, transfer, simp add: vec-cnj-def)
hence  $\text{poincare-line } 0_h (\text{of-complex } 1) = \text{poincare-line } (\text{of-complex } (-1)) (\text{of-complex } 1)$ 
  by (metis is-poincare-line-poincare-line is-poincare-line-trough-zero-trough-infnty not-zero-on-unit-circle of-complex-inj
of-complex-one one-neq-neg-one one-on-unit-circle poincare-line-0-real-is-x-axis poincare-line-circline-set(2) reciprocal-involution
reciprocal-one reciprocal-zero unique-circline-01inf)
thus ?thesis

```

```

using poincare-line-0-real-is-x-axis[of of-complex 1]
by auto
qed

```

#### 4.2.5 Transformations of constructed lines

Unit discs preserving Möbius transformations preserve the h-line construction

```

lemma unit-disc-fix-preserve-poincare-line [simp]:
  assumes unit-disc-fix M u ∈ unit-disc v ∈ unit-disc u ≠ v
  shows poincare-line (moebius-pt M u) (moebius-pt M v) = moebius-circline M (poincare-line u v)
proof (rule unique-poincare-line[symmetric])
  show moebius-pt M u ≠ moebius-pt M v
    using ⟨u ≠ v⟩
    by auto
next
  show moebius-pt M u ∈ circline-set (moebius-circline M (poincare-line u v))
    moebius-pt M v ∈ circline-set (moebius-circline M (poincare-line u v))
    unfolding circline-set-def
    using moebius-circline[of M poincare-line u v] ⟨u ≠ v⟩
    by auto
next
  from assms(1) have unit-circle-fix M
    by simp
  thus is-poincare-line (moebius-circline M (poincare-line u v))
    using unit-circle-fix-preserve-is-poincare-line assms
    by auto
next
  show moebius-pt M u ∈ unit-disc moebius-pt M v ∈ unit-disc
    using assms(2-3) unit-disc-fix-iff[OF assms(1)]
    by auto
qed

```

Conjugate preserve the h-line construction

```

lemma conjugate-preserve-poincare-line [simp]:
  assumes u ∈ unit-disc v ∈ unit-disc u ≠ v
  shows poincare-line (conjugate u) (conjugate v) = conjugate-circline (poincare-line u v)
proof -
  have conjugate u ≠ conjugate v
    using ⟨u ≠ v⟩
    by (auto simp add: conjugate-inj)
  moreover
  have conjugate u ∈ unit-disc conjugate v ∈ unit-disc
    using assms
    by auto
  moreover
  have conjugate u ∈ circline-set (conjugate-circline (poincare-line u v))
    conjugate v ∈ circline-set (conjugate-circline (poincare-line u v))
    using ⟨u ≠ v⟩
    by simp-all
  moreover
  have is-poincare-line (conjugate-circline (poincare-line u v))
    using is-poincare-line-poincare-line[OF ⟨u ≠ v⟩]
    by simp
  ultimately
  show ?thesis
    using unique-poincare-line[of conjugate u conjugate v conjugate-circline (poincare-line u v)]
    by simp
qed

```

#### 4.2.6 Collinear points and h-lines

```

lemma poincare-collinear3-poincare-line-general:
  assumes poincare-collinear {a, a1, a2} a1 ≠ a2 a1 ≠ inversion a2
  shows a ∈ circline-set (poincare-line a1 a2)
  using assms
  using poincare-collinear-def unique-poincare-line-general

```



by auto

**lemma** *poincare-line-poincare-collinear3-general*:  
assumes  $a \in \text{circline-set } (\text{poincare-line } a1 \ a2)$   $a1 \neq a2$   
shows *poincare-collinear*  $\{a, a1, a2\}$   
using *assms*  
unfolding *poincare-collinear-def*  
by (*rule-tac*  $x=\text{poincare-line } a1 \ a2$  in *exI*, *simp*)

**lemma** *poincare-collinear3-poincare-lines-equal-general*:  
assumes *poincare-collinear*  $\{a, a1, a2\}$   $a \neq a1$   $a \neq a2$   $a \neq \text{inversion } a1$   $a \neq \text{inversion } a2$   
shows *poincare-line*  $a \ a1 = \text{poincare-line } a \ a2$   
using *assms*  
using *unique-poincare-line-general*[*of*  $a \ a2$  *poincare-line*  $a \ a1$ ]  
by (*simp* add: *insert-commute* *poincare-collinear3-poincare-line-general*)

#### 4.2.7 Points collinear with $0_h$

**lemma** *poincare-collinear-zero-iff*:  
assumes *of-complex*  $y' \in \text{unit-disc}$  and *of-complex*  $z' \in \text{unit-disc}$  and  
 $y' \neq z'$  and  $y' \neq 0$  and  $z' \neq 0$   
shows *poincare-collinear*  $\{0_h, \text{of-complex } y', \text{of-complex } z'\} \longleftrightarrow$   
 $y' * \text{cnj } z' = \text{cnj } y' * z'$  (is ?lhs  $\longleftrightarrow$  ?rhs)

**proof**—

have *of-complex*  $y' \neq \text{of-complex } z'$   
using *assms*  
using *of-complex-inj*  
by *blast*

show ?thesis

**proof**

assume ?lhs

hence  $0_h \in \text{circline-set } (\text{poincare-line } (\text{of-complex } y') (\text{of-complex } z'))$   
using *unique-poincare-line*[*of* *of-complex*  $y'$  *of-complex*  $z'$ ]  
using *assms*  $\langle \text{of-complex } y' \neq \text{of-complex } z' \rangle$   
unfolding *poincare-collinear-def*  
by *auto*

moreover

let ?mix =  $y' * \text{cnj } z' - \text{cnj } y' * z'$

have *is-real*  $(i * ?mix)$

using *eq-cnj-iff-real*[*of* ?mix]

by *auto*

hence  $y' * \text{cnj } z' = \text{cnj } y' * z' \longleftrightarrow \text{Re } (i * ?mix) = 0$

using *complex.expand*[*of*  $i * ?mix \ 0$ ]

by (*metis* *complex-i-not-zero* *eq-iff-diff-eq-0* *mult-eq-0-iff* *zero-complex.simps(1)* *zero-complex.simps(2)*)

ultimately

show ?rhs

using  $\langle y' \neq z' \rangle \langle y' \neq 0 \rangle \langle z' \neq 0 \rangle$

unfolding *circline-set-def*

by *simp* (*transfer*, *transfer*, *auto* *simp* add: *vec-cnj-def* *split: if-split-asm*, *metis* *Re-complex-of-real* *Re-mult-real*

*Im-complex-of-real*)

**next**

assume ?rhs

thus ?lhs

using *assms*  $\langle \text{of-complex } y' \neq \text{of-complex } z' \rangle$

unfolding *poincare-collinear-def*

unfolding *circline-set-def*

apply (*rule-tac*  $x=\text{poincare-line } (\text{of-complex } y') (\text{of-complex } z')$  in *exI*)

apply *auto*

apply (*transfer*, *transfer*, *simp* add: *vec-cnj-def*)

done

**qed**

**qed**

**lemma** *poincare-collinear-zero-polar-form*:  
assumes *poincare-collinear*  $\{0_h, \text{of-complex } x, \text{of-complex } y\}$  and

```

     $x \neq 0$  and  $y \neq 0$  and of-complex  $x \in \text{unit-disc}$  and of-complex  $y \in \text{unit-disc}$ 
  shows  $\exists \varphi \text{ rx ry. } x = \text{cor rx} * \text{cis } \varphi \wedge y = \text{cor ry} * \text{cis } \varphi \wedge \text{rx} \neq 0 \wedge \text{ry} \neq 0$ 
proof-
  from  $\langle x \neq 0 \rangle \langle y \neq 0 \rangle$  obtain  $\varphi \varphi' \text{ rx ry}$  where
    polar:  $x = \text{cor rx} * \text{cis } \varphi \wedge y = \text{cor ry} * \text{cis } \varphi'$  and  $\varphi = \text{Arg } x \wedge \varphi' = \text{Arg } y$ 
    by (metis cmod-cis)
  hence  $\text{rx} \neq 0 \wedge \text{ry} \neq 0$ 
    using  $\langle x \neq 0 \rangle \langle y \neq 0 \rangle$ 
    by auto
  have of-complex  $y \in \text{circline-set (poincare-line } 0_h \text{ (of-complex } x))$ 
    using assms
    using unique-poincare-line[of  $0_h$  of-complex  $x$ ]
    unfolding poincare-collinear-def
    unfolding circline-set-def
    using of-complex-zero-iff
    by fastforce
  hence  $\text{cnj } x * y = x * \text{cnj } y$ 
    using  $\langle x \neq 0 \rangle \langle y \neq 0 \rangle$ 
    unfolding circline-set-def
    by simp (transfer, transfer, simp add: vec-cnj-def field-simps)
  hence  $\text{cis}(\varphi' - \varphi) = \text{cis}(\varphi - \varphi')$ 
    using polar  $\langle \text{rx} \neq 0 \rangle \langle \text{ry} \neq 0 \rangle$ 
    by (simp add: cis-mult)
  hence  $\sin(\varphi' - \varphi) = 0$ 
    using cis-diff-cis-opposite[of  $\varphi' - \varphi$ ]
    by simp
  then obtain  $k :: \text{int}$  where  $\varphi' - \varphi = k * \text{pi}$ 
    using sin-zero-iff-int2[of  $\varphi' - \varphi$ ]
    by auto
  hence *:  $\varphi' = \varphi + k * \text{pi}$ 
    by simp
  show ?thesis
proof (cases even  $k$ )
  case True
    then obtain  $k'$  where  $k = 2 * k'$ 
      using evenE by blast
    hence  $\text{cis } \varphi = \text{cis } \varphi'$ 
      using * cos-periodic-int sin-periodic-int
      by (simp add: cis.ctr field-simps)
    thus ?thesis
      using polar  $\langle \text{rx} \neq 0 \rangle \langle \text{ry} \neq 0 \rangle$ 
      by (rule-tac  $x=\varphi$  in  $\text{exI}$ , rule-tac  $x=\text{rx}$  in  $\text{exI}$ , rule-tac  $x=\text{ry}$  in  $\text{exI}$ ) simp
  case False
    then obtain  $k'$  where  $k = 2 * k' + 1$ 
      using oddE by blast
    hence  $\text{cis } \varphi = - \text{cis } \varphi'$ 
      using * cos-periodic-int sin-periodic-int
      by (simp add: cis.ctr complex-minus field-simps)
    thus ?thesis
      using polar  $\langle \text{rx} \neq 0 \rangle \langle \text{ry} \neq 0 \rangle$ 
      by (rule-tac  $x=\varphi$  in  $\text{exI}$ , rule-tac  $x=\text{rx}$  in  $\text{exI}$ , rule-tac  $x=-\text{ry}$  in  $\text{exI}$ ) simp
qed
qed

end
theory Poincare-Lines-Ideal-Points
imports Poincare-Lines
begin

```

### 4.3 Ideal points of h-lines

*Ideal points* of an h-line are points where the h-line intersects the unit disc.

### 4.3.1 Calculation of ideal points

We decided to define ideal points constructively, i.e., we calculate the coordinates of ideal points for a given h-line explicitly. Namely, if the h-line is determined by  $A$  and  $B$ , the two intersection points are

$$\frac{B}{|B|^2} \left( -A \pm i \cdot \sqrt{|B|^2 - A^2} \right).$$

**definition** *calc-ideal-point1-cvec* :: *complex*  $\Rightarrow$  *complex*  $\Rightarrow$  *complex-vec* **where**

[simp]: *calc-ideal-point1-cvec*  $A B =$   
 (let *discr* =  $\text{Re} ((\text{cmod } B)^2 - (\text{Re } A)^2)$  in  
 ( $B * (-A - i * \text{sqrt}(\text{discr}))$ ,  $(\text{cmod } B)^2$ ))

**definition** *calc-ideal-point2-cvec* :: *complex*  $\Rightarrow$  *complex*  $\Rightarrow$  *complex-vec* **where**

[simp]: *calc-ideal-point2-cvec*  $A B =$   
 (let *discr* =  $\text{Re} ((\text{cmod } B)^2 - (\text{Re } A)^2)$  in  
 ( $B * (-A + i * \text{sqrt}(\text{discr}))$ ,  $(\text{cmod } B)^2$ ))

**definition** *calc-ideal-points-cmat-cvec* :: *complex-mat*  $\Rightarrow$  *complex-vec set* **where**

[simp]: *calc-ideal-points-cmat-cvec*  $H =$   
 (if *is-poincare-line-cmat*  $H$  then  
 let  $(A, B, C, D) = H$   
 in {*calc-ideal-point1-cvec*  $A B$ , *calc-ideal-point2-cvec*  $A B$ }  
 else  
 $\{(-1, 1), (1, 1)\}$ )

**lift-definition** *calc-ideal-points-clmat-hcoords* :: *circline-mat*  $\Rightarrow$  *complex-homo-coords set* **is** *calc-ideal-points-cmat-cvec*  
**by** (*auto simp add: Let-def split: if-split-asm*)

**lift-definition** *calc-ideal-points* :: *circline*  $\Rightarrow$  *complex-homo set* **is** *calc-ideal-points-clmat-hcoords*

**proof** *transfer*

**fix**  $H1 H2$

**assume** *hh*: *hermitean*  $H1 \wedge H1 \neq \text{mat-zero}$  *hermitean*  $H2 \wedge H2 \neq \text{mat-zero}$

**obtain**  $A1 B1 C1 D1 A2 B2 C2 D2$  **where**  $*$ :  $H1 = (A1, B1, C1, D1)$   $H2 = (A2, B2, C2, D2)$

**by** (*cases H1, cases H2, auto*)

**assume** *circline-eq-cmat*  $H1 H2$

**then obtain**  $k$  **where**  $k: k \neq 0$   $H2 = \text{cor } k *_{sm} H1$

**by** *auto*

**thus** *rel-set* ( $\approx_v$ ) (*calc-ideal-points-cmat-cvec*  $H1$ ) (*calc-ideal-points-cmat-cvec*  $H2$ )

**proof** (*cases is-poincare-line-cmat H1*)

**case** *True*

**hence** *is-poincare-line-cmat*  $H2$

**using**  $k * \text{hermitean-mult-real}[of H1 k] hh$

**by** (*auto simp add: power2-eq-square norm-mult*)

**have**  $*$ :  $\text{sqrt} (|k| * \text{cmod } B1 * (|k| * \text{cmod } B1) - k * \text{Re } D1 * (k * \text{Re } D1)) =$   
 $|k| * \text{sqrt}(\text{cmod } B1 * \text{cmod } B1 - \text{Re } D1 * \text{Re } D1)$

**proof**–

**have**  $|k| * \text{cmod } B1 * (|k| * \text{cmod } B1) - k * \text{Re } D1 * (k * \text{Re } D1) =$   
 $k^2 * (\text{cmod } B1 * \text{cmod } B1 - \text{Re } D1 * \text{Re } D1)$

**by** (*simp add: power2-eq-square field-simps*)

**thus** *?thesis*

**by** (*simp add: real-sqrt-mult*)

**qed**

**show** *?thesis*

**using**  $\langle \text{is-poincare-line-cmat } H1 \rangle \langle \text{is-poincare-line-cmat } H2 \rangle$

**using**  $*$   $k$

**apply** (*simp add: Let-def*)

**apply** *safe*

**apply** (*simp add: power2-eq-square rel-set-def norm-mult*)

**apply** *safe*

**apply** (*cases*  $k > 0$ )

**apply** (*rule-tac*  $x = (\text{cor } k)^2$  **in** *exI*)

**apply** (*subst*  $**$ )

**apply** (*simp add: power2-eq-square field-simps*)

**apply** (*erule notE, rule-tac*  $x = (\text{cor } k)^2$  **in** *exI*)

**apply** (*subst*  $**$ )

```

    apply (simp add: power2-eq-square field-simps)
  apply (cases k > 0)
  apply (erule notE, rule-tac x=(cor k)2 in exI)
  apply (subst **)
  apply (simp add: power2-eq-square field-simps)
  apply (rule-tac x=(cor k)2 in exI)
  apply (subst **)
  apply (simp add: power2-eq-square field-simps)
  apply (cases k > 0)
  apply (rule-tac x=(cor k)2 in exI)
  apply (subst **)
  apply (simp add: power2-eq-square field-simps)
  apply (erule notE, rule-tac x=(cor k)2 in exI)
  apply (subst **)
  apply (simp add: power2-eq-square field-simps)
  apply (rule-tac x=(cor k)2 in exI)
  apply (cases k > 0)
  apply (erule notE, rule-tac x=(cor k)2 in exI)
  apply (subst **)
  apply (simp add: power2-eq-square field-simps)
  apply (subst **)
  apply (simp add: power2-eq-square field-simps)
done
next
case False
hence ¬ is-poincare-line-cmat H2
  using k * hermitean-mult-real[of H1 k] hh
  by (auto simp add: power2-eq-square norm-mult)
have rel-set (≈v) {(- 1, 1), (1, 1)} {(- 1, 1), (1, 1)}
  by (simp add: rel-set-def)
thus ?thesis
  using ⟨¬ is-poincare-line-cmat H1⟩ ⟨¬ is-poincare-line-cmat H2⟩
  using *
  by (auto simp add: Let-def)
qed
qed

```

Correctness of the calculation

We show that for every h-line its two calculated ideal points are different and are on the intersection of that line and the unit circle.

Calculated ideal points are on the unit circle

**lemma** *calc-ideal-point-1-unit:*

```

assumes is-real A (cmod B)2 > (cmod A)2
assumes (z1, z2) = calc-ideal-point1-cvec A B
shows z1 * cnj z1 = z2 * cnj z2
proof -
  let ?discr = Re ((cmod B)2 - (Re A)2)
  have ?discr > 0
    using assms
    by (simp add: cmod-power2)
  have (B*(-A - i*sqrt(?discr))) * cnj (B*(-A - i*sqrt(?discr))) = (B * cnj B) * (A2 + cor (abs ?discr))
    using ⟨is-real A⟩ eq-cnj-iff-real[of A]
    by (simp add: field-simps power2-eq-square)
  also have ... = (B * cnj B) * (cmod B)2
    using ⟨?discr > 0⟩
    using assms
    using complex-of-real-Re[of (cmod B)2 - (Re A)2] complex-of-real-Re[of A] ⟨is-real A⟩
    by (simp add: power2-eq-square)
  also have ... = (cmod B)2 * cnj ((cmod B)2)
    using complex-cnj-complex-of-real complex-mult-cnj-cmod
    by presburger
  finally show ?thesis
    using assms
    by simp

```

qed

lemma *calc-ideal-point-2-unit*:

assumes *is-real*  $A$   $(\text{cmod } B)^2 > (\text{cmod } A)^2$

assumes  $(z1, z2) = \text{calc-ideal-point2-cvec } A B$

shows  $z1 * \text{cnj } z1 = z2 * \text{cnj } z2$

proof –

let  $?discr = \text{Re } ((\text{cmod } B)^2 - (\text{Re } A)^2)$

have  $?discr > 0$

using *assms*

by (*simp add: cmod-power2*)

have  $(B * (-A + i * \text{sqrt}(?discr))) * \text{cnj } (B * (-A + i * \text{sqrt}(?discr))) = (B * \text{cnj } B) * (A^2 + \text{cor } (\text{abs } ?discr))$

using  $\langle \text{is-real } A \rangle \text{ eq-cnj-iff-real[of } A]$

by (*simp add: field-simps power2-eq-square*)

also have  $\dots = (B * \text{cnj } B) * (\text{cmod } B)^2$

using  $\langle ?discr > 0 \rangle$

using *assms*

using *complex-of-real-Re[of  $(\text{cmod } B)^2 - (\text{Re } A)^2$ ] complex-of-real-Re[of } A] \langle \text{is-real } A \rangle*

by (*simp add: power2-eq-square*)

also have  $\dots = (\text{cmod } B)^2 * \text{cnj } ((\text{cmod } B)^2)$

using *complex-cnj-complex-of-real complex-mult-cnj-cmod*

by *presburger*

finally show *?thesis*

using *assms*

by *simp*

qed

lemma *calc-ideal-points-on-unit-circle*:

shows  $\forall z \in \text{calc-ideal-points } H. z \in \text{circline-set unit-circle}$

unfolding *circline-set-def*

apply *simp*

proof (*transfer, transfer*)

fix  $H$

assume *hh*: *hermitean*  $H \wedge H \neq \text{mat-zero}$

obtain  $A B C D$  where  $*$ :  $H = (A, B, C, D)$

by (*cases H, auto*)

have  $\forall (z1, z2) \in \text{calc-ideal-points-cmat-cvec } H. z1 * \text{cnj } z1 = z2 * \text{cnj } z2$

using *hermitean-elems[of } A B C D]*

unfolding *calc-ideal-points-cmat-cvec-def*

using *calc-ideal-point-1-unit[of } A B]*

using *calc-ideal-point-2-unit[of } A B]*

using *hh \**

apply (*cases calc-ideal-point1-cvec } A B, cases calc-ideal-point2-cvec } A B)*

apply (*auto simp add: Let-def simp del: calc-ideal-point1-cvec-def calc-ideal-point2-cvec-def*)

done

thus *Ball (calc-ideal-points-cmat-cvec } H) (on-circline-cmat-cvec unit-circle-cmat)*

using *on-circline-cmat-cvec-unit*

by (*auto simp del: on-circline-cmat-cvec-def calc-ideal-points-cmat-cvec-def*)

qed

Calculated ideal points are on the h-line

lemma *calc-ideal-point1-sq*:

assumes  $(z1, z2) = \text{calc-ideal-point1-cvec } A B$  *is-real*  $A$   $(\text{cmod } B)^2 > (\text{cmod } A)^2$

shows  $z1 * \text{cnj } z1 + z2 * \text{cnj } z2 = 2 * (B * \text{cnj } B)^2$

proof –

let  $?discr = \text{Re } ((\text{cmod } B)^2 - (\text{Re } A)^2)$

have  $?discr > 0$

using *assms*

by (*simp add: cmod-power2*)

have  $z1 * \text{cnj } z1 = (B * \text{cnj } B) * (-A + i * \text{sqrt}(?discr)) * (-A - i * \text{sqrt}(?discr))$

using *assms eq-cnj-iff-real[of } A]*

by (*simp*)

also have  $\dots = (B * \text{cnj } B) * (A^2 + ?discr)$

using *complex-of-real-Re[of } A] \langle \text{is-real } A \rangle \langle ?discr > 0 \rangle*

by (*simp add: power2-eq-square field-simps*)

finally

```

have z1 * cnj z1 = (B * cnj B)2
  using complex-of-real-Re[of (cmod B)2 - (Re A)2] complex-of-real-Re[of A] ‹is-real A›
  using complex-mult-cnj-cmod[of B]
  by (simp add: power2-eq-square)
moreover
have z2 * cnj z2 = (B * cnj B)2
  using assms
  by simp
ultimately
show ?thesis
  by simp
qed

```

lemma calc-ideal-point2-sq:

```

assumes (z1, z2) = calc-ideal-point2-cvec A B is-real A (cmod B)2 > (cmod A)2
shows z1 * cnj z1 + z2 * cnj z2 = 2 * (B * cnj B)2

```

proof -

```

let ?discr = Re ((cmod B)2 - (Re A)2)
have ?discr > 0
  using assms
  by (simp add: cmod-power2)
have z1 * cnj z1 = (B * cnj B) * (-A + i*sqrt(?discr)) * (-A - i*sqrt(?discr))
  using assms eq-cnj-iff-real[of A]
  by simp
also have ... = (B * cnj B) * (A2 + ?discr)
  using complex-of-real-Re[of A] ‹is-real A› ‹?discr > 0›
  by (simp add: power2-eq-square field-simps)

```

finally

```

have z1 * cnj z1 = (B * cnj B)2
  using complex-of-real-Re[of (cmod B)2 - (Re A)2] complex-of-real-Re[of A] ‹is-real A›
  using complex-mult-cnj-cmod[of B]
  by (simp add: power2-eq-square)
moreover
have z2 * cnj z2 = (B * cnj B)2
  using assms
  by simp
ultimately
show ?thesis
  by simp

```

qed

lemma calc-ideal-point1-mix:

```

assumes (z1, z2) = calc-ideal-point1-cvec A B is-real A (cmod B)2 > (cmod A)2
shows B * cnj z1 * z2 + cnj B * z1 * cnj z2 = - 2 * A * (B * cnj B)2

```

proof -

```

have B*cnj z1 + cnj B*z1 = -2*A*B*cnj B
  using assms eq-cnj-iff-real[of A]
  by (simp, simp add: field-simps)
moreover
have cnj z2 = z2
  using assms
  by simp
hence B*cnj z1*z2 + cnj B*z1*cnj z2 = (B*cnj z1 + cnj B*z1)*z2
  by (simp add: field-simps)
ultimately
have B*cnj z1*z2 + cnj B*z1*cnj z2 = -2*A*(B*cnj B)*z2
  by simp
also have ... = -2*A*(B * cnj B)2
  using assms
  using complex-mult-cnj-cmod[of B]
  by (simp add: power2-eq-square)

```

finally

```
show ?thesis
```

qed

```

lemma calc-ideal-point2-mix:
  assumes (z1, z2) = calc-ideal-point2-cvec A B is-real A (cmod B)2 > (cmod A)2
  shows B * cnj z1 * z2 + cnj B * z1 * cnj z2 = - 2 * A * (B * cnj B)2
proof-
  have B*cnj z1 + cnj B*z1 = -2*A*B*cnj B
    using assms eq-cnj-iff-real[of A]
    by (simp, simp add: field-simps)
  moreover
  have cnj z2 = z2
    using assms
    by simp
  hence B*cnj z1*z2 + cnj B*z1*cnj z2 = (B*cnj z1 + cnj B*z1)*z2
    by (simp add: field-simps)
  ultimately
  have B*cnj z1*z2 + cnj B*z1*cnj z2 = -2*A*(B* cnj B)*z2
    by simp
  also have ... = -2*A*(B * cnj B)2
    using assms
    using complex-mult-cnj-cmod[of B]
    by (simp add: power2-eq-square)
  finally
  show ?thesis
  .
qed

lemma calc-ideal-point1-on-circline:
  assumes (z1, z2) = calc-ideal-point1-cvec A B is-real A (cmod B)2 > (cmod A)2
  shows A*z1*cnj z1 + B*cnj z1*z2 + cnj B*z1*cnj z2 + A*z2*cnj z2 = 0 (is ?lhs = 0)
proof-
  have ?lhs = A * (z1 * cnj z1 + z2 * cnj z2) + (B * cnj z1 * z2 + cnj B * z1 * cnj z2)
    by (simp add: field-simps)
  also have ... = 2*A*(B*cnj B)2 + (-2*A*(B*cnj B)2)
    using calc-ideal-point1-sq[OF assms]
    using calc-ideal-point1-mix[OF assms]
    by simp
  finally
  show ?thesis
    by simp
qed

lemma calc-ideal-point2-on-circline:
  assumes (z1, z2) = calc-ideal-point2-cvec A B is-real A (cmod B)2 > (cmod A)2
  shows A*z1*cnj z1 + B*cnj z1*z2 + cnj B*z1*cnj z2 + A*z2*cnj z2 = 0 (is ?lhs = 0)
proof-
  have ?lhs = A * (z1 * cnj z1 + z2 * cnj z2) + (B * cnj z1 * z2 + cnj B * z1 * cnj z2)
    by (simp add: field-simps)
  also have ... = 2*A*(B*cnj B)2 + (-2*A*(B*cnj B)2)
    using calc-ideal-point2-sq[OF assms]
    using calc-ideal-point2-mix[OF assms]
    by simp
  finally
  show ?thesis
    by simp
qed

lemma calc-ideal-points-on-circline:
  assumes is-poincare-line H
  shows  $\forall z \in \text{calc-ideal-points } H. z \in \text{circline-set } H$ 
  using assms
  unfolding circline-set-def
  apply simp
proof (transfer, transfer)
  fix H
  assume hh: hermitean H  $\wedge$  H  $\neq$  mat-zero
  obtain A B C D where *: H = (A, B, C, D)
    by (cases H, auto)

```

```

obtain z11 z12 z21 z22 where **: (z11, z12) = calc-ideal-point1-cvec A B (z21, z22) = calc-ideal-point2-cvec A B
by (cases calc-ideal-point1-cvec A B, cases calc-ideal-point2-cvec A B) auto

assume is-poincare-line-cmat H
hence  $\forall (z1, z2) \in \text{calc-ideal-points-cmat-cvec } H. A*z1*cnj z1 + B*cnj z1*z2 + C*z1*cnj z2 + D*z2*cnj z2 = 0$ 
using * ** hh
using hermitean-elems[of A B C D]
using calc-ideal-point1-on-circline[of z11 z12 A B]
using calc-ideal-point2-on-circline[of z21 z22 A B]
by (auto simp del: calc-ideal-point1-cvec-def calc-ideal-point2-cvec-def)
thus Ball (calc-ideal-points-cmat-cvec H) (on-circline-cmat-cvec H)
using on-circline-cmat-cvec-circline-equation *
by (auto simp del: on-circline-cmat-cvec-def calc-ideal-points-cmat-cvec-def simp add: field-simps)
qed

```

Calculated ideal points of an h-line are different

```

lemma calc-ideal-points-cvec-different [simp]:
assumes (cmod B)2 > (cmod A)2 is-real A
shows  $\neg (\text{calc-ideal-point1-cvec } A B \approx_v \text{calc-ideal-point2-cvec } A B)$ 
using assms
by (auto) (auto simp add: cmod-def)

```

```

lemma calc-ideal-points-different:
assumes is-poincare-line H
shows  $\exists i1 \in (\text{calc-ideal-points } H). \exists i2 \in (\text{calc-ideal-points } H). i1 \neq i2$ 
using assms
proof (transfer, transfer)
fix H
assume hh: hermitean H  $\wedge$  H  $\neq$  mat-zero is-poincare-line-cmat H
obtain A B C D where *: H = (A, B, C, D)
by (cases H, auto)
hence is-real A using hh hermitean-elems by auto
thus  $\exists i1 \in \text{calc-ideal-points-cmat-cvec } H. \exists i2 \in \text{calc-ideal-points-cmat-cvec } H. \neg i1 \approx_v i2$ 
using * hh calc-ideal-points-cvec-different[of A B]
apply (rule-tac x=calc-ideal-point1-cvec A B in bestI)
apply (rule-tac x=calc-ideal-point2-cvec A B in bestI)
by auto
qed

```

```

lemma two-calc-ideal-points [simp]:
assumes is-poincare-line H
shows card (calc-ideal-points H) = 2
proof –
have  $\exists x \in \text{calc-ideal-points } H. \exists y \in \text{calc-ideal-points } H. \forall z \in \text{calc-ideal-points } H. z = x \vee z = y$ 
by (transfer, transfer, case-tac H, simp del: calc-ideal-point1-cvec-def calc-ideal-point2-cvec-def)
then obtain x y where *: calc-ideal-points H = {x, y}
by auto
moreover
have x  $\neq$  y
using calc-ideal-points-different[OF assms] *
by auto
ultimately
show ?thesis
by auto
qed

```

### 4.3.2 Ideal points

Next we give a genuine definition of ideal points – these are the intersections of the h-line with the unit circle

```

definition ideal-points :: circline  $\Rightarrow$  complex-homo set where
  ideal-points H = circline-intersection H unit-circle

```

Ideal points are on the unit circle and on the h-line

```

lemma ideal-points-on-unit-circle:
shows  $\forall z \in \text{ideal-points } H. z \in \text{circline-set unit-circle}$ 

```



**unfolding** *ideal-points-def circline-intersection-def circline-set-def*  
**by** *simp*

**lemma** *ideal-points-on-circline*:

**shows**  $\forall z \in \text{ideal-points } H. z \in \text{circline-set } H$

**unfolding** *ideal-points-def circline-intersection-def circline-set-def*  
**by** *simp*

For each h-line there are exactly two ideal points

**lemma** *two-ideal-points*:

**assumes** *is-poincare-line H*

**shows**  $\text{card } (\text{ideal-points } H) = 2$

**proof**–

**have**  $H \neq \text{unit-circle}$

**using** *assms not-is-poincare-line-unit-circle*

**by** *auto*

**let**  $?int = \text{circline-intersection } H \text{ unit-circle}$

**obtain**  $i1 \ i2$  **where**  $i1 \in ?int \ i2 \in ?int \ i1 \neq i2$

**using** *calc-ideal-points-on-circline[OF assms]*

**using** *calc-ideal-points-on-unit-circle[of H]*

**using** *calc-ideal-points-different[OF assms]*

**unfolding** *circline-intersection-def circline-set-def*

**by** *auto*

**thus** *?thesis*

**unfolding** *ideal-points-def*

**using** *circline-intersection-at-most-2-points[OF  $H \neq \text{unit-circle}$ ]*

**using** *card-geq-2-iff-contains-2-elems[of ?int]*

**by** *auto*

**qed**

They are exactly the two points that our calculation finds

**lemma** *ideal-points-unique*:

**assumes** *is-poincare-line H*

**shows**  $\text{ideal-points } H = \text{calc-ideal-points } H$

**proof**–

**have**  $\text{calc-ideal-points } H \subseteq \text{ideal-points } H$

**using** *calc-ideal-points-on-circline[OF assms]*

**using** *calc-ideal-points-on-unit-circle[of H]*

**unfolding** *ideal-points-def circline-intersection-def circline-set-def*

**by** *auto*

**moreover**

**have**  $H \neq \text{unit-circle}$

**using** *not-is-poincare-line-unit-circle assms*

**by** *auto*

**hence** *finite*  $(\text{ideal-points } H)$

**using** *circline-intersection-at-most-2-points[of H unit-circle]*

**unfolding** *ideal-points-def*

**by** *auto*

**ultimately**

**show** *?thesis*

**using** *card-subset-eq[of ideal-points H calc-ideal-points H]*

**using** *two-calc-ideal-points[OF assms]*

**using** *two-ideal-points[OF assms]*

**by** *auto*

**qed**

For each h-line we can obtain two different ideal points

**lemma** *obtain-ideal-points*:

**assumes** *is-poincare-line H*

**obtains**  $i1 \ i2$  **where**  $i1 \neq i2 \ \text{ideal-points } H = \{i1, i2\}$

**using** *two-ideal-points[OF assms] card-eq-2-iff-doubleton[of ideal-points H]*

**by** *blast*

Ideal points of each h-line constructed from two points in the disc are different than those two points

**lemma** *ideal-points-different*:

**assumes**  $u \in \text{unit-disc } v \in \text{unit-disc } u \neq v$   
**assumes**  $\text{ideal-points } (\text{poincare-line } u \ v) = \{i1, i2\}$   
**shows**  $i1 \neq i2 \ u \neq i1 \ u \neq i2 \ v \neq i1 \ v \neq i2$   
**proof**–  
**have**  $i1 \in \text{ocircline-set } \text{ounit-circle } i2 \in \text{ocircline-set } \text{ounit-circle}$   
**using**  $\text{assms}(3) \ \text{assms}(4) \ \text{ideal-points-on-unit-circle } \text{is-poincare-line-poincare-line}$   
**by**  $\text{fastforce+}$   
**thus**  $u \neq i1 \ u \neq i2 \ v \neq i1 \ v \neq i2$   
**using**  $\text{assms}(1-2)$   
**using**  $\text{disc-inter-ocircline-set}[of \ \text{ounit-circle}]$   
**unfolding**  $\text{unit-disc-def}$   
**by**  $\text{auto}$   
**show**  $i1 \neq i2$   
**using**  $\text{assms}$   
**by**  $(\text{metis } \text{doubleton-eq-iff } \text{is-poincare-line-poincare-line } \text{obtain-ideal-points})$   
**qed**

H-line is uniquely determined by its ideal points

**lemma**  $\text{ideal-points-line-unique}$ :  
**assumes**  $\text{is-poincare-line } H \ \text{ideal-points } H = \{i1, i2\}$   
**shows**  $H = \text{poincare-line } i1 \ i2$   
**by**  $(\text{smt } \text{assms}(1) \ \text{assms}(2) \ \text{calc-ideal-points-on-unit-circle } \text{circline-set-def } \text{ex-poincare-line-points } \text{ideal-points-different}(1) \ \text{ideal-points-on-circline } \text{ideal-points-unique } \text{insertI1 } \text{insert-commute } \text{inversion-unit-circle } \text{mem-Collect-eq } \text{unique-poincare-line-general})$

Ideal points of some special h-lines

Ideal points of  $x$ -axis

**lemma**  $\text{ideal-points-x-axis}$   
 $[\text{simp}]$ :  $\text{ideal-points } x\text{-axis} = \{\text{of-complex } (-1), \text{of-complex } 1\}$   
**proof**  $(\text{subst } \text{ideal-points-unique}, \text{simp})$   
**have**  $\text{calc-ideal-points-clmat-hcoords } x\text{-axis-clmat} = \{\text{of-complex-hcoords } (-1), \text{of-complex-hcoords } 1\}$   
**by**  $\text{transfer } \text{auto}$   
**thus**  $\text{calc-ideal-points } x\text{-axis} = \{\text{of-complex } (-1), \text{of-complex } 1\}$   
**by**  $(\text{simp } \text{add: } \text{calc-ideal-points.abs-eq } \text{of-complex.abs-eq } x\text{-axis-def})$   
**qed**

Ideal points are proportional vectors only if h-line is a line segment passing through zero

**lemma**  $\text{ideal-points-proportional}$ :  
**assumes**  $\text{is-poincare-line } H \ \text{ideal-points } H = \{i1, i2\} \ \text{to-complex } i1 = \text{cor } k * \text{to-complex } i2$   
**shows**  $0_h \in \text{circline-set } H$   
**proof**–  
**have**  $i1 \neq i2$   
**using**  $\langle \text{ideal-points } H = \{i1, i2\} \rangle$   
**using**  $\langle \text{is-poincare-line } H \rangle \ \text{ex-poincare-line-points } \text{ideal-points-different}(1) \ \text{by } \text{blast}$   
  
**have**  $i1 \in \text{circline-set } \text{unit-circle } i2 \in \text{circline-set } \text{unit-circle}$   
**using**  $\text{assms } \text{calc-ideal-points-on-unit-circle } \text{ideal-points-unique}$   
**by**  $\text{blast+}$   
  
**hence**  $\text{cmod } (\text{cor } k) = 1$   
**using**  $\langle \text{to-complex } i1 = \text{cor } k * \text{to-complex } i2 \rangle$   
**by**  $(\text{metis } (\text{mono-tags}, \text{lifting}) \ \text{circline-set-unit-circle } \text{imageE } \text{mem-Collect-eq } \text{mult.right-neutral } \text{norm-mult } \text{to-complex-of-complex } \text{unit-circle-set-def})$   
**hence**  $k = -1$   
**using**  $\langle \text{to-complex } i1 = \text{cor } k * \text{to-complex } i2 \rangle \ \langle i1 \neq i2 \rangle$   
**using**  $\langle i1 \in \text{circline-set } \text{unit-circle} \rangle \ \langle i2 \in \text{circline-set } \text{unit-circle} \rangle$   
**by**  $(\text{smt } (\text{verit}, \text{best}) \ \text{mult-cancel-right1 } \text{norm-of-real } \text{not-inf-on-unit-circle'' } \text{of-complex-to-complex } \text{of-real-1})$   
  
**have**  $\forall i1 \in \text{calc-ideal-points } H. \ \forall i2 \in \text{calc-ideal-points } H. \ \text{is-poincare-line } H \ \wedge \ i1 \neq i2 \ \wedge \ \text{to-complex } i1 = - \text{to-complex } i2 \ \longrightarrow$   
 $0_h \in \text{circline-set } H$   
**unfolding**  $\text{circline-set-def}$   
**proof**  $(\text{simp}, \text{transfer}, \text{transfer}, \text{safe})$   
**fix**  $A \ B \ C \ D \ i11 \ i12 \ i21 \ i22 \ k$   
**assume**  $H$ :  $\text{hermitean } (A, B, C, D) \ (A, B, C, D) \neq \text{mat-zero}$   
**assume**  $\text{line: } \text{is-poincare-line-cmat } (A, B, C, D)$

**assume**  $i1: (i11, i12) \in \text{calc-ideal-points-cmat-cvec } (A, B, C, D)$   
**assume**  $i2: (i21, i22) \in \text{calc-ideal-points-cmat-cvec } (A, B, C, D)$   
**assume**  $\neg (i11, i12) \approx_v (i21, i22)$   
**assume** *opposite: to-complex-cvec*  $(i11, i12) = - \text{ to-complex-cvec } (i21, i22)$

**let**  $?discr = \text{sqrt } ((\text{cmod } B)^2 - (\text{Re } D)^2)$   
**let**  $?den = (\text{cmod } B)^2$   
**let**  $?i1 = B * (- D - i * ?discr)$   
**let**  $?i2 = B * (- D + i * ?discr)$

**have**  $i11 = ?i1 \vee i11 = ?i2 \ i12 = ?den$   
 $i21 = ?i1 \vee i21 = ?i2 \ i22 = ?den$   
**using**  $i1 \ i2 \ H \ \text{line}$   
**by** *(auto split: if-split-asm)*  
**hence**  $i: i11 = ?i1 \wedge i21 = ?i2 \vee i11 = ?i2 \wedge i21 = ?i1$   
**using**  $\langle \neg (i11, i12) \approx_v (i21, i22) \rangle$   
**by** *auto*

**have**  $?den \neq 0$   
**using** *line*  
**by** *auto*

**hence**  $i11 = - i21$   
**using** *opposite*  $\langle i12 = ?den \rangle \langle i22 = ?den \rangle$   
**by** *(simp add: nonzero-neg-divide-eq-eq2)*

**hence**  $?i1 = - ?i2$   
**using**  $i$   
**by** *(metis add.inverse-inverse)*

**hence**  $D = 0$   
**using**  $\langle ?den \neq 0 \rangle$   
**by** *(simp add: field-simps)*

**thus** *on-circline-cmat-cvec*  $(A, B, C, D) \ 0_v$   
**by** *(simp add: vec-cnj-def)*

**qed**

**thus** *?thesis*  
**using** *assms*  $\langle k = -1 \rangle$   
**using** *calc-ideal-points-different ideal-points-unique*  
**by** *fastforce*

**qed**

Transformations of ideal points

Möbius transformations that fix the unit disc when acting on h-lines map their ideal points to ideal points.

**lemma** *ideal-points-moebius-circline* [*simp*]:

**assumes** *unit-circle-fix*  $M$  *is-poincare-line*  $H$

**shows** *ideal-points*  $(\text{moebius-circline } M \ H) = (\text{moebius-pt } M) \text{ ' } (\text{ideal-points } H)$  (**is**  $?I' = ?M \text{ ' } ?I$ )

**proof**–

**obtain**  $i1 \ i2$  **where**  $*$ :  $i1 \neq i2 \ ?I = \{i1, i2\}$

**using** *assms*(2)

**by** *(rule obtain-ideal-points)*

**let**  $?Mi1 = ?M \ i1$  **and**  $?Mi2 = ?M \ i2$

**have**  $?Mi1 \in ?M \text{ ' } (\text{circline-set } H)$

$?Mi2 \in ?M \text{ ' } (\text{circline-set } H)$

$?Mi1 \in ?M \text{ ' } (\text{circline-set unit-circle})$

$?Mi2 \in ?M \text{ ' } (\text{circline-set unit-circle})$

**using**  $*$

**unfolding** *ideal-points-def circline-intersection-def circline-set-def*

**by** *blast+*

**hence**  $?Mi1 \in ?I'$

$?Mi2 \in ?I'$

**using** *unit-circle-fix-iff*[of  $M$ ] *assms*

```

  unfolding ideal-points-def circline-intersection-def circline-set-def
  by (metis mem-Collect-eq moebius-circline)+
moreover
have ?Mi1 ≠ ?Mi2
  using bij-moebius-pt[of M] *
  using moebius-pt-invert by blast
moreover
have is-poincare-line (moebius-circline M H)
  using assms unit-circle-fix-preserve-is-poincare-line
  by simp
ultimately
have ?I' = {?Mi1, ?Mi2}
  using two-ideal-points[of moebius-circline M H]
  using card-eq-2-doubleton[of ?I' ?Mi1 ?Mi2]
  by simp
thus ?thesis
  using *(2)
  by auto
qed

```

```

lemma ideal-points-poincare-line-moebius [simp]:
  assumes unit-disc-fix M u ∈ unit-disc v ∈ unit-disc u ≠ v
  assumes ideal-points (poincare-line u v) = {i1, i2}
  shows ideal-points (poincare-line (moebius-pt M u) (moebius-pt M v)) = {moebius-pt M i1, moebius-pt M i2}
  using assms
  by auto

```

Conjugation also maps ideal points to ideal points

```

lemma ideal-points-conjugate [simp]:
  assumes is-poincare-line H
  shows ideal-points (conjugate-circline H) = conjugate ' (ideal-points H) (is ?I' = ?M ' ?I)
proof -
  obtain i1 i2 where *: i1 ≠ i2 ?I = {i1, i2}
  using assms
  by (rule obtain-ideal-points)
  let ?Mi1 = ?M i1 and ?Mi2 = ?M i2
  have ?Mi1 ∈ ?M ' (circline-set H)
    ?Mi2 ∈ ?M ' (circline-set H)
    ?Mi1 ∈ ?M ' (circline-set unit-circle)
    ?Mi2 ∈ ?M ' (circline-set unit-circle)
  using *
  unfolding ideal-points-def circline-intersection-def circline-set-def
  by blast+
  hence ?Mi1 ∈ ?I'
    ?Mi2 ∈ ?I'
  unfolding ideal-points-def circline-intersection-def circline-set-def
  using circline-set-conjugate-circline circline-set-def conjugate-unit-circle-set
  by blast+
  moreover
  have ?Mi1 ≠ ?Mi2
  using ⟨i1 ≠ i2⟩
  by (auto simp add: conjugate-inj)
  moreover
  have is-poincare-line (conjugate-circline H)
  using assms
  by simp
  ultimately
  have ?I' = {?Mi1, ?Mi2}
  using two-ideal-points[of conjugate-circline H]
  using card-eq-2-doubleton[of ?I' ?Mi1 ?Mi2]
  by simp
  thus ?thesis
  using *(2)
  by auto
qed

```

```

lemma ideal-points-poincare-line-conjugate [simp]:
  assumes  $u \in \text{unit-disc } v \in \text{unit-disc } u \neq v$ 
  assumes ideal-points (poincare-line  $u v$ ) = {i1, i2}
  shows ideal-points (poincare-line (conjugate  $u$ ) (conjugate  $v$ )) = {conjugate i1, conjugate i2}
  using assms
  by auto

```

**end**

**theory** *Poincare-Distance*

**imports** *Poincare-Lines-Ideal-Points Hyperbolic-Functions*

**begin**

## 5 H-distance in the Poincaré model

Informally, the *h-distance* between the two h-points is defined as the absolute value of the logarithm of the cross ratio between those two points and the two ideal points.

**abbreviation** *Re-cross-ratio* **where** *Re-cross-ratio*  $z u v w \equiv \text{Re } (\text{to-complex } (\text{cross-ratio } z u v w))$

**definition** *calc-poincare-distance* :: *complex-homo*  $\Rightarrow$  *complex-homo*  $\Rightarrow$  *complex-homo*  $\Rightarrow$  *complex-homo*  $\Rightarrow$  *real* **where**  
 [simp]: *calc-poincare-distance*  $u i1 v i2 = \text{abs } (\ln (\text{Re-cross-ratio } u i1 v i2))$

**definition** *poincare-distance-pred* :: *complex-homo*  $\Rightarrow$  *complex-homo*  $\Rightarrow$  *real*  $\Rightarrow$  *bool* **where**

[simp]: *poincare-distance-pred*  $u v d \longleftrightarrow$   
 $(u = v \wedge d = 0) \vee (u \neq v \wedge (\forall i1 i2. \text{ideal-points } (\text{poincare-line } u v) = \{i1, i2\} \longrightarrow d = \text{calc-poincare-distance } u i1 v i2))$

**definition** *poincare-distance* :: *complex-homo*  $\Rightarrow$  *complex-homo*  $\Rightarrow$  *real* **where**

*poincare-distance*  $u v = (\text{THE } d. \text{poincare-distance-pred } u v d)$

We shown that the described cross-ratio is always finite, positive real number.

**lemma** *distance-cross-ratio-real-positive*:

**assumes**  $u \in \text{unit-disc}$  **and**  $v \in \text{unit-disc}$  **and**  $u \neq v$

**shows**  $\forall i1 i2. \text{ideal-points } (\text{poincare-line } u v) = \{i1, i2\} \longrightarrow$

$\text{cross-ratio } u i1 v i2 \neq \infty_h \wedge \text{is-real } (\text{to-complex } (\text{cross-ratio } u i1 v i2)) \wedge \text{Re-cross-ratio } u i1 v i2 > 0$

(**is** ?*P*  $u v$ )

**proof** (*rule wlog-positive-x-axis*[*OF assms*])

**fix**  $x$

**assume** \*: *is-real*  $x$   $0 < \text{Re } x$   $\text{Re } x < 1$

**hence**  $x \neq -1$   $x \neq 1$

**by** *auto*

**hence** \*\*: *of-complex*  $x \neq \infty_h$  *of-complex*  $x \neq 0_h$  *of-complex*  $x \neq \text{of-complex } (-1)$  *of-complex*  $1 \neq \text{of-complex } x$   
*of-complex*  $x \in \text{circline-set } x\text{-axis}$

**using** \*

**unfolding** *circline-set-x-axis*

**by** (*auto simp add: of-complex-inj*)

**have** \*\*\*:  $0_h \neq \text{of-complex } (-1)$   $0_h \neq \text{of-complex } 1$

**by** (*metis of-complex-zero-iff zero-neq-neg-one, simp*)

**have** \*\*\*\*:  $-x - 1 \neq 0$   $x - 1 \neq 0$

**using**  $\langle x \neq -1 \rangle \langle x \neq 1 \rangle$

**by** (*metis add.inverse-inverse eq-iff-diff-eq-0, simp*)

**have** *poincare-line*  $0_h$  (*of-complex*  $x$ ) = *x-axis*

**using** \*\*

**by** (*simp add: poincare-line-0-real-is-x-axis*)

**thus** ?*P*  $0_h$  (*of-complex*  $x$ )

**using** \* \*\* \*\*\* \*\*\*\*

**using** *cross-ratio-not-inf*[*of*  $0_h$  *of-complex*  $1$  *of-complex*  $(-1)$  *of-complex*  $x$ ]

**using** *cross-ratio-not-inf*[*of*  $0_h$  *of-complex*  $(-1)$  *of-complex*  $1$  *of-complex*  $x$ ]

**using** *cross-ratio-real*[*of*  $0$   $-1$   $x$   $1$ ] *cross-ratio-real*[*of*  $0$   $1$   $x$   $-1$ ]

**apply** (*auto simp add: poincare-line-0-real-is-x-axis doubleton-eq-iff circline-set-x-axis*)

**apply** (*subst cross-ratio, simp-all, subst Re-complex-div-gt-0, simp, subst mult-neg-neg, simp-all*)+

**done**

```

next
  fix M u v
  let ?Mu = moebius-pt M u and ?Mv = moebius-pt M v
  assume *: unit-disc-fix M u ∈ unit-disc v ∈ unit-disc u ≠ v
           ?P ?Mu ?Mv
  show ?P u v
  proof safe
    fix i1 i2
    let ?cr = cross-ratio u i1 v i2
    assume **: ideal-points (poincare-line u v) = {i1, i2}
    have i1 ≠ u i1 ≠ v i2 ≠ u i2 ≠ v i1 ≠ i2
      using ideal-points-different[OF *(2-3), of i1 i2] ** ⟨u ≠ v⟩
    by auto
    hence 0 < Re (to-complex ?cr) ∧ is-real (to-complex ?cr) ∧ ?cr ≠ ∞h
      using * **
    apply (erule-tac x=moebius-pt M i1 in allE)
    apply (erule-tac x=moebius-pt M i2 in allE)
    apply (subst (asm) ideal-points-poincare-line-moebius[of M u v i1 i2], simp-all)
    done
    thus 0 < Re (to-complex ?cr) is-real (to-complex ?cr) ?cr = ∞h ⇒ False
      by simp-all
  qed
qed

```

Next we can show that for every different points from the unit disc there is exactly one number that satisfies the h-distance predicate.

**lemma** *distance-unique*:

```

  assumes u ∈ unit-disc and v ∈ unit-disc
  shows ∃! d. poincare-distance-pred u v d
  proof (cases u = v)
    case True
      thus ?thesis
        by auto
    next
      case False
        obtain i1 i2 where *: i1 ≠ i2 ideal-points (poincare-line u v) = {i1, i2}
          using obtain-ideal-points[OF is-poincare-line-poincare-line] ⟨u ≠ v⟩
          by blast
        let ?d = calc-poincare-distance u i1 v i2
        show ?thesis
        proof (rule ex1I)
          show poincare-distance-pred u v ?d
            using * ⟨u ≠ v⟩
          proof (simp del: calc-poincare-distance-def, safe)
            fix i1' i2'
            assume {i1, i2} = {i1', i2'}
            hence **: (i1' = i1 ∧ i2' = i2) ∨ (i1' = i2 ∧ i2' = i1)
              using doubleton-eq-iff[of i1 i2 i1' i2']
              by blast
            have all-different: u ≠ i1 u ≠ i2 v ≠ i1 v ≠ i2 u ≠ i1' u ≠ i2' v ≠ i1' v ≠ i2' i1 ≠ i2
              using ideal-points-different[OF assms, of i1 i2] * ** ⟨u ≠ v⟩
              by auto
            show calc-poincare-distance u i1 v i2 = calc-poincare-distance u i1' v i2'
            proof –
              let ?cr = cross-ratio u i1 v i2
              let ?cr' = cross-ratio u i1' v i2'
              have Re (to-complex ?cr) > 0 is-real (to-complex ?cr)
                Re (to-complex ?cr') > 0 is-real (to-complex ?cr')
                using False distance-cross-ratio-real-positive[OF assms(1-2)] * **
                by auto
              thus ?thesis
                using **
                using cross-ratio-not-zero cross-ratio-not-inf all-different
            qed
          qed
        qed

```

```

    by auto (subst cross-ratio-commute-24, subst reciprocal-real, simp-all add: ln-div)
  qed
qed
next
  fix d
  assume poincare-distance-pred u v d
  thus d = ?d
    using * ⟨u ≠ v⟩
    by auto
  qed
qed

```

```

lemma poincare-distance-satisfies-pred [simp]:
  assumes u ∈ unit-disc and v ∈ unit-disc
  shows poincare-distance-pred u v (poincare-distance u v)
    using distance-unique[OF assms] theI'[of poincare-distance-pred u v]
  unfolding poincare-distance-def
  by blast

```

```

lemma poincare-distance-I:
  assumes u ∈ unit-disc and v ∈ unit-disc and u ≠ v and ideal-points (poincare-line u v) = {i1, i2}
  shows poincare-distance u v = calc-poincare-distance u i1 v i2
  using assms
  using poincare-distance-satisfies-pred[OF assms(1-2)]
  by simp

```

```

lemma poincare-distance-refl [simp]:
  assumes u ∈ unit-disc
  shows poincare-distance u u = 0
  using assms
  using poincare-distance-satisfies-pred[OF assms assms]
  by simp

```

Unit disc preserving Möbius transformations preserve h-distance.

```

lemma unit-disc-fix-preserve-poincare-distance [simp]:
  assumes unit-disc-fix M and u ∈ unit-disc and v ∈ unit-disc
  shows poincare-distance (moebius-pt M u) (moebius-pt M v) = poincare-distance u v
proof (cases u = v)
  case True
  have moebius-pt M u ∈ unit-disc moebius-pt M v ∈ unit-disc
    using unit-disc-fix-iff[OF assms(1), symmetric] assms
    by blast+
  thus ?thesis
    using assms ⟨u = v⟩
    by simp
  next
  case False
  obtain i1 i2 where *: ideal-points (poincare-line u v) = {i1, i2}
    using ⟨u ≠ v⟩
    by (rule obtain-ideal-points[OF is-poincare-line-poincare-line[of u v]])
  let ?Mu = moebius-pt M u and ?Mv = moebius-pt M v and ?Mi1 = moebius-pt M i1 and ?Mi2 = moebius-pt M i2

  have **: ?Mu ∈ unit-disc ?Mv ∈ unit-disc
    using assms
    using unit-disc-fix-iff
    by blast+

  have ***: ?Mu ≠ ?Mv
    using ⟨u ≠ v⟩
    by simp

  have poincare-distance u v = calc-poincare-distance u i1 v i2
    using poincare-distance-I[OF assms(2-3) ⟨u ≠ v⟩ *]
    by auto
  moreover
  have unit-circle-fix M

```

```

using assms
by simp
hence ++: ideal-points (poincare-line ?Mu ?Mv) = {?Mi1, ?Mi2}
  using ⟨u ≠ v⟩ assms *
  by simp
have poincare-distance ?Mu ?Mv = calc-poincare-distance ?Mu ?Mi1 ?Mv ?Mi2
  by (rule poincare-distance-I[OF ** *** ++])
moreover
have calc-poincare-distance ?Mu ?Mi1 ?Mv ?Mi2 = calc-poincare-distance u i1 v i2
  using ideal-points-different[OF assms(2-3) ⟨u ≠ v⟩ *]
  unfolding calc-poincare-distance-def
  by (subst moebius-preserve-cross-ratio[symmetric], simp-all)
ultimately
show ?thesis
  by simp
qed

```

Knowing ideal points for x-axis, we can easily explicitly calculate distances.

**lemma** *poincare-distance-x-axis-x-axis*:

**assumes**  $x \in \text{unit-disc}$  **and**  $y \in \text{unit-disc}$  **and**  $x \in \text{circline-set } x\text{-axis}$  **and**  $y \in \text{circline-set } x\text{-axis}$

**shows**  $\text{poincare-distance } x \ y =$

(let  $x' = \text{to-complex } x$ ;  $y' = \text{to-complex } y$   
in  $\text{abs} (\ln (\text{Re} (((1 + x') * (1 - y')) / ((1 - x') * (1 + y'))))))$ )

**proof**—

**obtain**  $x' \ y'$  **where**  $*$ :  $x = \text{of-complex } x'$   $y = \text{of-complex } y'$

**using**  $\text{inf-or-of-complex}[of \ x] \ \text{inf-or-of-complex}[of \ y] \ \langle x \in \text{unit-disc} \rangle \ \langle y \in \text{unit-disc} \rangle$

**by** *auto*

**have**  $\text{cmod } x' < 1 \ \text{cmod } y' < 1$

**using**  $\langle x \in \text{unit-disc} \rangle \ \langle y \in \text{unit-disc} \rangle \ *$

**by** (*metis unit-disc-iff-cmod-lt-1*) $+$

**hence**  $**$ :  $x' \neq 1 \ x' \neq -1 \ y' \neq 1 \ y' \neq -1$

**by** *auto*

**have**  $1 + y' \neq 0$

**using**  $**$

**by** (*metis add.left-cancel add-neg-numeral-special*( $\gamma$ ))

**show** *?thesis*

**proof** (*cases*  $x = y$ )

**case** *True*

**thus** *?thesis*

**using**  $\text{assms}(1-2)$

**using**  $\text{unit-disc-iff-cmod-lt-1}[of \ \text{to-complex } x] \ * \ * \ \langle 1 + y' \neq 0 \rangle$

**by** *auto*

**next**

**case** *False*

**hence**  $\text{poincare-line } x \ y = x\text{-axis}$

**using**  $\text{poincare-line-x-axis}[OF \ \text{assms}]$

**by** *simp*

**hence**  $\text{ideal-points } (\text{poincare-line } x \ y) = \{\text{of-complex } (-1), \text{of-complex } 1\}$

**by** *simp*

**hence**  $\text{poincare-distance } x \ y = \text{calc-poincare-distance } x \ (\text{of-complex } (-1)) \ y \ (\text{of-complex } 1)$

**using**  $\text{poincare-distance-I} \ \text{assms} \ \langle x \neq y \rangle$

**by** *auto*

**also have**  $\dots = \text{abs} (\ln (\text{Re} (((x' + 1) * (y' - 1)) / ((x' - 1) * (y' + 1))))))$

**using**  $* \ \langle \text{cmod } x' < 1 \rangle \ \langle \text{cmod } y' < 1 \rangle$

**by** (*simp, transfer, transfer, auto*)

**finally**

**show** *?thesis*

**using**  $*$

**by** (*metis (no-types, lifting) add commute minus-diff-eq minus-divide-divide mult-minus-left mult-minus-right*

*to-complex-of-complex*)

**qed**

**qed**



**lemma** *poincare-distance-zero-x-axis*:

**assumes**  $x \in \text{unit-disc}$  **and**  $x \in \text{circline-set } x\text{-axis}$   
**shows**  $\text{poincare-distance } 0_h x = (\text{let } x' = \text{to-complex } x \text{ in } \text{abs } (\ln (\text{Re } ((1 - x') / (1 + x')))))$   
**using** *assms*  
**using** *poincare-distance-x-axis-x-axis*[of  $0_h x$ ]  
**by** (*simp add: Let-def*)

**lemma** *poincare-distance-zero*:

**assumes**  $x \in \text{unit-disc}$   
**shows**  $\text{poincare-distance } 0_h x = (\text{let } x' = \text{to-complex } x \text{ in } \text{abs } (\ln (\text{Re } ((1 - \text{cmod } x') / (1 + \text{cmod } x')))))$  (**is**  $?P x$ )  
**proof** (*cases*  $x = 0_h$ )  
**case** *True*  
**thus**  $?thesis$   
**by** *auto*  
**next**  
**case** *False*  
**show**  $?thesis$   
**proof** (*rule wlog-rotation-to-positive-x-axis*)  
**show**  $x \in \text{unit-disc } x \neq 0_h$  **by** *fact+*  
**next**  
**fix**  $\varphi u$   
**assume**  $u \in \text{unit-disc } u \neq 0_h$   $?P (\text{moebius-pt } (\text{moebius-rotation } \varphi) u)$   
**thus**  $?P u$   
**using** *unit-disc-fix-preserve-poincare-distance*[of *moebius-rotation*  $\varphi 0_h u$ ]  
**by** (*cases*  $u = \infty_h$ ) (*simp-all add: Let-def*)  
**next**  
**fix**  $x$   
**assume** *is-real*  $x 0 < \text{Re } x \text{Re } x < 1$   
**thus**  $?P (\text{of-complex } x)$   
**using** *poincare-distance-zero-x-axis*[of *of-complex*  $x$ ]  
**by** *simp (auto simp add: circline-set-x-axis cmod-eq-Re complex-is-Real-iff)*  
**qed**  
**qed**

**lemma** *poincare-distance-zero-opposite* [*simp*]:

**assumes** *of-complex*  $z \in \text{unit-disc}$   
**shows**  $\text{poincare-distance } 0_h (\text{of-complex } (-z)) = \text{poincare-distance } 0_h (\text{of-complex } z)$   
**proof**–  
**have**  $*$ : *of-complex*  $(-z) \in \text{unit-disc}$   
**using** *assms*  
**by** *auto*  
**show**  $?thesis$   
**using** *poincare-distance-zero*[*OF assms*]  
**using** *poincare-distance-zero*[*OF \**]  
**by** *simp*  
**qed**

## 5.1 Distance explicit formula

Instead of the h-distance itself, very frequently its hyperbolic cosine is analyzed.

**abbreviation** *cosh-dist*  $u v \equiv \text{cosh } (\text{poincare-distance } u v)$

**lemma** *cosh-poincare-distance-cross-ratio-average*:

**assumes**  $u \in \text{unit-disc } v \in \text{unit-disc } u \neq v$  *ideal-points*  $(\text{poincare-line } u v) = \{i1, i2\}$   
**shows**  $\text{cosh-dist } u v =$   
 $((\text{Re-cross-ratio } u i1 v i2) + (\text{Re-cross-ratio } v i1 u i2)) / 2$

**proof**–

**let**  $?cr = \text{cross-ratio } u i1 v i2$   
**let**  $?crRe = \text{Re } (\text{to-complex } ?cr)$   
**have**  $?cr \neq \infty_h$  *is-real*  $(\text{to-complex } ?cr) ?crRe > 0$   
**using** *distance-cross-ratio-real-positive*[*OF assms(1-3)*] *assms(4)*  
**by** *simp-all*  
**then obtain**  $cr$  **where**  $*$ : *cross-ratio*  $u i1 v i2 = \text{of-complex } cr$   $cr \neq 0$  *is-real*  $cr \text{Re } cr > 0$   
**using** *inf-or-of-complex*[of *cross-ratio*  $u i1 v i2$ ]  
**by** (*smt to-complex-of-complex zero-complex.simps(1)*)

```

thus ?thesis
using *
using assms cross-ratio-commute-13[of v i1 u i2]
unfolding poincare-distance-I[OF assms] calc-poincare-distance-def cosh-def
by (cases Re cr ≥ 1)
  (auto simp add: ln-div[of 0] exp-minus field-simps Re-divide power2-eq-square complex.expand)
qed

```

**definition** *poincare-distance-formula'* :: *complex*  $\Rightarrow$  *complex*  $\Rightarrow$  *real* **where**  
*[simp]:* *poincare-distance-formula' u v* =  $1 + 2 * ((\text{cmod } (u - v))^2 / ((1 - (\text{cmod } u))^2 * (1 - (\text{cmod } v)^2)))$

Next we show that the following formula expresses h-distance between any two h-points (note that the ideal points do not figure anymore).

**definition** *poincare-distance-formula* :: *complex*  $\Rightarrow$  *complex*  $\Rightarrow$  *real* **where**  
*[simp]:* *poincare-distance-formula u v* =  $\text{arcosh } (\text{poincare-distance-formula}' u v)$

**lemma** *blaschke-preserve-distance-formula* *[simp]:*

**assumes** *of-complex k*  $\in$  *unit-disc u*  $\in$  *unit-disc v*  $\in$  *unit-disc*

**shows** *poincare-distance-formula* (*to-complex* (*moebius-pt* (*blaschke k*) *u*)) (*to-complex* (*moebius-pt* (*blaschke k*) *v*)) =  
*poincare-distance-formula* (*to-complex u*) (*to-complex v*)

**proof** (*cases k = 0*)

**case** *True*

**thus** ?thesis

**by** *simp*

**next**

**case** *False*

**obtain** *u' v'* **where**  $*$ :  $u' = \text{to-complex } u$   $v' = \text{to-complex } v$

**by** *auto*

**have**  $\text{cmod } u' < 1$   $\text{cmod } v' < 1$   $\text{cmod } k < 1$

**using** *assms \**

**using** *inf-or-of-complex[of u] inf-or-of-complex[of v]*

**by** *auto*

**obtain** *nu du nv dv d kk ddu ddv* **where**

$**$ :  $nu = u' - k$   $du = 1 - \text{cnj } k * u'$   $nv = v' - k$   $dv = 1 - \text{cnj } k * v'$

$d = u' - v'$   $ddu = 1 - u' * \text{cnj } u'$   $ddv = 1 - v' * \text{cnj } v'$   $kk = 1 - k * \text{cnj } k$

**by** *auto*

**have**  $d: nu * dv - nv * du = d * kk$

**by** (*subst \*\**) + (*simp add: field-simps*)

**have**  $ddu: du * \text{cnj } du - nu * \text{cnj } nu = ddu * kk$

**by** (*subst \*\**) + (*simp add: field-simps*)

**have**  $ddv: dv * \text{cnj } dv - nv * \text{cnj } nv = ddv * kk$

**by** (*subst \*\**) + (*simp add: field-simps*)

**have**  $du \neq 0$

**proof** (*rule ccontr*)

**assume**  $\neg$  ?thesis

**hence**  $\text{cmod } (1 - \text{cnj } k * u') = 0$

**using**  $\langle du = 1 - \text{cnj } k * u' \rangle$

**by** *auto*

**hence**  $\text{cmod } (\text{cnj } k * u') = 1$

**by** *auto*

**thus** *False*

**using**  $\langle \text{cmod } k < 1 \rangle \langle \text{cmod } u' < 1 \rangle$

**using** *mult-strict-mono[of cmod k 1 cmod u' 1]*

**by** (*simp add: norm-mult*)

**qed**

**have**  $dv \neq 0$

**proof** (*rule ccontr*)

**assume**  $\neg$  ?thesis

**hence**  $\text{cmod } (1 - \text{cnj } k * v') = 0$

**using**  $\langle dv = 1 - \text{cnj } k * v' \rangle$

**by** *auto*

**hence**  $cmod (cnj k * v') = 1$   
**by** *auto*  
**thus** *False*  
**using**  $\langle cmod k < 1 \rangle \langle cmod v' < 1 \rangle$   
**using** *mult-strict-mono*[of  $cmod k 1 cmod v' 1$ ]  
**by** (*simp add: norm-mult*)  
**qed**

**have**  $kk \neq 0$   
**proof** (*rule ccontr*)  
**assume**  $\neg ?thesis$   
**hence**  $cmod (1 - k * cnj k) = 0$   
**using**  $\langle kk = 1 - k * cnj k \rangle$   
**by** *auto*  
**hence**  $cmod (k * cnj k) = 1$   
**by** *auto*  
**thus** *False*  
**using**  $\langle cmod k < 1 \rangle$   
**using** *mult-strict-mono*[of  $cmod k 1 cmod k 1$ ]  
**using** *complex-mod-sqrt-Re-mult-cnj* **by** *auto*  
**qed**

**note**  $nz = \langle du \neq 0 \rangle \langle dv \neq 0 \rangle \langle kk \neq 0 \rangle$

**have**  $nu / du - nv / dv = (nu*dv - nv*du) / (du * dv)$   
**using** *nz*  
**by** (*simp add: field-simps*)  
**hence**  $(cmod (nu/du - nv/dv))^2 = cmod ((d*kk) / (du*dv) * (cnj ((d*kk) / (du*dv))))$  (**is** *?lhs = -*)  
**unfolding** *complex-mod-mult-cnj*[*symmetric*]  
**by** (*subst (asm) d*) *simp*  
**also have**  $\dots = cmod ((d*cnj d*kk*kk) / (du*cnj du*dv*cnj dv))$   
**by** (*simp add: field-simps norm-mult norm-divide*)  
**finally have**  $1: ?lhs = cmod ((d*cnj d*kk*kk) / (du*cnj du*dv*cnj dv))$  .

**have**  $(1 - ((cmod nu) / (cmod du))^2) * (1 - ((cmod nv) / (cmod dv))^2) =$   
 $(1 - cmod((nu * cnj nu) / (du * cnj du))) * (1 - cmod((nv * cnj nv) / (dv * cnj dv)))$  (**is** *?rhs = -*)  
**by** (*metis norm-divide complex-mod-mult-cnj power-divide*)  
**also have**  $\dots = cmod(((du*cnj du - nu*cnj nu) / (du * cnj du)) * ((dv*cnj dv - nv*cnj nv) / (dv * cnj dv)))$   
**proof** -

**have**  $u' \neq 1 / cnj k \ v' \neq 1 / cnj k$   
**using**  $\langle cmod u' < 1 \rangle \langle cmod v' < 1 \rangle \langle cmod k < 1 \rangle$   
**by** (*auto simp add: False norm-divide*)

**moreover**

**have**  $cmod k \neq 1$   
**using**  $\langle cmod k < 1 \rangle$   
**by** *linarith*

**ultimately**

**have**  $cmod (nu/du) < 1 \ cmod (nv/dv) < 1$   
**using**  $** (1-4)$

**using** *unit-disc-fix-discI*[*OF blaschke-unit-disc-fix*[*OF*  $\langle cmod k < 1 \rangle$ ]  $\langle u \in unit-disc \rangle$   $\langle u' = to-complex u \rangle$ ]

**using** *unit-disc-fix-discI*[*OF blaschke-unit-disc-fix*[*OF*  $\langle cmod k < 1 \rangle$ ]  $\langle v \in unit-disc \rangle$   $\langle v' = to-complex v \rangle$ ]

**using** *inf-or-of-complex*[of  $u$ ]  $\langle u \in unit-disc \rangle$  *inf-or-of-complex*[of  $v$ ]  $\langle v \in unit-disc \rangle$

**using** *moebius-pt-blaschke*[of  $k u$ ] **using** *moebius-pt-blaschke*[of  $k v$ ]

**by** *auto*

**hence**  $(cmod (nu/du))^2 < 1 \ (cmod (nv/dv))^2 < 1$   
**by** (*simp-all add: cmod-def*)

**hence**  $cmod (nu * cnj nu / (du * cnj du)) < 1 \ cmod (nv * cnj nv / (dv * cnj dv)) < 1$   
**by** (*metis complex-mod-mult-cnj norm-divide power-divide*) +

**moreover**

**have** *is-real*  $(nu * cnj nu / (du * cnj du))$  *is-real*  $(nv * cnj nv / (dv * cnj dv))$

**using** *eq-cnj-iff-real*[of  $nu * cnj nu / (du * cnj du)$ ]

**using** *eq-cnj-iff-real*[of  $nv * cnj nv / (dv * cnj dv)$ ]

**by** (*auto simp add: mult commute*)

**moreover**

**have**  $Re (nu * cnj nu / (du * cnj du)) \geq 0 \ Re (nv * cnj nv / (dv * cnj dv)) \geq 0$

```

    using ⟨du ≠ 0⟩ ⟨dv ≠ 0⟩
    unfolding complex-mult-cnj-cmod
    by simp-all
  ultimately
  have 1 - cmod (nu * cnj nu / (du * cnj du)) = cmod (1 - nu * cnj nu / (du * cnj du))
    1 - cmod (nv * cnj nv / (dv * cnj dv)) = cmod (1 - nv * cnj nv / (dv * cnj dv))
    by (simp-all add: cmod-def)
  thus ?thesis
    using nz
    by (simp add: diff-divide-distrib norm-mult)
qed
also have ... = cmod(((d du * k k) / (du * cnj du)) * ((d dv * k k) / (dv * cnj dv)))
  by (subst ddu, subst ddv, simp)
also have ... = cmod((d du * d dv * k k * k k) / (du * cnj du * dv * cnj dv))
  by (simp add: field-simps)
finally have 2: ?rhs = cmod((d du * d dv * k k * k k) / (du * cnj du * dv * cnj dv))
  .

have ?lhs / ?rhs =
  cmod ((d * cnj d * k k * k k) / (du * cnj du * dv * cnj dv)) / cmod((d du * d dv * k k * k k) / (du * cnj du * dv * cnj dv))
  by (subst 1, subst 2, simp)
also have ... = cmod ((d * cnj d) / (d du * d dv))
  using nz by (simp add: norm-mult norm-divide)
also have ... = (cmod d)2 / ((1 - (cmod u)2) * (1 - (cmod v)2))
proof-
  have (cmod u)2 < 1 (cmod v)2 < 1
    using ⟨cmod u' < 1⟩ ⟨cmod v' < 1⟩
    by (simp-all add: cmod-def)
  hence cmod (1 - u' * cnj u') = 1 - (cmod u)2 cmod (1 - v' * cnj v') = 1 - (cmod v)2
    by (auto simp add: cmod-eq-Re cmod-power2 power2-eq-square[symmetric])
  thus ?thesis
    using nz
    by (simp add: *(6) *(7) norm-divide norm-mult power2-eq-square)
qed
finally
have 3: ?lhs / ?rhs = (cmod d)2 / ((1 - (cmod u)2) * (1 - (cmod v)2)) .

have cmod k ≠ 1 u' ≠ 1 / cnj k v' ≠ 1 / cnj k u ≠ ∞h v ≠ ∞h}
  using ⟨cmod k < 1⟩ ⟨u ∈ unit-disc⟩ ⟨v ∈ unit-disc⟩ * ⟨k ≠ 0⟩ ** ⟨k k ≠ 0⟩ nz
  by auto
thus ?thesis using assms
  using * ** 3
  using moebius-pt-blaschke[of k u]
  using moebius-pt-blaschke[of k v]
  by (simp add: norm-divide)
qed

```

To prove the equivalence between the h-distance definition and the distance formula, we shall employ the without loss of generality principle. Therefore, we must show that the distance formula is preserved by h-isometries.

Rotation preserve *poincare-distance-formula*.

**lemma** *rotation-preserve-distance-formula* [simp]:

**assumes**  $u \in \text{unit-disc}$   $v \in \text{unit-disc}$

**shows** *poincare-distance-formula* (to-complex (moebius-pt (moebius-rotation  $\varphi$ ) u)) (to-complex (moebius-pt (moebius-rotation  $\varphi$ ) v)) =

*poincare-distance-formula* (to-complex u) (to-complex v)

**using** *assms*

**using** *inf-or-of-complex*[of u] *inf-or-of-complex*[of v]

**by** (auto simp: norm-mult)

Unit disc fixing Möbius preserve *poincare-distance-formula*.

**lemma** *unit-disc-fix-preserve-distance-formula* [simp]:

**assumes** *unit-disc-fix*  $M$   $u \in \text{unit-disc}$   $v \in \text{unit-disc}$

**shows** *poincare-distance-formula* (to-complex (moebius-pt  $M$  u)) (to-complex (moebius-pt  $M$  v)) =  
*poincare-distance-formula* (to-complex u) (to-complex v) (is ?P' u v M)

**proof** -

```

have  $\forall u \in \text{unit-disc}. \forall v \in \text{unit-disc}. ?P' u v M$  (is ?P M)
proof (rule wlog-unit-disc-fix[OF assms(1)])
  fix k
  assume  $cmod k < 1$ 
  hence of-complex  $k \in \text{unit-disc}$ 
    by simp
  thus ?P (blaschke k)
    using blaschke-preserve-distance-formula
    by simp
next
fix  $\varphi$ 
show ?P (moebius-rotation  $\varphi$ )
  using rotation-preserve-distance-formula
  by simp
next
fix M1 M2
assume *: ?P M1 and **: ?P M2 and u11: unit-disc-fix M1 unit-disc-fix M2
thus ?P (M1 + M2)
  by (auto simp del: poincare-distance-formula-def)
qed
thus ?thesis
  using assms
  by simp
qed

```

The equivalence between the two h-distance representations.

**lemma** *poincare-distance-formula*:

```

assumes  $u \in \text{unit-disc}$  and  $v \in \text{unit-disc}$ 
shows  $\text{poincare-distance } u v = \text{poincare-distance-formula } (to\text{-complex } u) (to\text{-complex } v)$  (is ?P u v)
proof (rule wlog-x-axis)
  fix x
  assume *: is-real  $x$   $0 \leq Re\ x$   $Re\ x < 1$ 
  show ?P  $0_h$  (of-complex  $x$ ) (is ?lhs = ?rhs)
  proof-
    have of-complex  $x \in \text{unit-disc}$  of-complex  $x \in \text{circline-set } x\text{-axis}$   $cmod\ x < 1$ 
      using *  $cmod\text{-eq-}Re$ 
      by (auto simp add: circline-set-x-axis)
    hence ?lhs =  $|\ln (Re ((1 - x) / (1 + x)))|$ 
      using poincare-distance-zero-x-axis[of of-complex  $x$ ]
      by simp
    moreover
    have ?rhs =  $|\ln (Re ((1 - x) / (1 + x)))|$ 
    proof-
      let ?x =  $1 + 2 * (cmod\ x)^2 / (1 - (cmod\ x)^2)$ 
      have  $0 \leq 2 * (cmod\ x)^2 / (1 - (cmod\ x)^2)$ 
        by (smt <cmod  $x < 1$ > divide-nonneg-nonneg norm-ge-zero power-le-one zero-le-power2)
      hence arcosh-real-gt:  $1 \leq ?x$ 
        by auto
      have ?rhs =  $\text{arcosh } ?x$ 
        by simp
      also have ... =  $\ln ((1 + (cmod\ x)^2) / (1 - (cmod\ x)^2) + 2 * (cmod\ x) / (1 - (cmod\ x)^2))$ 
      proof-
        have  $1 - (cmod\ x)^2 > 0$ 
          using <cmod  $x < 1$ >
          by (smt norm-not-less-zero one-power2 power2-eq-imp-eq power-mono)
        hence 1:  $?x = (1 + (cmod\ x)^2) / (1 - (cmod\ x)^2)$ 
          by (simp add: field-simps)
        have 2:  $?x^2 - 1 = (4 * (cmod\ x)^2) / (1 - (cmod\ x)^2)^2$ 
          using < $1 - (cmod\ x)^2 > 0$ >
          apply (subst 1)
          unfolding power-divide
          by (subst divide-diff-eq-iff, simp, simp add: power2-eq-square field-simps)
        show ?thesis
          using < $1 - (cmod\ x)^2 > 0$ >
          apply (subst arcosh-real-def[OF arcosh-real-gt])
          apply (subst 2)

```

```

    apply (subst 1)
    apply (subst real-sqrt-divide)
    apply (subst real-sqrt-mult)
    apply simp
    done
qed
also have ... = ln (((1 + (cmod x))2) / (1 - (cmod x)2))
  apply (subst add-divide-distrib[symmetric])
  apply (simp add: field-simps power2-eq-square)
  done
also have ... = ln ((1 + cmod x) / (1 - (cmod x)))
  using ⟨cmod x < 1⟩
  using square-diff-square-factored[of 1 cmod x]
  by (simp add: power2-eq-square)
also have ... = |ln (Re ((1 - x) / (1 + x)))|
proof-
  have *: Re ((1 - x) / (1 + x)) ≤ 1 Re ((1 - x) / (1 + x)) > 0
    using ⟨is-real x⟩ ⟨Re x ≥ 0⟩ ⟨Re x < 1⟩
    using complex-is-Real-iff
    by auto
  hence |ln (Re ((1 - x) / (1 + x)))| = -ln (Re ((1 - x) / (1 + x)))
    by auto
  hence |ln (Re ((1 - x) / (1 + x)))| = ln (Re ((1 + x) / (1 - x)))
    using ln-div[of 1 Re ((1 - x)/(1 + x))] * ⟨is-real x⟩
    by (simp add: complex-is-Real-iff)
  moreover
  have ln ((1 + cmod x) / (1 - cmod x)) = ln ((1 + Re x) / (1 - Re x))
    using ⟨Re x ≥ 0⟩ ⟨is-real x⟩
    using cmod-eq-Re by auto
  moreover
  have (1 + Re x) / (1 - Re x) = Re ((1 + x) / (1 - x))
    using ⟨is-real x⟩ ⟨Re x < 1⟩
    by (smt Re-divide-real eq-iff-diff-eq-0 minus-complex.simps one-complex.simps plus-complex.simps)
  ultimately
  show ?thesis
    by simp
qed
finally
show ?thesis
  ·
qed
ultimately
show ?thesis
  by simp
qed
next
fix M u v
assume *: unit-disc-fix M u ∈ unit-disc v ∈ unit-disc
assume ?P (moebius-pt M u) (moebius-pt M v)
thus ?P u v
  using *(1-3)
  by (simp del: poincare-distance-formula-def)
next
show u ∈ unit-disc v ∈ unit-disc
  by fact+
qed

```

Some additional properties proved easily using the distance formula.

*poincare-distance* is symmetric.

**lemma** *poincare-distance-sym*:

```

assumes u ∈ unit-disc and v ∈ unit-disc
shows poincare-distance u v = poincare-distance v u
using assms
using poincare-distance-formula[OF assms(1) assms(2)]
using poincare-distance-formula[OF assms(2) assms(1)]

```

by (*simp add: mult.commute norm-minus-commute*)

**lemma** *poincare-distance-formula'-ge-1*:  
**assumes**  $u \in \text{unit-disc}$  **and**  $v \in \text{unit-disc}$   
**shows**  $1 \leq \text{poincare-distance-formula}'$  (*to-complex*  $u$ ) (*to-complex*  $v$ )  
**using** *unit-disc-cmod-square-lt-1*[*OF assms(1)*] *unit-disc-cmod-square-lt-1*[*OF assms(2)*]  
**by** *auto*

*poincare-distance* is non-negative.

**lemma** *poincare-distance-ge0*:  
**assumes**  $u \in \text{unit-disc}$  **and**  $v \in \text{unit-disc}$   
**shows**  $\text{poincare-distance } u \ v \geq 0$   
**using** *poincare-distance-formula'-ge-1*  
**unfolding** *poincare-distance-formula*[*OF assms(1) assms(2)*]  
**unfolding** *poincare-distance-formula-def*  
**unfolding** *poincare-distance-formula'-def*  
**by** (*rule arcosh-ge-0, simp-all add: assms*)

**lemma** *cosh-dist*:  
**assumes**  $u \in \text{unit-disc}$  **and**  $v \in \text{unit-disc}$   
**shows** *cosh-dist*  $u \ v = \text{poincare-distance-formula}'$  (*to-complex*  $u$ ) (*to-complex*  $v$ )  
**using** *poincare-distance-formula*[*OF assms*] *poincare-distance-formula'-ge-1*[*OF assms*]  
**by** *simp*

*poincare-distance* is zero only if the two points are equal.

**lemma** *poincare-distance-eq-0-iff*:  
**assumes**  $u \in \text{unit-disc}$  **and**  $v \in \text{unit-disc}$   
**shows**  $\text{poincare-distance } u \ v = 0 \iff u = v$   
**using** *assms*  
**apply** *auto*  
**using** *poincare-distance-formula'-ge-1*[*OF assms*]  
**using** *unit-disc-cmod-square-lt-1*[*OF assms(1)*] *unit-disc-cmod-square-lt-1*[*OF assms(2)*]  
**unfolding** *poincare-distance-formula*[*OF assms(1) assms(2)*]  
**unfolding** *poincare-distance-formula-def*  
**unfolding** *poincare-distance-formula'-def*  
**apply** (*subst (asm) arcosh-eq-0-iff*)  
**apply** *assumption*  
**apply** (*simp add: unit-disc-to-complex-inj*)  
**done**

Conjugate preserve *poincare-distance-formula*.

**lemma** *conjugate-preserve-poincare-distance* [*simp*]:  
**assumes**  $u \in \text{unit-disc}$  **and**  $v \in \text{unit-disc}$   
**shows**  $\text{poincare-distance } (\text{conjugate } u) \ (\text{conjugate } v) = \text{poincare-distance } u \ v$   
**proof** –  
**obtain**  $u' \ v'$  **where**  $*$ :  $u = \text{of-complex } u'$   $v = \text{of-complex } v'$   
**using** *assms inf-or-of-complex*[*of u*] *inf-or-of-complex*[*of v*]  
**by** *auto*  
  
**have**  $**$ :  $\text{conjugate } u \in \text{unit-disc}$   $\text{conjugate } v \in \text{unit-disc}$   
**using**  $*$  *assms*  
**by** *auto*  
  
**show** *?thesis*  
**using**  $*$   
**using** *poincare-distance-formula*[*OF assms*]  
**using** *poincare-distance-formula*[*OF \*\**]  
**by** (*metis complex-cnj-diff complex-mod-cnj conjugate-of-complex poincare-distance-def poincare-distance-formula'-def*  
*poincare-distance-formula-def to-complex-of-complex*)  
**qed**

## 5.2 Existence and uniqueness of points with a given distance

**lemma** *ex-x-axis-poincare-distance-negative'*:  
**fixes**  $d :: \text{real}$

```

assumes  $d \geq 0$ 
shows  $let\ z = (1 - exp\ d) / (1 + exp\ d)$ 
  in is-real  $z \wedge Re\ z \leq 0 \wedge Re\ z > -1 \wedge$ 
  of-complex  $z \in unit-disc \wedge of-complex\ z \in circline-set\ x-axis \wedge$ 
  poincare-distance  $0_h (of-complex\ z) = d$ 
proof-
  have  $exp\ d \geq 1$ 
    using assms
    using one-le-exp-iff[of d, symmetric]
    by blast

  hence  $1 + exp\ d \neq 0$ 
    by linarith

  let  $?z = (1 - exp\ d) / (1 + exp\ d)$ 

  have  $?z \leq 0$ 
    using  $\langle exp\ d \geq 1 \rangle$ 
    by (simp add: divide-nonpos-nonneg)

  moreover

  have  $?z > -1$ 
    using exp-gt-zero[of d]
    by (smt divide-less-eq-1-neg nonzero-minus-divide-right)

  moreover

  hence  $abs\ ?z < 1$ 
    using  $\langle ?z \leq 0 \rangle$ 
    by simp
  hence  $cmod\ ?z < 1$ 
    by (metis norm-of-real)
  hence of-complex  $?z \in unit-disc$ 
    by simp

  moreover
  have of-complex  $?z \in circline-set\ x-axis$ 
    unfolding circline-set-x-axis
    by simp

  moreover
  have  $(1 - ?z) / (1 + ?z) = exp\ d$ 
proof-
  have  $1 + ?z = 2 / (1 + exp\ d)$ 
    using  $\langle 1 + exp\ d \neq 0 \rangle$ 
    by (subst add-divide-eq-iff, auto)
  moreover
  have  $1 - ?z = 2 * exp\ d / (1 + exp\ d)$ 
    using  $\langle 1 + exp\ d \neq 0 \rangle$ 
    by (subst diff-divide-eq-iff, auto)
  ultimately
  show ?thesis
    using  $\langle 1 + exp\ d \neq 0 \rangle$ 
    by simp
qed

  ultimately
  show ?thesis
    using poincare-distance-zero-x-axis[of of-complex ?z]
    using  $\langle d \geq 0 \rangle \langle exp\ d \geq 1 \rangle$ 
    by simp (simp add: cmod-eq-Re)
qed

lemma ex-x-axis-poincare-distance-negative:
  assumes  $d \geq 0$ 

```



**shows**  $\exists z. \text{is-real } z \wedge \text{Re } z \leq 0 \wedge \text{Re } z > -1 \wedge$   
 $\text{of-complex } z \in \text{unit-disc} \wedge \text{of-complex } z \in \text{circline-set } x\text{-axis} \wedge$   
 $\text{poincare-distance } 0_h (\text{of-complex } z) = d$  (**is**  $\exists z. ?P z$ )  
**using** *ex-x-axis-poincare-distance-negative*'[*OF assms*]  
**unfolding** *Let-def*  
**by** *blast*

For each real number  $d$  there is exactly one point on the positive x-axis such that h-distance between 0 and that point is  $d$ .

**lemma** *unique-x-axis-poincare-distance-negative*:

**assumes**  $d \geq 0$   
**shows**  $\exists! z. \text{is-real } z \wedge \text{Re } z \leq 0 \wedge \text{Re } z > -1 \wedge$   
 $\text{poincare-distance } 0_h (\text{of-complex } z) = d$  (**is**  $\exists! z. ?P z$ )

**proof**–

**let**  $?z = (1 - \exp d) / (1 + \exp d)$

**have**  $?P ?z$   
**using** *ex-x-axis-poincare-distance-negative*'[*OF assms*]  
**unfolding** *Let-def*  
**by** *blast*

**moreover**

**have**  $\forall z'. ?P z' \longrightarrow z' = ?z$

**proof**–

**let**  $?g = \lambda x'. |\ln (\text{Re } ((1 - x') / (1 + x')))|$   
**let**  $?A = \{x. \text{is-real } x \wedge \text{Re } x > -1 \wedge \text{Re } x \leq 0\}$   
**have** *inj-on* (*poincare-distance*  $0_h \circ \text{of-complex}$ )  $?A$   
**proof** (*rule comp-inj-on*)  
**show** *inj-on of-complex*  $?A$   
**using** *of-complex-inj*  
**unfolding** *inj-on-def*  
**by** *blast*

**next**

**show** *inj-on* (*poincare-distance*  $0_h$ ) (*of-complex* '  $?A$ ) (**is** *inj-on*  $?f$  (*of-complex* '  $?A$ ))  
**proof** (*subst inj-on-cong*)  
**have**  $*$ : *of-complex* '  $?A =$   
 $\{z. z \in \text{unit-disc} \wedge z \in \text{circline-set } x\text{-axis} \wedge \text{Re } (\text{to-complex } z) \leq 0\}$  (**is**  $- = ?B$ )  
**by** (*auto simp add: cmod-eq-Re circline-set-x-axis*)

**fix**  $x$

**assume**  $x \in \text{of-complex ' } ?A$

**hence**  $x \in ?B$

**using**  $*$

**by** *simp*

**thus** *poincare-distance*  $0_h x = (?g \circ \text{to-complex}) x$

**using** *poincare-distance-zero-x-axis*

**by** (*simp add: Let-def*)

**next**

**have**  $*$ : *to-complex* ' *of-complex* '  $?A = ?A$

**by** (*auto simp add: image-iff*)

**show** *inj-on* ( $?g \circ \text{to-complex}$ ) (*of-complex* '  $?A$ )

**proof** (*rule comp-inj-on*)

**show** *inj-on to-complex* (*of-complex* '  $?A$ )

**unfolding** *inj-on-def*

**by** *auto*

**next**

**have** *inj-on*  $?g ?A$

**unfolding** *inj-on-def*

**proof**(*safe*)

**fix**  $x y$

**assume** *hh*: *is-real*  $x$  *is-real*  $y - 1 < \text{Re } x \text{Re } x \leq 0$

$- 1 < \text{Re } y \text{Re } y \leq 0 \ |\ln (\text{Re } ((1 - x) / (1 + x)))| = |\ln (\text{Re } ((1 - y) / (1 + y)))|$

**have** *is-real*  $((1 - x)/(1 + x))$

```

    using ‹is-real x› div-reals[of 1-x 1+x]
    by auto
  have is-real ((1 - y)/(1 + y))
    using ‹is-real y› div-reals[of 1-y 1+y]
    by auto

  have Re (1 + x) > 0
    using ‹- 1 < Re x› by auto
  hence 1 + x ≠ 0
    by force
  have Re (1 - x) ≥ 0
    using ‹Re x ≤ 0› by auto
  hence Re ((1 - x)/(1 + x)) > 0
    using Re-divide-real ‹0 < Re (1 + x)› complex-eq-if-Re-eq hh(1) hh(4) by auto
  have Re(1 - x) ≥ Re (1 + x)
    using hh by auto
  hence Re ((1 - x)/(1 + x)) ≥ 1
    using ‹Re (1 + x) > 0› ‹is-real ((1 - x)/(1 + x))›
  by (smt Re-divide-real arg-0-iff hh(1) le-divide-eq-1-pos one-complex.simps(2) plus-complex.simps(2))

  have Re (1 + y) > 0
    using ‹- 1 < Re y› by auto
  hence 1 + y ≠ 0
    by force
  have Re (1 - y) ≥ 0
    using ‹Re y ≤ 0› by auto
  hence Re ((1 - y)/(1 + y)) > 0
    using Re-divide-real ‹0 < Re (1 + y)› complex-eq-if-Re-eq hh by auto
  have Re(1 - y) ≥ Re (1 + y)
    using hh by auto
  hence Re ((1 - y)/(1 + y)) ≥ 1
    using ‹Re (1 + y) > 0› ‹is-real ((1 - y)/(1 + y))›
  by (smt Re-divide-real arg-0-iff hh le-divide-eq-1-pos one-complex.simps(2) plus-complex.simps(2))

  have ln (Re ((1 - x) / (1 + x))) = ln (Re ((1 - y) / (1 + y)))
    using ‹Re ((1 - y)/(1 + y)) ≥ 1› ‹Re ((1 - x)/(1 + x)) ≥ 1› hh
    by auto
  hence Re ((1 - x) / (1 + x)) = Re ((1 - y) / (1 + y))
    using ‹Re ((1 - y)/(1 + y)) > 0› ‹Re ((1 - x)/(1 + x)) > 0›
    by auto
  hence (1 - x) / (1 + x) = (1 - y) / (1 + y)
    using ‹is-real ((1 - y)/(1 + y))› ‹is-real ((1 - x)/(1 + x))›
    using complex-eq-if-Re-eq by blast
  hence (1 - x) * (1 + y) = (1 - y) * (1 + x)
    using ‹1 + y ≠ 0› ‹1 + x ≠ 0›
    by (simp add:field-simps)
  thus x = y
    by (simp add:field-simps)
qed
thus inj-on ?g (to-complex ‹ of-complex ‹ ?A)
  using *
  by simp
qed
qed
qed
thus ?thesis
  using ‹?P ?z›
  unfolding inj-on-def
  by auto
qed
ultimately
show ?thesis
  by blast
qed

```

lemma ex-x-axis-poincare-distance-positive:

**assumes**  $d \geq 0$   
**shows**  $\exists z. \text{is-real } z \wedge \text{Re } z \geq 0 \wedge \text{Re } z < 1 \wedge$   
 $\text{of-complex } z \in \text{unit-disc} \wedge \text{of-complex } z \in \text{circline-set } x\text{-axis} \wedge$   
 $\text{poincare-distance } 0_h (\text{of-complex } z) = d$  (**is**  $\exists z. \text{is-real } z \wedge \text{Re } z \geq 0 \wedge \text{Re } z < 1 \wedge ?P z$ )  
**proof**–  
**obtain**  $z$  **where**  $*$ :  $\text{is-real } z \wedge \text{Re } z \leq 0 \wedge \text{Re } z > -1$   $?P z$   
**using** *ex-x-axis-poincare-distance-negative*[*OF assms*]  
**by** *auto*  
**hence**  $**$ :  $\text{of-complex } z \in \text{unit-disc} \wedge \text{of-complex } z \in \text{circline-set } x\text{-axis}$   
**by** (*auto simp add: cmod-eq-Re*)  
**have**  $\text{is-real } (-z) \wedge \text{Re } (-z) \geq 0 \wedge \text{Re } (-z) < 1 \wedge ?P (-z)$   
**using**  $**$   
**by** (*simp add: circline-set-x-axis*)  
**thus**  $?thesis$   
**by** *blast*  
**qed**

**lemma** *unique-x-axis-poincare-distance-positive*:

**assumes**  $d \geq 0$   
**shows**  $\exists! z. \text{is-real } z \wedge \text{Re } z \geq 0 \wedge \text{Re } z < 1 \wedge$   
 $\text{poincare-distance } 0_h (\text{of-complex } z) = d$  (**is**  $\exists! z. \text{is-real } z \wedge \text{Re } z \geq 0 \wedge \text{Re } z < 1 \wedge ?P z$ )

**proof**–

**obtain**  $z$  **where**  $*$ :  $\text{is-real } z \wedge \text{Re } z \leq 0 \wedge \text{Re } z > -1$   $?P z$   
**using** *unique-x-axis-poincare-distance-negative*[*OF assms*]  
**by** *auto*  
**hence**  $**$ :  $\text{of-complex } z \in \text{unit-disc} \wedge \text{of-complex } z \in \text{circline-set } x\text{-axis}$   
**by** (*auto simp add: cmod-eq-Re circline-set-x-axis*)  
**show**  $?thesis$

**proof**

**show**  $\text{is-real } (-z) \wedge \text{Re } (-z) \geq 0 \wedge \text{Re } (-z) < 1 \wedge ?P (-z)$   
**using**  $**$   
**by** *simp*

**next**

**fix**  $z'$   
**assume**  $\text{is-real } z' \wedge \text{Re } z' \geq 0 \wedge \text{Re } z' < 1 \wedge ?P z'$   
**hence**  $\text{is-real } (-z') \wedge \text{Re } (-z') \leq 0 \wedge \text{Re } (-z') > -1 \wedge ?P (-z')$   
**by** (*auto simp add: circline-set-x-axis cmod-eq-Re*)  
**hence**  $-z' = z$   
**using** *unique-x-axis-poincare-distance-negative*[*OF assms*]  $*$   
**by** *blast*  
**thus**  $z' = -z$   
**by** *auto*

**qed**

**qed**

Equal distance implies that segments are isometric - this means that congruence could be defined either by two segments having the same distance or by requiring existence of an isometry that maps one segment to the other.

**lemma** *poincare-distance-eq-ex-moebius*:

**assumes** *in-disc*:  $u \in \text{unit-disc}$  **and**  $v \in \text{unit-disc}$  **and**  $u' \in \text{unit-disc}$  **and**  $v' \in \text{unit-disc}$   
**assumes** *poincare-distance*  $u v = \text{poincare-distance } u' v'$   
**shows**  $\exists M. \text{unit-disc-fix } M \wedge \text{moebius-pt } M u = u' \wedge \text{moebius-pt } M v = v'$  (**is**  $?P' u v u' v'$ )

**proof** (*cases*  $u = v$ )

**case** *True*

**thus**  $?thesis$

**using** *assms poincare-distance-eq-0-iff*[*of u' v'*]

**by** (*simp add: unit-disc-fix-transitive*)

**next**

**case** *False*

**have**  $\forall u' v'. u \neq v \wedge u' \in \text{unit-disc} \wedge v' \in \text{unit-disc} \wedge \text{poincare-distance } u v = \text{poincare-distance } u' v' \longrightarrow$   
 $?P' u' v' u v$  (**is**  $?P u v$ )

**proof** (*rule wlog-positive-x-axis*[**where**  $P = ?P$ ])

**fix**  $x$

**assume**  $\text{is-real } x \wedge 0 < \text{Re } x \wedge \text{Re } x < 1$

**hence**  $\text{of-complex } x \in \text{unit-disc} \wedge \text{of-complex } x \in \text{circline-set } x\text{-axis}$

**unfolding** *circline-set-x-axis*

**by** (*auto simp add: cmod-eq-Re*)

**show**  $?P \ 0_h$  (of-complex  $x$ )  
**proof safe**  
**fix**  $u' \ v'$   
**assume**  $0_h \neq$  of-complex  $x$  **and**  $in-disc: u' \in unit-disc \ v' \in unit-disc$  **and**  
 $poincare-distance \ 0_h$  (of-complex  $x$ ) =  $poincare-distance \ u' \ v'$   
**hence**  $u' \neq v'$   $poincare-distance \ u' \ v' > 0$   
**using**  $poincare-distance-eq-0-iff$ [of  $0_h$  of-complex  $x$ ]  $\langle of-complex \ x \in unit-disc \rangle$   
**using**  $poincare-distance-ge0$ [of  $0_h$  of-complex  $x$ ]  
**by auto**  
**then obtain**  $M$  **where**  $M: unit-disc-fix \ M \ moebius-pt \ M \ u' = 0_h \ moebius-pt \ M \ v' \in positive-x-axis$   
**using**  $ex-unit-disc-fix-to-zero-positive-x-axis$ [of  $u' \ v'$ ]  $in-disc$   
**by auto**

**then obtain**  $Mv'$  **where**  $Mv': moebius-pt \ M \ v' = of-complex \ Mv'$   
**using**  $inf-or-of-complex$ [of  $moebius-pt \ M \ v'$ ]  $in-disc \ unit-disc-fix-iff$ [of  $M$ ]  
**by** ( $metis \ image-eqI \ inf-notin-unit-disc$ )

**have**  $moebius-pt \ M \ v' \in unit-disc$   
**using**  $M(1) \ \langle v' \in unit-disc \rangle$   
**by auto**

**have**  $Re \ Mv' > 0$   $is-real \ Mv' \ Re \ Mv' < 1$   
**using**  $M \ Mv'$   $of-complex-inj \ \langle moebius-pt \ M \ v' \in unit-disc \rangle$   
**unfolding**  $positive-x-axis-def \ circline-set-x-axis$   
**using**  $cmod-eq-Re$   
**by auto fastforce**

**have**  $poincare-distance \ 0_h$  ( $moebius-pt \ M \ v'$ ) =  $poincare-distance \ u' \ v'$   
**using**  $M(1)$   
**using**  $in-disc$   
**by** ( $subst \ M(2)$ [ $symmetric$ ],  $simp$ )

**have**  $Mv' = x$   
**using**  $\langle poincare-distance \ 0_h$  ( $moebius-pt \ M \ v'$ ) =  $poincare-distance \ u' \ v' \rangle \ Mv'$   
**using**  $\langle poincare-distance \ 0_h$  (of-complex  $x$ ) =  $poincare-distance \ u' \ v' \rangle$   
**using**  $unique-x-axis-poincare-distance-positive$ [of  $poincare-distance \ u' \ v'$ ]  
 $\langle poincare-distance \ u' \ v' > 0 \rangle$   
**using**  $\langle Re \ Mv' > 0 \rangle \ \langle Re \ Mv' < 1 \rangle \ \langle is-real \ Mv' \rangle$   
**using**  $\langle is-real \ x \rangle \ \langle Re \ x > 0 \rangle \ \langle Re \ x < 1 \rangle$   
**unfolding**  $positive-x-axis-def$   
**by auto**

**thus**  $?P' \ u' \ v' \ 0_h$  (of-complex  $x$ )  
**using**  $M \ Mv'$   
**by auto**

**qed**

**next**  
**show**  $u \in unit-disc \ v \in unit-disc \ u \neq v$   
**by fact+**

**next**  
**fix**  $M \ u \ v$   
**let**  $?Mu = moebius-pt \ M \ u$  **and**  $?Mv = moebius-pt \ M \ v$   
**assume** 1:  $unit-disc-fix \ M \ u \in unit-disc \ v \in unit-disc \ u \neq v$   
**hence** 2:  $?Mu \neq ?Mv \ ?Mu \in unit-disc \ ?Mv \in unit-disc$   
**by auto**  
**assume** 3:  $?P$  ( $moebius-pt \ M \ u$ ) ( $moebius-pt \ M \ v$ )  
**show**  $?P \ u \ v$   
**proof safe**  
**fix**  $u' \ v'$   
**assume** 4:  $u' \in unit-disc \ v' \in unit-disc \ poincare-distance \ u \ v = poincare-distance \ u' \ v'$   
**hence**  $poincare-distance \ ?Mu \ ?Mv = poincare-distance \ u \ v$   
**using** 1  
**by simp**  
**then obtain**  $M'$  **where** 5:  $unit-disc-fix \ M' \ moebius-pt \ M' \ u' = ?Mu \ moebius-pt \ M' \ v' = ?Mv$   
**using** 2 3 4

```

    by auto
  let ?M = (-M) + M'
  have unit-disc-fix ?M ∧ moebius-pt ?M u' = u ∧ moebius-pt ?M v' = v
    using 5 ⟨unit-disc-fix M⟩
    using unit-disc-fix-moebius-comp[of -M M']
    using unit-disc-fix-moebius-inv[of M]
    by simp
  thus ∃ M. unit-disc-fix M ∧ moebius-pt M u' = u ∧ moebius-pt M v' = v
    by blast
qed
qed
then obtain M where unit-disc-fix M ∧ moebius-pt M u' = u ∧ moebius-pt M v' = v
  using assms ⟨u ≠ v⟩
  by blast
hence unit-disc-fix (-M) ∧ moebius-pt (-M) u = u' ∧ moebius-pt (-M) v = v'
  using unit-disc-fix-moebius-inv[of M]
  by auto
thus ?thesis
  by blast
qed

lemma unique-midpoint-x-axis:
  assumes x: is-real x -1 < Re x Re x < 1 and
    y: is-real y -1 < Re y Re y < 1 and
    x ≠ y
  shows ∃! z. -1 < Re z ∧ Re z < 1 ∧ is-real z ∧ poincare-distance (of-complex z) (of-complex x) = poincare-distance
    (of-complex z) (of-complex y) (is ∃! z. ?R z (of-complex x) (of-complex y))
proof-
  let ?x = of-complex x and ?y = of-complex y
  let ?P = λ x y. ∃! z. ?R z x y
  have ∀ x. -1 < Re x ∧ Re x < 1 ∧ is-real x ∧ of-complex x ≠ ?y ⟶ ?P (of-complex x) ?y (is ?Q (of-complex y))
  proof (rule wlog-real-zero)
    show ?y ∈ unit-disc
      using y
      by (simp add: cmod-eq-Re)
  next
    show is-real (to-complex ?y)
      using y
      by simp
  next
    show ?Q 0h
  proof (rule allI, rule impI, (erule conjE)+)
    fix x
    assume x: -1 < Re x Re x < 1 is-real x
    let ?x = of-complex x
    assume ?x ≠ 0h
    hence x ≠ 0
      by auto
    hence Re x ≠ 0
      using x
      using complex-neq-0
      by auto
  have *: ∀ a. -1 < a ∧ a < 1 ⟶
    (poincare-distance (of-complex (cor a)) ?x = poincare-distance (of-complex (cor a)) 0h ⟷
      (Re x) * a * a - 2 * a + Re x = 0)
  proof (rule allI, rule impI)
    fix a :: real
    assume -1 < a ∧ a < 1
    hence of-complex (cor a) ∈ unit-disc
      by auto
    moreover
    have (a - Re x)2 / ((1 - a2) * (1 - (Re x)2)) = a2 / (1 - a2) ⟷
      (Re x) * a * a - 2 * a + Re x = 0 (is ?lhs ⟷ ?rhs)
    proof-
      have 1 - a2 ≠ 0

```

```

    using ⟨-1 < a ∧ a < 1⟩
    by (metis cancel-comm-monoid-add-class.diff-cancel diff-eq-diff-less less-numeral-extra(4) power2-eq-1-iff
right-minus-eq)
  hence ?lhs ↔ (a - Re x)2 / (1 - (Re x)2) = a2
    by (smt divide-cancel-right divide-divide-eq-left mult.commute)
  also have ... ↔ (a - Re x)2 = a2 * (1 - (Re x)2)
  proof -
    have 1 - (Re x)2 ≠ 0
      using x
      by (smt power2-eq-1-iff)
    thus ?thesis
      by (simp add: divide-eq-eq)
  qed
  also have ... ↔ a2 * (Re x)2 - 2*a*Re x + (Re x)2 = 0
    by (simp add: power2-diff field-simps)
  also have ... ↔ Re x * (a2 * Re x - 2 * a + Re x) = 0
    by (simp add: power2-eq-square field-simps)
  also have ... ↔ ?rhs
    using ⟨Re x ≠ 0⟩
    by (simp add: mult.commute mult.left-commute power2-eq-square)
  finally
  show ?thesis
  ·
  qed
  moreover
  have arcosh (1 + 2 * ((a - Re x)2 / ((1 - a2) * (1 - (Re x)2))) = arcosh (1 + 2 * a2 / (1 - a2)) ↔ ?lhs
    using ⟨-1 < a ∧ a < 1⟩ x mult-left-cancel[of 2::real (a - Re x)2 / ((1 - a2) * (1 - (Re x)2)) a2 / (1 - a2)]
    by (subst arcosh-eq-iff, simp-all add: square-le-1)
  ultimately
  show poincare-distance (of-complex (cor a)) (of-complex x) = poincare-distance (of-complex (cor a)) 0_h ↔
    (Re x) * a * a - 2 * a + Re x = 0
    using x
    by (auto simp add: poincare-distance-formula cmod-eq-Re)
  qed

show ?P ?x 0_h
proof
  let ?a = (1 - sqrt(1 - (Re x)2)) / (Re x)
  let ?b = (1 + sqrt(1 - (Re x)2)) / (Re x)

  have is-real ?a
    by simp
  moreover
  have 1 - (Re x)2 > 0
    using x
    by (smt power2-eq-1-iff square-le-1)
  have |?a| < 1
  proof (cases Re x > 0)
    case True
    have (1 - Re x)2 < 1 - (Re x)2
      using ⟨Re x > 0⟩ x
      by (simp add: power2-eq-square field-simps)
    hence 1 - Re x < sqrt(1 - (Re x)2)
      using real-less-rsqrt by fastforce
    thus ?thesis
      using ⟨1 - (Re x)2 > 0⟩ ⟨Re x > 0⟩
      by simp
  next
    case False
    hence Re x < 0
      using ⟨Re x ≠ 0⟩
      by simp
  have 1 + Re x > 0
    using ⟨Re x > -1⟩
    by simp

```

```

hence  $2 * Re\ x + 2 * Re\ x * Re\ x < 0$ 
using  $\langle Re\ x < 0 \rangle$ 
by (metis comm-semiring-class.distrib mult.commute mult-2-right mult-less-0-iff one-add-one zero-less-double-add-iff-zero-less-s)
hence  $(1 + Re\ x)^2 < 1 - (Re\ x)^2$ 
by (simp add: power2-eq-square field-simps)
hence  $1 + Re\ x < sqrt\ (1 - (Re\ x)^2)$ 
using  $\langle 1 - (Re\ x)^2 > 0 \rangle$ 
using real-less-rsqrt by blast
thus ?thesis
using  $\langle Re\ x < 0 \rangle$ 
by (simp add: field-simps)
qed
hence  $-1 < ?a\ ?a < 1$ 
by linarith+
moreover
have  $(Re\ x) * ?a * ?a - 2 * ?a + Re\ x = 0$ 
using  $\langle Re\ x \neq 0 \rangle\ \langle 1 - (Re\ x)^2 > 0 \rangle$ 
by (simp add: field-simps power2-eq-square)
ultimately
show  $-1 < Re\ (cor\ ?a) \wedge Re\ (cor\ ?a) < 1 \wedge is-real\ ?a \wedge poincare-distance\ (of-complex\ ?a)\ (of-complex\ x) =$ 
 $poincare-distance\ (of-complex\ ?a)\ 0_h$ 
using *
by auto

fix z
assume **:  $-1 < Re\ z \wedge Re\ z < 1 \wedge is-real\ z \wedge$ 
 $poincare-distance\ (of-complex\ z)\ (of-complex\ x) = poincare-distance\ (of-complex\ z)\ 0_h$ 
hence  $Re\ x * Re\ z * Re\ z - 2 * Re\ z + Re\ x = 0$ 
using *[rule-format, of Re z] x
by auto
moreover
have  $sqrt\ (4 - 4 * Re\ x * Re\ x) = 2 * sqrt\ (1 - Re\ x * Re\ x)$ 
proof-
have  $sqrt\ (4 - 4 * Re\ x * Re\ x) = sqrt\ (4 * (1 - Re\ x * Re\ x))$ 
by simp
thus ?thesis
by (simp only: real-sqrt-mult, simp)
qed
moreover
have  $(2 - 2 * sqrt\ (1 - Re\ x * Re\ x)) / (2 * Re\ x) = ?a$ 
proof-
have  $(2 - 2 * sqrt\ (1 - Re\ x * Re\ x)) / (2 * Re\ x) =$ 
 $(2 * (1 - sqrt\ (1 - Re\ x * Re\ x))) / (2 * Re\ x)$ 
by simp
thus ?thesis
by (subst (asm) mult-divide-mult-cancel-left) (auto simp add: power2-eq-square)
qed
moreover
have  $(2 + 2 * sqrt\ (1 - Re\ x * Re\ x)) / (2 * Re\ x) = ?b$ 
proof-
have  $(2 + 2 * sqrt\ (1 - Re\ x * Re\ x)) / (2 * Re\ x) =$ 
 $(2 * (1 + sqrt\ (1 - Re\ x * Re\ x))) / (2 * Re\ x)$ 
by simp
thus ?thesis
by (subst (asm) mult-divide-mult-cancel-left) (auto simp add: power2-eq-square)
qed
ultimately
have  $Re\ z = ?a \vee Re\ z = ?b$ 
using discriminant-nonneg[of Re x - 2 Re x Re z] discrim-def[of Re x - 2 Re x]
using  $\langle Re\ x \neq 0 \rangle\ \langle -1 < Re\ x \rangle\ \langle Re\ x < 1 \rangle\ \langle 1 - (Re\ x)^2 > 0 \rangle$ 
by (auto simp add: power2-eq-square)
have  $|?b| > 1$ 
proof (cases Re x > 0)
case True
have  $(Re\ x - 1)^2 < 1 - (Re\ x)^2$ 
using  $\langle Re\ x > 0 \rangle\ x$ 

```

```

    by (simp add: power2-eq-square field-simps)
  hence  $\text{Re } x - 1 < \text{sqrt } (1 - (\text{Re } x)^2)$ 
    using real-less-rsqrt
    by simp
  thus ?thesis
    using  $\langle 1 - (\text{Re } x)^2 > 0 \rangle \langle \text{Re } x > 0 \rangle$ 
    by simp
next
  case False
  hence  $\text{Re } x < 0$ 
    using  $\langle \text{Re } x \neq 0 \rangle$ 
    by simp
  have  $1 + \text{Re } x > 0$ 
    using  $\langle \text{Re } x > -1 \rangle$ 
    by simp
  hence  $2 * \text{Re } x + 2 * \text{Re } x * \text{Re } x < 0$ 
    using  $\langle \text{Re } x < 0 \rangle$ 
  by (metis comm-semiring-class.distrib mult.commute mult-2-right mult-less-0-iff one-add-one zero-less-double-add-iff-zero-less-si)
  hence  $1 - (\text{Re } x)^2 > (-1 - \text{Re } x)^2$ 
    by (simp add: field-simps power2-eq-square)
  hence  $\text{sqrt } (1 - (\text{Re } x)^2) > -1 - \text{Re } x$ 
    using real-less-rsqrt
    by simp
  thus ?thesis
    using  $\langle \text{Re } x < 0 \rangle$ 
    by (simp add: field-simps)
qed
hence  $?b < -1 \vee ?b > 1$ 
  by auto

hence  $\text{Re } z = ?a$ 
  using  $\langle \text{Re } z = ?a \vee \text{Re } z = ?b \rangle **$ 
  by auto
thus  $z = ?a$ 
  using ** complex-of-real-Re
  by fastforce
qed
qed
next
  fix  $a u$ 
  let  $?M = \text{moebius-pt } (\text{blaschke } a)$ 
  let  $?Mu = ?M u$ 
  assume  $u \in \text{unit-disc is-real } a \text{ cmod } a < 1$ 
  assume *:  $?Q ?Mu$ 
  show  $?Q u$ 
  proof (rule allI, rule impI, (erule conjE)+)
    fix  $x$ 
    assume  $x: -1 < \text{Re } x \text{ Re } x < 1 \text{ is-real } x \text{ of-complex } x \neq u$ 
    let  $?Mx = ?M (\text{of-complex } x)$ 
    have  $\text{of-complex } x \in \text{unit-disc}$ 
      using  $x \text{ cmod-eq-Re}$ 
      by auto
    hence  $?Mx \in \text{unit-disc}$ 
      using  $\langle \text{is-real } a \rangle \langle \text{cmod } a < 1 \rangle \text{blaschke-unit-disc-fix}[of a]$ 
      using  $\text{unit-disc-fix-discI}$ 
      by blast
    hence  $?Mx \neq \infty_h$ 
      by auto
    moreover
    have  $\text{of-complex } x \in \text{circline-set } x\text{-axis}$ 
      using  $x$ 
      by auto
    hence  $?Mx \in \text{circline-set } x\text{-axis}$ 
      using  $\text{blaschke-real-preserve-x-axis}[OF \langle \text{is-real } a \rangle \langle \text{cmod } a < 1 \rangle, \text{of of-complex } x]$ 
      by auto
    hence  $-1 < \text{Re } (\text{to-complex } ?Mx) \wedge \text{Re } (\text{to-complex } ?Mx) < 1 \wedge \text{is-real } (\text{to-complex } ?Mx)$ 

```



```

using ⟨?Mx ≠ ∞h⟩ ⟨?Mx ∈ unit-disc⟩
unfolding circline-set-x-axis
by (auto simp add: cmod-eq-Re)
moreover
have ?Mx ≠ ?Mu
  using ⟨of-complex x ≠ u⟩
  by simp
ultimately
have ?P ?Mx ?Mu
  using *[rule-format, of to-complex ?Mx] ⟨?Mx ≠ ∞h⟩
  by simp
then obtain Mz where
  ?R Mz ?Mx ?Mu
  by blast
have of-complex Mz ∈ unit-disc of-complex Mz ∈ circline-set x-axis
  using ⟨?R Mz ?Mx ?Mu⟩
  using cmod-eq-Re
  by auto

let ?Minv = - (blaschke a)
let ?z = moebius-pt ?Minv (of-complex Mz)
have ?z ∈ unit-disc
  using ⟨of-complex Mz ∈ unit-disc⟩ ⟨cmod a < 1⟩
  by auto
moreover
have ?z ∈ circline-set x-axis
  using ⟨of-complex Mz ∈ circline-set x-axis⟩
  using blaschke-real-preserve-x-axis ⟨is-real a⟩ ⟨cmod a < 1⟩
  by fastforce
ultimately
have z1: -1 < Re (to-complex ?z) Re (to-complex ?z) < 1 is-real (to-complex ?z)
  using inf-or-of-complex[of ?z]
  unfolding circline-set-x-axis
  by (auto simp add: cmod-eq-Re)

have z2: poincare-distance ?z (of-complex x) = poincare-distance ?z u
  using ⟨?R Mz ?Mx ?Mu⟩ ⟨cmod a < 1⟩ ⟨?z ∈ unit-disc⟩ ⟨of-complex x ∈ unit-disc⟩ ⟨u ∈ unit-disc⟩
by (metis blaschke-preserve-distance-formula blaschke-unit-disc-fix moebius-pt-comp-inv-right poincare-distance-formula
uminus-moebius-def unit-disc-fix-discI unit-disc-iff-cmod-lt-1)
show ?P (of-complex x) u
proof
  show ?R (to-complex ?z) (of-complex x) u
    using z1 z2 ⟨?z ∈ unit-disc⟩ inf-or-of-complex[of ?z]
    by auto
next
fix z'
assume ?R z' (of-complex x) u
hence of-complex z' ∈ unit-disc of-complex z' ∈ circline-set x-axis
  by (auto simp add: cmod-eq-Re)
let ?Mz' = ?M (of-complex z')
have ?Mz' ∈ unit-disc ?Mz' ∈ circline-set x-axis
  using ⟨of-complex z' ∈ unit-disc⟩ ⟨of-complex z' ∈ circline-set x-axis⟩ ⟨cmod a < 1⟩ ⟨is-real a⟩
  using blaschke-unit-disc-fix unit-disc-fix-discI
  using blaschke-real-preserve-x-axis circline-set-x-axis
  by blast+
hence -1 < Re (to-complex ?Mz') Re (to-complex ?Mz') < 1 is-real (to-complex ?Mz')
  unfolding circline-set-x-axis
  by (auto simp add: cmod-eq-Re)
moreover
have poincare-distance ?Mz' ?Mx = poincare-distance ?Mz' ?Mu
  using ⟨?R z' (of-complex x) u⟩
  using ⟨cmod a < 1⟩ ⟨of-complex x ∈ unit-disc⟩ ⟨of-complex z' ∈ unit-disc⟩ ⟨u ∈ unit-disc⟩
  by auto
ultimately
have ?R (to-complex ?Mz') ?Mx ?Mu
  using ⟨?Mz' ∈ unit-disc⟩ inf-or-of-complex[of ?Mz']

```

```

    by auto
  hence ?Mz' = of-complex Mz
    using ‹?P ?Mx ?Mu› ‹?R Mz ?Mx ?Mu›
  by (metis ‹moebius-pt (blaschke a) (of-complex z') ∈ unit-disc› ‹of-complex Mz ∈ unit-disc› to-complex-of-complex
unit-disc-to-complex-inj)
  thus z' = to-complex ?z
    by (simp add: moebius-pt-invert)
qed
qed
qed
thus ?thesis
  using assms
  by (metis to-complex-of-complex)
qed

```

### 5.3 Triangle inequality

**lemma** *poincare-distance-formula-zero-sum*:

**assumes**  $u \in \text{unit-disc}$  **and**  $v \in \text{unit-disc}$

**shows**  $\text{poincare-distance } u \ 0_h + \text{poincare-distance } 0_h \ v =$

$$\begin{aligned} & (\text{let } u' = \text{cmod } (\text{to-complex } u); v' = \text{cmod } (\text{to-complex } v) \\ & \text{in } \text{arcosh } (((1 + u'^2) * (1 + v'^2) + 4 * u' * v') / ((1 - u'^2) * (1 - v'^2)))) \end{aligned}$$

**proof**–

**obtain**  $u' \ v'$  **where**  $uv: u' = \text{to-complex } u \ v' = \text{to-complex } v$

**by** *auto*

**have**  $uv': u = \text{of-complex } u' \ v = \text{of-complex } v'$

**using**  $uv$  *assms inf-or-of-complex[of u] inf-or-of-complex[of v]*

**by** *auto*

**let**  $?u' = \text{cmod } u' \ \text{and} \ ?v' = \text{cmod } v'$

**have** *disc*:  $?u'^2 < 1 \ ?v'^2 < 1$

**using** *unit-disc-cmod-square-lt-1[OF ‹u ∈ unit-disc›]*

**using** *unit-disc-cmod-square-lt-1[OF ‹v ∈ unit-disc›]*  $uv$

**by** *auto*

**thm** *arcosh-add*

**have**  $\text{arcosh } (1 + 2 * ?u'^2 / (1 - ?u'^2)) + \text{arcosh } (1 + 2 * ?v'^2 / (1 - ?v'^2)) =$

$$\text{arcosh } (((1 + ?u'^2) * (1 + ?v'^2) + 4 * ?u' * ?v') / ((1 - ?u'^2) * (1 - ?v'^2))) \ (\text{is } \text{arcosh } ?ll + \text{arcosh } ?rr = \text{arcosh } ?r)$$

**proof** (*subst arcosh-add*)

**show**  $?ll \geq 1 \ ?rr \geq 1$

**using** *disc*

**by** *auto*

**next**

**show**  $\text{arcosh } ((1 + 2 * ?u'^2 / (1 - ?u'^2)) * (1 + 2 * ?v'^2 / (1 - ?v'^2)) +$

$$\text{sqrt } (((1 + 2 * ?u'^2 / (1 - ?u'^2))^2 - 1) * ((1 + 2 * ?v'^2 / (1 - ?v'^2))^2 - 1))) =$$

$$\text{arcosh } ?r \ (\text{is } \text{arcosh } ?l = -)$$

**proof**–

**have**  $1 + 2 * ?u'^2 / (1 - ?u'^2) = (1 + ?u'^2) / (1 - ?u'^2)$

**using** *disc*

**by** (*subst add-divide-eq-iff, simp-all*)

**moreover**

**have**  $1 + 2 * ?v'^2 / (1 - ?v'^2) = (1 + ?v'^2) / (1 - ?v'^2)$

**using** *disc*

**by** (*subst add-divide-eq-iff, simp-all*)

**moreover**

**have**  $\text{sqrt } (((1 + 2 * ?u'^2 / (1 - ?u'^2))^2 - 1) * ((1 + 2 * ?v'^2 / (1 - ?v'^2))^2 - 1)) =$

$$(4 * ?u' * ?v') / ((1 - ?u'^2) * (1 - ?v'^2)) \ (\text{is } \text{sqrt } ?s = ?t)$$

**proof**–

**have**  $?s = ?t^2$

**using** *disc*

**apply** (*subst add-divide-eq-iff, simp*)+

**apply** (*subst power-divide*)+

**apply** *simp*

**apply** (*subst divide-diff-eq-iff, simp*)+

**apply** (*simp add: power2-eq-square field-simps*)

```

    done
  thus ?thesis
    using disc
    by simp
qed
ultimately
have ?l = ?r
  using disc
  by simp (subst add-divide-distrib, simp)
thus ?thesis
  by simp
qed
qed
thus ?thesis
  using w' assms
  using poincare-distance-formula
  by (simp add: Let-def)
qed

```

lemma poincare-distance-triangle-inequality:

assumes  $u \in \text{unit-disc}$  and  $v \in \text{unit-disc}$  and  $w \in \text{unit-disc}$   
 shows  $\text{poincare-distance } u \ v + \text{poincare-distance } v \ w \geq \text{poincare-distance } u \ w$  (is  $?P' \ u \ v \ w$ )

proof -

have  $\forall w. w \in \text{unit-disc} \longrightarrow ?P' \ u \ v \ w$  (is  $?P \ v \ u$ )

proof (rule wlog-x-axis[where  $P=?P$ ])

fix  $x$

assume  $\text{is-real } x \ 0 \leq \text{Re } x \ \text{Re } x < 1$

hence  $\text{of-complex } x \in \text{unit-disc}$

by (simp add: cmod-eq-Re)

show  $?P \ 0_h \ (\text{of-complex } x)$

proof safe

fix  $w$

assume  $w \in \text{unit-disc}$

then obtain  $w'$  where  $w: w = \text{of-complex } w'$

using  $\text{inf-or-of-complex}[of \ w]$

by auto

let  $?x = \text{cmod } x$  and  $?w = \text{cmod } w'$  and  $?xw = \text{cmod } (x - w')$

have  $\text{disc}: ?x^2 < 1 \ ?w^2 < 1$

using  $\text{unit-disc-cmod-square-lt-1}[OF \ \langle \text{of-complex } x \in \text{unit-disc} \rangle]$

using  $\text{unit-disc-cmod-square-lt-1}[OF \ \langle w \in \text{unit-disc} \rangle] \ w$

by auto

have  $\text{poincare-distance } (\text{of-complex } x) \ 0_h + \text{poincare-distance } 0_h \ w =$

$\text{arcosh } (((1 + ?x^2) * (1 + ?w^2) + 4 * ?x * ?w) / ((1 - ?x^2) * (1 - ?w^2)))$  (is  $- = \text{arcosh } ?r1$ )

using  $\text{poincare-distance-formula-zero-sum}[OF \ \langle \text{of-complex } x \in \text{unit-disc} \rangle \ \langle w \in \text{unit-disc} \rangle] \ w$

by (simp add: Let-def)

moreover

have  $\text{poincare-distance } (\text{of-complex } x) \ (\text{of-complex } w') =$

$\text{arcosh } (((1 - ?x^2) * (1 - ?w^2) + 2 * ?xw^2) / ((1 - ?x^2) * (1 - ?w^2)))$  (is  $- = \text{arcosh } ?r2$ )

using disc

using  $\text{poincare-distance-formula}[OF \ \langle \text{of-complex } x \in \text{unit-disc} \rangle \ \langle w \in \text{unit-disc} \rangle] \ w$

by (subst add-divide-distrib) simp

moreover

have  $*$ :  $(1 - ?x^2) * (1 - ?w^2) + 2 * ?xw^2 \leq (1 + ?x^2) * (1 + ?w^2) + 4 * ?x * ?w$

proof -

have  $(\text{cmod } (x - w'))^2 \leq (\text{cmod } x + \text{cmod } w')^2$

using  $\text{norm-triangle-ineq}_4[of \ x \ w']$

by (simp add: power-mono)

thus ?thesis

by (simp add: field-simps power2-sum)

qed

have  $\text{arcosh } ?r1 \geq \text{arcosh } ?r2$

proof (subst arcosh-mono)

```

    show ?r1 ≥ 1
      using disc
      by (smt * le-divide-eq-1-pos mult-pos-pos zero-le-power2)
  next
    show ?r2 ≥ 1
      using disc
      by simp
  next
    show ?r1 ≥ ?r2
      using disc
      using *
      by (subst divide-right-mono, simp-all)
qed
ultimately
show poincare-distance (of-complex x) w ≤ poincare-distance (of-complex x) 0_h + poincare-distance 0_h w
  using ⟨of-complex x ∈ unit-disc⟩ ⟨w ∈ unit-disc⟩ w
  using poincare-distance-formula
  by simp
qed
next
show v ∈ unit-disc u ∈ unit-disc
  by fact+
next
fix M u v
assume *: unit-disc-fix M u ∈ unit-disc v ∈ unit-disc
assume **: ?P (moebius-pt M u) (moebius-pt M v)
show ?P u v
proof safe
  fix w
  assume w ∈ unit-disc
  thus ?P' v u w
    using * ** [rule-format, of moebius-pt M w]
    by simp
qed
qed
thus ?thesis
  using assms
  by auto
qed

end
theory Poincare-Circles
  imports Poincare-Distance
begin

```

## 6 H-circles in the Poincaré model

Circles consist of points that are at the same distance from the center.

**definition** *poincare-circle* :: *complex-homo* ⇒ *real* ⇒ *complex-homo set* **where**  
*poincare-circle* z r = {z'. z' ∈ unit-disc ∧ poincare-distance z z' = r}

Each h-circle in the Poincaré model is represented by an Euclidean circle in the model — the center and radius of that euclidean circle are determined by the following formulas.

**definition** *poincare-circle-euclidean* :: *complex-homo* ⇒ *real* ⇒ *euclidean-circle* **where**  
*poincare-circle-euclidean* z r =  
 (let R = (cosh r - 1) / 2;  
 z' = to-complex z;  
 cz = 1 - (cmod z')<sup>2</sup>;  
 k = cz \* R + 1  
 in (z' / k, cz \* sqrt(R \* (R + 1)) / k))

That Euclidean circle has a positive radius and is always fully within the disc.

**lemma** *poincare-circle-in-disc*:

**assumes** r > 0 **and** z ∈ unit-disc **and** (ze, re) = *poincare-circle-euclidean* z r

```

shows cmod ze < 1 re > 0  $\forall$  x  $\in$  circle ze re. cmod x < 1
proof-
let ?R = (cosh r - 1) / 2
let ?z' = to-complex z
let ?cz = 1 - (cmod ?z')2
let ?k = ?cz * ?R + 1
let ?ze = ?z' / ?k
let ?re = ?cz * sqrt(?R * (?R + 1)) / ?k

from  $\langle z \in \text{unit-disc} \rangle$ 
obtain z' where z': z = of-complex z'
  using inf-or-of-complex[of z]
  by auto

hence z' = ?z'
  by simp

obtain cz where cz: cz = (1 - (cmod z')2)
  by auto

have cz > 0 cz  $\leq$  1
  using  $\langle z \in \text{unit-disc} \rangle$  z' cz
  using unit-disc-cmod-square-lt-1
  by fastforce+

obtain R where R: R = ?R
  by blast

have R > 0
  using cosh-gt-1[of r]  $\langle r > 0 \rangle$ 
  by (subst R) simp

obtain k where k: k = cz * R + 1
  by auto

have k > 1
  using k  $\langle R > 0 \rangle$   $\langle cz > 0 \rangle$ 
  by simp

hence cmod k = k
  by simp

let ?RR = cz * sqrt(R * (R + 1)) / k

have cmod z' + cz * sqrt(R * (R + 1)) < k
proof-
  have ((R+1)-R)2 > 0
    by simp
  hence (R+1)2 - 2*R*(R+1) + R2 > 0
    unfolding power2-diff
    by (simp add: field-simps)
  hence (R+1)2 + 2*R*(R+1) + R2 - 4*R*(R+1) > 0
    by simp
  hence (2*R+1)2 / 4 > R*(R+1)
    using power2-sum[of R+1 R]
    by (simp add: field-simps)
  hence sqrt(R*(R+1)) < (2*R+1) / 2
    using  $\langle R > 0 \rangle$ 
    by (smt arith-geo-mean-sqrt power-divide real-sqrt-four real-sqrt-pow2 zero-le-mult-iff)
  hence sqrt(R*(R+1)) - R < 1/2
    by (simp add: field-simps)
  hence (1 + (cmod z')) * (sqrt(R*(R+1)) - R) < (1 + (cmod z')) * (1 / 2)
    by (subst mult-strict-left-mono, simp, smt norm-not-less-zero, simp)
also have ... < 1
  using  $\langle z \in \text{unit-disc} \rangle$  z'
  by auto

```

```

finally have  $(1 - \text{cmod } z') * ((1 + \text{cmod } z') * (\text{sqrt}(R*(R+1)) - R)) < (1 - \text{cmod } z') * 1$ 
  using  $\langle z \in \text{unit-disc} \rangle z'$ 
  by  $(\text{subst mult-strict-left-mono, simp-all})$ 
hence  $\text{cz} * (\text{sqrt}(R*(R+1)) - R) < 1 - \text{cmod } z'$ 
  using  $\text{square-diff-square-factored}[of\ 1\ \text{cmod } z']$ 
  by  $(\text{subst cz, subst (asm) mult.assoc[symmetric]}, \text{simp add: power2-eq-square field-simps})$ 
hence  $\text{cmod } z' + \text{cz} * \text{sqrt}(R*(R+1)) < 1 + R * \text{cz}$ 
  by  $(\text{simp add: field-simps})$ 
thus  $?thesis$ 
  using  $k$ 
  by  $(\text{simp add: field-simps})$ 
qed
hence  $\text{cmod } z' / k + \text{cz} * \text{sqrt}(R * (R + 1)) / k < 1$ 
  using  $\langle k > 1 \rangle$ 
  unfolding  $\text{add-divide-distrib}[symmetric]$ 
  by  $\text{simp}$ 
hence  $\text{cmod } (z' / k) + \text{cz} * \text{sqrt}(R * (R + 1)) / k < 1$ 
  using  $\langle k > 1 \rangle$ 
  by  $\text{simp}$ 
hence  $\text{cmod } ?ze + ?re < 1$ 
  using  $k\ \text{cz}\ \langle R = ?R \rangle\ z'$ 
  by  $\text{simp}$ 

moreover

have  $\text{cz} * \text{sqrt}(R * (R + 1)) / k > 0$ 
  using  $\langle \text{cz} > 0 \rangle\ \langle R > 0 \rangle\ \langle k > 1 \rangle$ 
  by  $\text{auto}$ 
hence  $?re > 0$ 
  using  $k\ \text{cz}\ \langle R = ?R \rangle\ z'$ 
  by  $\text{simp}$ 

moreover

have  $\text{cmod } ?ze < 1$ 
  using  $\langle \text{cmod } ?ze + ?re < 1 \rangle\ \langle ?re > 0 \rangle$ 
  by  $\text{simp}$ 

moreover

have  $ze = ?ze\ re = ?re$ 
  using  $\langle (ze, re) = \text{poincare-circle-euclidean } z\ r \rangle$ 
  unfolding  $\text{poincare-circle-euclidean-def Let-def}$ 
  by  $\text{simp-all}$ 

moreover

have  $\forall x \in \text{circle } ze\ re. \text{cmod } x \leq \text{cmod } ze + re$ 
  using  $\text{norm-triangle-ineq2}[of\ -\ ze]$ 
  unfolding  $\text{circle-def}$ 
  by  $(\text{smt mem-Collect-eq})$ 

ultimately

show  $\text{cmod } ze < 1\ re > 0\ \forall x \in \text{circle } ze\ re. \text{cmod } x < 1$ 
  by  $\text{auto}$ 
qed

```

The connection between the points on the h-circle and its corresponding Euclidean circle.

**lemma** *poincare-circle-is-euclidean-circle:*

**assumes**  $z \in \text{unit-disc}$  **and**  $r > 0$

**shows** *let*  $(Ze, Re) = \text{poincare-circle-euclidean } z\ r$

*in of-complex*  $\langle (\text{circle } Ze\ Re) = \text{poincare-circle } z\ r$

**proof**–

{  
  **fix**  $x$

```

let ?z = to-complex z

from assms obtain  $z'$  where  $z' : z = \text{of-complex } z' \text{ cmod } z' < 1$ 
  using inf-or-of-complex[of z]
  by auto

have *:  $\bigwedge x. \text{cmod } x < 1 \implies 1 - (\text{cmod } x)^2 > 0$ 
by (metis less-iff-diff-less-0 minus-diff-eq mult.left-neutral neg-less-0-iff-less norm-mult-less norm-power power2-eq-square)

let ?R = (cosh r - 1) / 2
obtain R where R: R = ?R
  by blast

let ?cx = 1 - (cmod x)2 and ?cz = 1 - (cmod z')2 and ?czx = (cmod (z' - x))2

let ?k = 1 + R * ?cz
obtain k where k: k = ?k
  by blast
have R > 0
  using R cosh-gt-1[OF  $\langle r > 0 \rangle$ ]
  by simp

hence k > 1
  using assms  $z' k * [\text{of } z']$ 
  by auto
hence **: cor k  $\neq 0$ 
  by (smt of-real-eq-0-iff)

have of-complex  $x \in \text{poincare-circle } z r \iff \text{cmod } x < 1 \wedge \text{poincare-distance } z (\text{of-complex } x) = r$ 
  unfolding poincare-circle-def
  by auto
also have ...  $\iff \text{cmod } x < 1 \wedge \text{poincare-distance-formula}' ?z x = \cosh r$ 
  using poincare-distance-formula[of z of-complex x] cosh-dist[of z of-complex x]
  unfolding poincare-distance-formula-def
  using assms
  using arcosh-cosh-real
  by auto
also have ...  $\iff \text{cmod } x < 1 \wedge ?czx / (?cz * ?cx) = ?R$ 
  using  $z'$ 
  by (simp add: field-simps)
also have ...  $\iff \text{cmod } x < 1 \wedge ?czx = ?R * ?cx * ?cz$ 
  using assms  $z' * [\text{of } z'] * [\text{of } x]$ 
  using nonzero-divide-eq-eq[of  $(1 - (\text{cmod } x)^2) * (1 - (\text{cmod } z')^2) (\text{cmod } (z' - x))^2 ?R$ ]
  by (auto, simp add: field-simps)
also have ...  $\iff \text{cmod } x < 1 \wedge (z' - x) * (\text{cnj } z' - \text{cnj } x) = R * ?cz * (1 - x * \text{cnj } x)$  (is -  $\iff$  -  $\wedge$  ?l = ?r)
proof-
  let ?l = (z' - x) * (cnj z' - cnj x) and ?r = R * (1 - Re (z' * cnj z')) * (1 - x * cnj x)
  have is-real ?l
    using eq-cnj-iff-real[of ?l]
    by simp
  moreover
  have is-real ?r
    using eq-cnj-iff-real[of  $1 - x * \text{cnj } x$ ]
    using Im-complex-of-real[of  $R * (1 - \text{Re } (z' * \text{cnj } z'))$ ]
    by simp
  ultimately
  show ?thesis
    apply (subst R[symmetric])
    apply (subst cmod-square) +
    apply (subst complex-eq-if-Re-eq, simp-all add: field-simps)
  done
qed
also have ...  $\iff \text{cmod } x < 1 \wedge z' * \text{cnj } z' - x * \text{cnj } z' - \text{cnj } x * z' + x * \text{cnj } x = R * ?cz - R * ?cz * x * \text{cnj } x$ 
  unfolding right-diff-distrib left-diff-distrib
  by (simp add: field-simps)

```

```

also have ...  $\longleftrightarrow$   $cmod\ x < 1 \wedge k * (x * cnj\ x) - x * cnj\ z' - cnj\ x * z' + z' * cnj\ z' = R * ?cz$  (is -  $\longleftrightarrow$  -  $\wedge$ 
?lhs = ?rhs)
  by (subst k) (auto simp add: field-simps)
also have ...  $\longleftrightarrow$   $cmod\ x < 1 \wedge (k * x * cnj\ x - x * cnj\ z' - cnj\ x * z' + z' * cnj\ z') / k = (R * ?cz) / k$ 
  using **
  by (auto simp add: Groups.mult-ac(1))
also have ...  $\longleftrightarrow$   $cmod\ x < 1 \wedge x * cnj\ x - x * cnj\ z' / k - cnj\ x * z' / k + z' * cnj\ z' / k = (R * ?cz) / k$ 
  using **
  unfolding add-divide-distrib diff-divide-distrib
  by auto
also have ...  $\longleftrightarrow$   $cmod\ x < 1 \wedge (x - z'/k) * cnj(x - z'/k) = (R * ?cz) / k + (z' / k) * cnj(z' / k) - z' * cnj\ z' / k$ 
  by (auto simp add: field-simps diff-divide-distrib)
also have ...  $\longleftrightarrow$   $cmod\ x < 1 \wedge (cmod\ (x - z'/k))^2 = (R * ?cz) / k + (cmod\ z')^2 / k^2 - (cmod\ z')^2 / k$ 
  apply (subst complex-mult-cnj-cmod)+
  apply (subst complex-eq-if-Re-eq)
  apply (simp-all add: power-divide)
  done
also have ...  $\longleftrightarrow$   $cmod\ x < 1 \wedge (cmod\ (x - z'/k))^2 = (R * ?cz * k + (cmod\ z')^2 - (cmod\ z')^2 * k) / k^2$ 
  using **
  unfolding add-divide-distrib diff-divide-distrib
  by (simp add: power2-eq-square)
also have ...  $\longleftrightarrow$   $cmod\ x < 1 \wedge (cmod\ (x - z'/k))^2 = ?cz^2 * R * (R + 1) / k^2$  (is -  $\longleftrightarrow$  -  $\wedge$   $?a^2 = ?b$ )
proof-
  have *:  $R * (1 - (cmod\ z')^2) * k + (cmod\ z')^2 - (cmod\ z')^2 * k = (1 - (cmod\ z')^2)^2 * R * (R + 1)$ 
    by (subst k)+ (simp add: field-simps power2-diff)
  thus ?thesis
    by (subst *, simp)
qed
also have ...  $\longleftrightarrow$   $cmod\ x < 1 \wedge cmod\ (x - z'/k) = ?cz * sqrt\ (R * (R + 1)) / k$ 
  using <R > 0> * [of z'] ** <k > 1> <z ∈ unit-disc> z'
  using real-sqrt-unique [of ?a ?b, symmetric]
  by (auto simp add: real-sqrt-divide real-sqrt-mult power-divide power-mult-distrib)
finally
have of-complex  $x \in$  poincare-circle  $z\ r \longleftrightarrow$   $cmod\ x < 1 \wedge x \in$  circle  $(z'/k)$  ( $?cz * sqrt(R * (R+1)) / k$ )
  unfolding circle-def z' k R
  by simp
hence of-complex  $x \in$  poincare-circle  $z\ r \longleftrightarrow$  (let (Ze, Re) = poincare-circle-euclidean  $z\ r$  in  $cmod\ x < 1 \wedge x \in$ 
circle Ze Re)
  unfolding poincare-circle-euclidean-def Let-def circle-def
  using z' R k
  by (simp add: field-simps)
hence of-complex  $x \in$  poincare-circle  $z\ r \longleftrightarrow$  (let (Ze, Re) = poincare-circle-euclidean  $z\ r$  in  $x \in$  circle Ze Re)
  using poincare-circle-in-disc [OF <r > 0> <z ∈ unit-disc>]
  by auto
} note * = this
show ?thesis
  unfolding Let-def
proof safe
  fix Ze Re x
  assume poincare-circle-euclidean  $z\ r = (Ze, Re)$   $x \in$  circle Ze Re
  thus of-complex  $x \in$  poincare-circle  $z\ r$ 
    using * [of x]
    by simp
next
  fix Ze Re x
  assume **: poincare-circle-euclidean  $z\ r = (Ze, Re)$   $x \in$  poincare-circle  $z\ r$ 
  then obtain  $x'$  where  $x'$ :  $x =$  of-complex  $x'$ 
    unfolding poincare-circle-def
    using inf-or-of-complex [of x]
    by auto
  hence  $x' \in$  circle Ze Re
    using * [of x'] **
    by simp
  thus  $x \in$  of-complex ' circle Ze Re
    using x'

```



by auto  
qed  
qed

## 6.1 Intersection of circles in special positions

Two h-circles centered at the x-axis intersect at mutually conjugate points

**lemma** *intersect-poincare-circles-x-axis*:

**assumes**  $z$ : *is-real*  $z1$  **and** *is-real*  $z2$  **and**  $r1 > 0$  **and**  $r2 > 0$  **and**  
 $-1 < \text{Re } z1$  **and**  $\text{Re } z1 < 1$  **and**  $-1 < \text{Re } z2$  **and**  $\text{Re } z2 < 1$  **and**  
 $z1 \neq z2$

**assumes**  $x1$ :  $x1 \in \text{poincare-circle (of-complex } z1) r1 \cap \text{poincare-circle (of-complex } z2) r2$  **and**  
 $x2$ :  $x2 \in \text{poincare-circle (of-complex } z1) r1 \cap \text{poincare-circle (of-complex } z2) r2$  **and**  
 $x1 \neq x2$

**shows**  $x1 = \text{conjugate } x2$

**proof**–

**have** *in-disc*:  $\text{of-complex } z1 \in \text{unit-disc of-complex } z2 \in \text{unit-disc}$   
**using** *assms*  
**by** (*auto simp add: cmod-eq-Re*)

**obtain**  $x1' x2'$  **where**  $x'$ :  $x1 = \text{of-complex } x1' x2 = \text{of-complex } x2'$   
**using**  $x1 x2$   
**using** *inf-or-of-complex[of x1] inf-or-of-complex[of x2]*  
**unfolding** *poincare-circle-def*  
**by** *auto*

**obtain**  $Ze1 Re1$  **where**  $1$ :  $(Ze1, Re1) = \text{poincare-circle-euclidean (of-complex } z1) r1$   
**by** (*metis poincare-circle-euclidean-def*)

**obtain**  $Ze2 Re2$  **where**  $2$ :  $(Ze2, Re2) = \text{poincare-circle-euclidean (of-complex } z2) r2$   
**by** (*metis poincare-circle-euclidean-def*)

**have** *circle*:  $x1' \in \text{circle } Ze1 Re1 \cap \text{circle } Ze2 Re2$   $x2' \in \text{circle } Ze1 Re1 \cap \text{circle } Ze2 Re2$   
**using** *poincare-circle-is-euclidean-circle[of of-complex z1 r1]*  
**using** *poincare-circle-is-euclidean-circle[of of-complex z2 r2]*  
**using** *assms 1 2 <of-complex z1 ∈ unit-disc> <of-complex z2 ∈ unit-disc> x'*  
**by** *auto (metis image-iff of-complex-inj)+*

**have** *is-real*  $Ze1$  *is-real*  $Ze2$   
**using**  $1 2$  *<is-real z1> <is-real z2>*  
**by** (*simp-all add: poincare-circle-euclidean-def Let-def*)

**have**  $Re1 > 0$   $Re2 > 0$   
**using**  $1 2$  *in-disc <r1 > 0> <r2 > 0>*  
**using** *poincare-circle-in-disc(2)[of r1 of-complex z1 Ze1 Re1]*  
**using** *poincare-circle-in-disc(2)[of r2 of-complex z2 Ze2 Re2]*  
**by** *auto*

**have**  $Ze1 \neq Ze2$

**proof** (*rule ccontr*)

**assume**  $\neg$  *?thesis*

**hence** *eq*:  $Ze1 = Ze2$   $Re1 = Re2$

**using** *circle(1)*

**unfolding** *circle-def*

**by** *auto*

**let**  $?A = Ze1 - Re1$  **and**  $?B = Ze1 + Re1$

**have**  $?A \in \text{circle } Ze1 Re1$   $?B \in \text{circle } Ze1 Re1$

**using** *<Re1 > 0>*

**unfolding** *circle-def*

**by** *simp-all*

**hence** *of-complex ?A ∈ poincare-circle (of-complex z1) r1 of-complex ?B ∈ poincare-circle (of-complex z1) r1*  
*of-complex ?A ∈ poincare-circle (of-complex z2) r2 of-complex ?B ∈ poincare-circle (of-complex z2) r2*

**using** *eq*

**using** *poincare-circle-is-euclidean-circle[OF <of-complex z1 ∈ unit-disc> <r1 > 0>]*

**using** *poincare-circle-is-euclidean-circle[OF <of-complex z2 ∈ unit-disc> <r2 > 0>]*

**using**  $1 2$

**by** *auto blast+*

**hence** *poincare-distance* (of-complex  $z1$ ) (of-complex  $?A$ ) = *poincare-distance* (of-complex  $z1$ ) (of-complex  $?B$ )  
*poincare-distance* (of-complex  $z2$ ) (of-complex  $?A$ ) = *poincare-distance* (of-complex  $z2$ ) (of-complex  $?B$ )  
 $-1 < \text{Re} (Ze1 - Re1) \text{Re} (Ze1 - Re1) < 1$   $-1 < \text{Re} (Ze1 + Re1) \text{Re} (Ze1 + Re1) < 1$   
**using**  $\langle \text{is-real } Ze1 \rangle \langle \text{is-real } Ze2 \rangle$   
**unfolding** *poincare-circle-def*  
**by** (*auto simp add: cmod-eq-Re*)  
**hence**  $z1 = z2$   
**using** *unique-midpoint-x-axis*[of  $Ze1 - Re1$   $Ze1 + Re1$ ]  
**using**  $\langle \text{is-real } Ze1 \rangle \langle \text{is-real } z1 \rangle \langle \text{is-real } z2 \rangle \langle Re1 > 0 \rangle \langle -1 < \text{Re } z1 \rangle \langle \text{Re } z1 < 1 \rangle \langle -1 < \text{Re } z2 \rangle \langle \text{Re } z2 < 1 \rangle$   
**by** *auto*  
**thus** *False*  
**using**  $\langle z1 \neq z2 \rangle$   
**by** *simp*  
**qed**

**hence**  $*$ :  $(\text{Re } x1')^2 + (\text{Im } x1')^2 - 2 * \text{Re } x1' * Ze1 + Ze1 * Ze1 - \text{cor} (\text{Re1} * \text{Re1}) = 0$   
 $(\text{Re } x1')^2 + (\text{Im } x1')^2 - 2 * \text{Re } x1' * Ze2 + Ze2 * Ze2 - \text{cor} (\text{Re2} * \text{Re2}) = 0$   
 $(\text{Re } x2')^2 + (\text{Im } x2')^2 - 2 * \text{Re } x2' * Ze1 + Ze1 * Ze1 - \text{cor} (\text{Re1} * \text{Re1}) = 0$   
 $(\text{Re } x2')^2 + (\text{Im } x2')^2 - 2 * \text{Re } x2' * Ze2 + Ze2 * Ze2 - \text{cor} (\text{Re2} * \text{Re2}) = 0$   
**using** *circle-equation*[of  $Re1$   $Ze1$ ] *circle-equation*[of  $Re2$   $Ze2$ ] *circle*  
**using** *eq-cnj-iff-real*[of  $Ze1$ ]  $\langle \text{is-real } Ze1 \rangle \langle \text{Re1} > 0 \rangle$   
**using** *eq-cnj-iff-real*[of  $Ze2$ ]  $\langle \text{is-real } Ze2 \rangle \langle \text{Re2} > 0 \rangle$   
**using** *complex-add-cnj*[of  $x1'$ ] *complex-add-cnj*[of  $x2'$ ]  
**using** *distrib-left*[of  $Ze1$   $x1'$  *cnj*  $x1'$ ] *distrib-left*[of  $Ze2$   $x1'$  *cnj*  $x1'$ ]  
**using** *distrib-left*[of  $Ze1$   $x2'$  *cnj*  $x2'$ ] *distrib-left*[of  $Ze2$   $x2'$  *cnj*  $x2'$ ]  
**by** (*auto simp add: complex-mult-cnj power2-eq-square field-simps*)

**hence**  $- 2 * \text{Re } x1' * Ze1 + Ze1 * Ze1 - \text{cor} (\text{Re1} * \text{Re1}) = - 2 * \text{Re } x1' * Ze2 + Ze2 * Ze2 - \text{cor} (\text{Re2} * \text{Re2})$   
 $- 2 * \text{Re } x2' * Ze1 + Ze1 * Ze1 - \text{cor} (\text{Re1} * \text{Re1}) = - 2 * \text{Re } x2' * Ze2 + Ze2 * Ze2 - \text{cor} (\text{Re2} * \text{Re2})$   
**by** (*smt add-diff-cancel-right' add-diff-eq eq-iff-diff-eq-0 minus-diff-eq mult-minus-left of-real-minus*)  
**hence**  $2 * \text{Re } x1' * (Ze2 - Ze1) = (Ze2 * Ze2 - \text{cor} (\text{Re2} * \text{Re2})) - (Ze1 * Ze1 - \text{cor} (\text{Re1} * \text{Re1}))$   
 $2 * \text{Re } x2' * (Ze2 - Ze1) = (Ze2 * Ze2 - \text{cor} (\text{Re2} * \text{Re2})) - (Ze1 * Ze1 - \text{cor} (\text{Re1} * \text{Re1}))$   
**by** *simp-all (simp add: field-simps)*  
**hence**  $2 * \text{Re } x1' * (Ze2 - Ze1) = 2 * \text{Re } x2' * (Ze2 - Ze1)$   
**by** *simp*  
**hence**  $\text{Re } x1' = \text{Re } x2'$   
**using**  $\langle Ze1 \neq Ze2 \rangle$   
**by** *simp*  
**moreover**  
**hence**  $(\text{Im } x1')^2 = (\text{Im } x2')^2$   
**using**  $\langle 1 \rangle \langle 3 \rangle$   
**by** (*simp add: is-real Ze1 complex-eq-if-Re-eq*)  
**hence**  $\text{Im } x1' = \text{Im } x2' \vee \text{Im } x1' = -\text{Im } x2'$   
**using** *power2-eq-iff*  
**by** *blast*  
**ultimately**  
**show** *?thesis*  
**using**  $\langle x' \langle x1 \neq x2 \rangle$   
**using** *complex.expand*  
**by** (*metis cnj.code complex-surj conjugate-of-complex*)  
**qed**

Two h-circles of the same radius centered at mutually conjugate points intersect at the x-axis

**lemma** *intersect-poincare-circles-conjugate-centers*:

**assumes** *in-disc*:  $z1 \in \text{unit-disc}$   $z2 \in \text{unit-disc}$  **and**  
 $z1 \neq z2$  **and**  $z1 = \text{conjugate } z2$  **and**  $r > 0$  **and**  
 $u \in \text{poincare-circle } z1 \ r \cap \text{poincare-circle } z2 \ r$   
**shows** *is-real* (to-complex  $u$ )

**proof**–

**obtain**  $z1e$   $r1e$   $z2e$   $r2e$  **where**  
*euclidean*:  $(z1e, r1e) = \text{poincare-circle-euclidean } z1 \ r$   
 $(z2e, r2e) = \text{poincare-circle-euclidean } z2 \ r$   
**by** (*metis poincare-circle-euclidean-def*)  
**obtain**  $z1'$   $z2'$  **where**  $z'$ :  $z1 = \text{of-complex } z1'$   $z2 = \text{of-complex } z2'$   
**using** *inf-or-of-complex*[of  $z1$ ] *inf-or-of-complex*[of  $z2$ ] *in-disc*  
**by** *auto*

```

obtain  $u'$  where  $u'$ :  $u = \text{of-complex } u'$ 
  using  $u \text{ inf-or-of-complex}[of\ u]$ 
  by (auto simp add: poincare-circle-def)
have  $z1' = \text{cnj } z2'$ 
  using  $\langle z1 = \text{conjugate } z2 \rangle z'$ 
  by (auto simp add: of-complex-inj)
moreover
let  $?cz = 1 - (\text{cmod } z2)^2$ 
let  $?den = ?cz * (\text{cosh } r - 1) / 2 + 1$ 
have  $?cz > 0$ 
  using in-disc  $z'$ 
  by (simp add: cmod-def)
hence  $?den \geq 1$ 
  using cosh-gt-1 [OF  $\langle r > 0 \rangle$ ]
  by auto
hence  $?den \neq 0$ 
  by simp
hence cor  $?den \neq 0$ 
  using of-real-eq-0-iff
  by blast
ultimately
have  $r1e = r2e\ z1e = \text{cnj } z2e\ z1e \neq z2e$ 
  using  $z' \text{ euclidean } \langle z1 \neq z2 \rangle$ 
  unfolding poincare-circle-euclidean-def Let-def
  by simp-all metis

hence  $u' \in \text{circle } (\text{cnj } z2e)\ r2e \cap \text{circle } z2e\ r2e\ z2e \neq \text{cnj } z2e$ 
  using euclidean  $u\ u'$ 
  using poincare-circle-is-euclidean-circle[of  $z1\ r$ ]
  using poincare-circle-is-euclidean-circle[of  $z2\ r$ ]
  using in-disc  $\langle r > 0 \rangle$ 
  by auto (metis image-iff of-complex-inj)+
hence  $(\text{cmod } (u' - z2e))^2 = (\text{cmod}(u' - \text{cnj } z2e))^2$ 
  by (simp add: circle-def)
hence  $(u' - z2e) * (\text{cnj } u' - \text{cnj } z2e) = (u' - \text{cnj } z2e) * (\text{cnj } u' - z2e)$ 
  by (metis complex-cnj-cnj complex-cnj-diff complex-norm-square)
hence  $(z2e - \text{cnj } z2e) * (u' - \text{cnj } u') = 0$ 
  by (simp add: field-simps)
thus ?thesis
  using  $u' \langle z2e \neq \text{cnj } z2e \rangle \text{ eq-cnj-iff-real}[of\ u']$ 
  by simp
qed

```

## 6.2 Congruent triangles

For every pair of triangles such that its three pairs of sides are pairwise equal there is an h-isometry (a unit disc preserving Möbius transform, eventually composed with a conjugation) that maps one triangle onto the other.

**lemma** *unit-disc-fix-f-congruent-triangles*:

**assumes**

*in-disc*:  $u \in \text{unit-disc } v \in \text{unit-disc } w \in \text{unit-disc}$  **and**  
*in-disc'*:  $u' \in \text{unit-disc } v' \in \text{unit-disc } w' \in \text{unit-disc}$  **and**  
*d*: *poincare-distance*  $u\ v = \text{poincare-distance } u'\ v'$   
*poincare-distance*  $v\ w = \text{poincare-distance } v'\ w'$   
*poincare-distance*  $u\ w = \text{poincare-distance } u'\ w'$

**shows**

$\exists M. \text{unit-disc-fix-f } M \wedge M\ u = u' \wedge M\ v = v' \wedge M\ w = w'$

**proof** (*cases*  $u = v \vee u = w \vee v = w$ )

**case** *True*

**thus** *?thesis*

**using** *assms*

**using** *poincare-distance-eq-0-iff*[*of*  $u'\ v'$ ]

**using** *poincare-distance-eq-0-iff*[*of*  $v'\ w'$ ]

**using** *poincare-distance-eq-0-iff*[*of*  $u'\ w'$ ]

**using** *poincare-distance-eq-ex-moebius*[*of*  $v\ w\ v'\ w'$ ]

**using** *poincare-distance-eq-ex-moebius*[*of*  $u\ w\ u'\ w'$ ]

```

using poincare-distance-eq-ex-moebius[of u v u' v']
by (metis unit-disc-fix-f-def)
next
case False

have  $\forall w u' v' w'. w \in \text{unit-disc} \wedge u' \in \text{unit-disc} \wedge v' \in \text{unit-disc} \wedge w' \in \text{unit-disc} \wedge w \neq u \wedge w \neq v \wedge$ 
  poincare-distance u v = poincare-distance u' v'  $\wedge$ 
  poincare-distance v w = poincare-distance v' w'  $\wedge$ 
  poincare-distance u w = poincare-distance u' w'  $\longrightarrow$ 
  ( $\exists M. \text{unit-disc-fix-f } M \wedge M u = u' \wedge M v = v' \wedge M w = w'$ ) (is  $?P u v$ )
proof (rule wlog-positive-x-axis[where  $P = ?P$ ])
  show  $v \in \text{unit-disc} u \in \text{unit-disc}$ 
    by fact+
next
  show  $u \neq v$ 
    using False
    by simp
next
  fix  $x$ 
  assume  $x: \text{is-real } x \ 0 < \text{Re } x \ \text{Re } x < 1$ 

  hence of-complex  $x \neq 0_h$ 
    using of-complex-zero-iff[of x]
    by (auto simp add: complex.expand)

  show  $?P \ 0_h$  (of-complex  $x$ )
  proof safe
    fix  $w u' v' w'$ 
    assume in-disc:  $w \in \text{unit-disc} u' \in \text{unit-disc} v' \in \text{unit-disc} w' \in \text{unit-disc}$ 
    assume poincare-distance  $0_h$  (of-complex  $x$ ) = poincare-distance  $u' v'$ 
    then obtain  $M'$  where  $M': \text{unit-disc-fix } M' \ \text{moebius-pt } M' \ u' = 0_h \ \text{moebius-pt } M' \ v' = (\text{of-complex } x)$ 
      using poincare-distance-eq-ex-moebius[of u' v' 0_h of-complex x in-disc  $x$ ]
      by (auto simp add: cmod-eq-Re)

    let  $?w = \text{moebius-pt } M' \ w'$ 
    have  $?w \in \text{unit-disc}$ 
      using  $\langle \text{unit-disc-fix } M' \rangle \ \langle w' \in \text{unit-disc} \rangle$ 
      by simp

    assume  $w \neq 0_h \ w \neq \text{of-complex } x$ 
    hence dist-gt-0: poincare-distance  $0_h \ w > 0$  poincare-distance (of-complex  $x$ )  $w > 0$ 
      using poincare-distance-eq-0-iff[of 0_h w in-disc poincare-distance-ge0[of 0_h w]]
      using poincare-distance-eq-0-iff[of of-complex x w in-disc poincare-distance-ge0[of of-complex x w]]
      using  $x$ 
      by (simp-all add: cmod-eq-Re)

    assume poincare-distance (of-complex  $x$ )  $w = \text{poincare-distance } v' w'$ 
      poincare-distance  $0_h \ w = \text{poincare-distance } u' w'$ 
    hence poincare-distance  $0_h \ ?w = \text{poincare-distance } 0_h \ w$ 
      poincare-distance (of-complex  $x$ )  $?w = \text{poincare-distance$  (of-complex  $x$ )  $w$ 
      using  $M'(1) \ M'(2)$ [symmetric]  $M'(3)$ [symmetric] in-disc
      using unit-disc-fix-preserve-poincare-distance[of M' u' w']
      using unit-disc-fix-preserve-poincare-distance[of M' v' w']
      by simp-all
      hence  $?w \in \text{poincare-circle } 0_h$  (poincare-distance  $0_h \ w$ )  $\cap$  poincare-circle (of-complex  $x$ ) (poincare-distance
      (of-complex  $x$ )  $w$ )
       $w \in \text{poincare-circle } 0_h$  (poincare-distance  $0_h \ w$ )  $\cap$  poincare-circle (of-complex  $x$ ) (poincare-distance (of-complex
       $x$ )  $w$ )
      using  $\langle ?w \in \text{unit-disc} \rangle \ \langle w \in \text{unit-disc} \rangle$ 
      unfolding poincare-circle-def
      by simp-all
    hence  $?w = w \vee ?w = \text{conjugate } w$ 
      using intersect-poincare-circles-x-axis[of 0 x poincare-distance 0_h w poincare-distance (of-complex  $x$ )  $w \ ?w \ w$ ]  $x$ 
      using  $\langle \text{of-complex } x \neq 0_h \rangle \ \text{dist-gt-0}$ 
      using poincare-distance-eq-0-iff
      by auto

```

```

thus  $\exists M. \text{unit-disc-fix-f } M \wedge M \ 0_h = u' \wedge M \text{ (of-complex } x) = v' \wedge M \ w = w'$ 
proof
  assume  $\text{moebius-pt } M' \ w' = w$ 
  thus ?thesis
    using  $M'$ 
    using  $\text{moebius-pt-invert[of } M' \ u' \ 0_h]$ 
    using  $\text{moebius-pt-invert[of } M' \ v' \ \text{of-complex } x]$ 
    using  $\text{moebius-pt-invert[of } M' \ w' \ w]$ 
    apply  $(\text{rule-tac } x = \text{moebius-pt } (-M') \ \text{in } \text{exI})$ 
    apply  $(\text{simp add: unit-disc-fix-f-def})$ 
    apply  $(\text{rule-tac } x = -M' \ \text{in } \text{exI, simp})$ 
    done
  next
    let  $?M = \text{moebius-pt } (-M') \circ \text{conjugate}$ 
    assume  $\text{moebius-pt } M' \ w' = \text{conjugate } w$ 
    hence  $?M \ w = w'$ 
    using  $\text{moebius-pt-invert[of } M' \ w' \ \text{conjugate } w]$ 
    by  $\text{simp}$ 
    moreover
    have  $?M \ 0_h = u' \ ?M \ \text{(of-complex } x) = v'$ 
    using  $\text{moebius-pt-invert[of } M' \ u' \ 0_h]$ 
    using  $\text{moebius-pt-invert[of } M' \ v' \ \text{of-complex } x]$ 
    using  $M' \ \langle \text{is-real } x \rangle \ \text{eq-cnjl-iff-real[of } x]$ 
    by  $\text{simp-all}$ 
    moreover
    have  $\text{unit-disc-fix-f } ?M$ 
    using  $\langle \text{unit-disc-fix } M' \rangle$ 
    unfolding  $\text{unit-disc-fix-f-def}$ 
    by  $(\text{rule-tac } x = -M' \ \text{in } \text{exI, simp})$ 
    ultimately
    show ?thesis
    by  $\text{blast}$ 
  qed
qed
next
fix  $M \ u \ v$ 
assume  $1: \text{unit-disc-fix } M \ u \in \text{unit-disc } v \in \text{unit-disc}$ 
let  $?Mu = \text{moebius-pt } M \ u$  and  $?Mv = \text{moebius-pt } M \ v$ 
assume  $2: ?P \ ?Mu \ ?Mv$ 
show  $?P \ u \ v$ 
proof safe
  fix  $w \ u' \ v' \ w'$ 
  let  $?Mw = \text{moebius-pt } M \ w$  and  $?Mu' = \text{moebius-pt } M \ u'$  and  $?Mv' = \text{moebius-pt } M \ v'$  and  $?Mw' = \text{moebius-pt}$ 
   $M \ w'$ 
  assume  $w \in \text{unit-disc } u' \in \text{unit-disc } v' \in \text{unit-disc } w' \in \text{unit-disc } w \neq u \ w \neq v$ 
     $\text{poincare-distance } u \ v = \text{poincare-distance } u' \ v'$ 
     $\text{poincare-distance } v \ w = \text{poincare-distance } v' \ w'$ 
     $\text{poincare-distance } u \ w = \text{poincare-distance } u' \ w'$ 
  then obtain  $M'$  where  $M': \text{unit-disc-fix-f } M' \ M' \ ?Mu = ?Mu' \ M' \ ?Mv = ?Mv' \ M' \ ?Mw = ?Mw'$ 
    using  $1 \ 2[\text{rule-format, of } ?Mw \ ?Mu' \ ?Mv' \ ?Mw']$ 
    by  $\text{auto}$ 

  let  $?M = \text{moebius-pt } (-M) \circ M' \circ \text{moebius-pt } M$ 
  show  $\exists M. \text{unit-disc-fix-f } M \wedge M \ u = u' \wedge M \ v = v' \wedge M \ w = w'$ 
proof  $(\text{rule-tac } x = ?M \ \text{in } \text{exI, safe})$ 
  show  $\text{unit-disc-fix-f } ?M$ 
    using  $M'(1) \ \langle \text{unit-disc-fix } M \rangle$ 
    by  $(\text{subst unit-disc-fix-f-comp, subst unit-disc-fix-f-comp, simp-all})$ 
  next
    show  $?M \ u = u' \ ?M \ v = v' \ ?M \ w = w'$ 
    using  $M'$ 
    by  $\text{auto}$ 
  qed
qed
qed
thus ?thesis

```

```

    using assms False
  by auto
qed

end
theory Poincare-Between
  imports Poincare-Distance
begin

```

## 7 H-betweenness in the Poincaré model

### 7.1 H-betweenness expressed by a cross-ratio

The point  $v$  is h-between  $u$  and  $w$  if the cross-ratio between the pairs  $u$  and  $w$  and  $v$  and inverse of  $v$  is real and negative.

```

definition poincare-between :: complex-homo  $\Rightarrow$  complex-homo  $\Rightarrow$  complex-homo  $\Rightarrow$  bool where
  poincare-between  $u\ v\ w \longleftrightarrow$ 
     $u = v \vee v = w \vee$ 
    (let  $cr = \text{cross-ratio } u\ v\ w\ (\text{inversion } v)$ 
     in  $is\text{-real } (to\text{-complex } cr) \wedge Re\ (to\text{-complex } cr) < 0$ )

```

#### 7.1.1 H-betweenness is preserved by h-isometries

Since they preserve cross-ratio and inversion, h-isometries (unit disc preserving Möbius transformations and conjugation) preserve h-betweenness.

```

lemma unit-disc-fix-moebius-preserve-poincare-between [simp]:
  assumes unit-disc-fix  $M$  and  $u \in \text{unit-disc}$  and  $v \in \text{unit-disc}$  and  $w \in \text{unit-disc}$ 
  shows poincare-between (moebius-pt  $M\ u$ ) (moebius-pt  $M\ v$ ) (moebius-pt  $M\ w$ )  $\longleftrightarrow$ 
    poincare-between  $u\ v\ w$ 
proof (cases  $u = v \vee v = w$ )
  case True
  thus ?thesis
    using assms
    unfolding poincare-between-def
    by auto
next
  case False
  moreover
  hence moebius-pt  $M\ u \neq$  moebius-pt  $M\ v \wedge$  moebius-pt  $M\ v \neq$  moebius-pt  $M\ w$ 
    by auto
  moreover
  have  $v \neq \text{inversion } v\ w \neq \text{inversion } v$ 
    using inversion-noteq-unit-disc[of  $v\ w$ ]
    using inversion-noteq-unit-disc[of  $v\ v$ ]
    using  $\langle v \in \text{unit-disc} \rangle \langle w \in \text{unit-disc} \rangle$ 
    by auto
  ultimately
  show ?thesis
    using assms
    using unit-circle-fix-moebius-pt-inversion[of  $M\ v$ , symmetric]
    unfolding poincare-between-def
    by (simp del: unit-circle-fix-moebius-pt-inversion)
qed

```

```

lemma conjugate-preserve-poincare-between [simp]:
  assumes  $u \in \text{unit-disc}$  and  $v \in \text{unit-disc}$  and  $w \in \text{unit-disc}$ 
  shows poincare-between (conjugate  $u$ ) (conjugate  $v$ ) (conjugate  $w$ )  $\longleftrightarrow$ 
    poincare-between  $u\ v\ w$ 
proof (cases  $u = v \vee v = w$ )
  case True
  thus ?thesis
    using assms
    unfolding poincare-between-def
    by auto

```

```

next
case False
moreover
hence conjugate u ≠ conjugate v ∧ conjugate v ≠ conjugate w
using conjugate-inj by blast
moreover
have v ≠ inversion v w ≠ inversion v
using inversion-noteq-unit-disc[of v w]
using inversion-noteq-unit-disc[of v v]
using ⟨v ∈ unit-disc⟩ ⟨w ∈ unit-disc⟩
by auto
ultimately
show ?thesis
using assms
using conjugate-cross-ratio[of v w inversion v u]
unfolding poincare-between-def
by (metis conjugate-id-iff conjugate-involution inversion-def inversion-sym o-apply)
qed

```

### 7.1.2 Some elementary properties of h-betweenness

```

lemma poincare-between-nonstrict [simp]:
shows poincare-between u u v and poincare-between u v v
by (simp-all add: poincare-between-def)

```

```

lemma poincare-between-sandwich:
assumes u ∈ unit-disc and v ∈ unit-disc
assumes poincare-between u v u
shows u = v

```

```

proof (rule ccontr)
assume ¬ ?thesis
thus False
using assms
using inversion-noteq-unit-disc[of v u]
using cross-ratio-1[of v u inversion v]
unfolding poincare-between-def Let-def
by auto

```

qed

```

lemma poincare-between-rev:
assumes u ∈ unit-disc and v ∈ unit-disc and w ∈ unit-disc
shows poincare-between u v w ⟷ poincare-between w v u
using assms
using inversion-noteq-unit-disc[of v w]
using inversion-noteq-unit-disc[of v u]
using cross-ratio-commute-13[of u v w inversion v]
using cross-ratio-not-inf[of w inversion v v u]
using cross-ratio-not-zero[of w v u inversion v]
using inf-or-of-complex[of cross-ratio w v u (inversion v)]
unfolding poincare-between-def
by (auto simp add: Let-def Im-complex-div-eq-0 Re-divide divide-less-0-iff)

```

### 7.1.3 H-betweenness and h-collinearity

Three points can be in an h-between relation only when they are h-collinear.

```

lemma poincare-between-poincare-collinear [simp]:
assumes in-disc: u ∈ unit-disc v ∈ unit-disc w ∈ unit-disc
assumes betw: poincare-between u v w
shows poincare-collinear {u, v, w}

```

```

proof (cases u = v ∨ v = w)

```

```

case True
thus ?thesis
using assms
by auto

```

```

next
case False

```

```

hence distinct: distinct [u, v, w, inversion v]
  using in-disc inversion-noteq-unit-disc[of v v] inversion-noteq-unit-disc[of v u] inversion-noteq-unit-disc[of v w]
  using betw poincare-between-sandwich[of w v]
  by (auto simp add: poincare-between-def Let-def)

then obtain H where *: {u, v, w, inversion v}  $\subseteq$  circline-set H
  using assms
  unfolding poincare-between-def
  using four-points-on-circline-iff-cross-ratio-real[of u v w inversion v]
  by auto
hence H = poincare-line u v
  using assms distinct
  using unique-circline-set[of u v inversion v]
  using poincare-line[of u v] poincare-line-inversion[of u v]
  unfolding circline-set-def
  by auto
thus ?thesis
  using * assms False
  unfolding poincare-collinear-def
  by (rule-tac x=poincare-line u v in exI) simp
qed

```

```

lemma poincare-between-poincare-line-uvw:
  assumes u  $\neq$  v and u  $\in$  unit-disc and v  $\in$  unit-disc and
    z  $\in$  unit-disc and poincare-between u v z
  shows z  $\in$  circline-set (poincare-line u v)
  using assms
  using poincare-between-poincare-collinear[of u v z]
  using unique-poincare-line[OF assms(1-3)]
  unfolding poincare-collinear-def
  by auto

```

```

lemma poincare-between-poincare-line-uzv:
  assumes u  $\neq$  v and u  $\in$  unit-disc and v  $\in$  unit-disc and
    z  $\in$  unit-disc poincare-between u z v
  shows z  $\in$  circline-set (poincare-line u v)
  using assms
  using poincare-between-poincare-collinear[of u z v]
  using unique-poincare-line[OF assms(1-3)]
  unfolding poincare-collinear-def
  by auto

```

#### 7.1.4 H-betweenness on Euclidean segments

If the three points lie on an h-line that is a Euclidean line (e.g., if it contains zero), h-betweenness can be characterized much simpler than in the definition.

```

lemma poincare-between-x-axis-u0v:
  assumes is-real u' and u'  $\neq$  0 and v'  $\neq$  0
  shows poincare-between (of-complex u') 0h (of-complex v')  $\longleftrightarrow$  is-real v'  $\wedge$  Re u' * Re v' < 0

```

**proof**–

```

  have Re u'  $\neq$  0
    using  $\langle$ is-real u' $\rangle$   $\langle$ u'  $\neq$  0 $\rangle$ 
    using complex-eq-if-Re-eq
    by auto
  have nz: of-complex u'  $\neq$  0h of-complex v'  $\neq$  0h
    by (simp-all add:  $\langle$ u'  $\neq$  0 $\rangle$   $\langle$ v'  $\neq$  0 $\rangle$ )
  hence 0h  $\neq$  of-complex v'
    by metis

```

```

  let ?cr = cross-ratio (of-complex u') 0h (of-complex v')  $\infty$ h
  have is-real (to-complex ?cr)  $\wedge$  Re (to-complex ?cr) < 0  $\longleftrightarrow$  is-real v'  $\wedge$  Re u' * Re v' < 0
    using cross-ratio-0inf[of v' u']  $\langle$ v'  $\neq$  0 $\rangle$   $\langle$ u'  $\neq$  0 $\rangle$   $\langle$ is-real u' $\rangle$ 
    by (metis Re-complex-div-lt-0 Re-mult-real complex-cnj-divide divide-cancel-left eq-cnj-iff-real to-complex-of-complex)
  thus ?thesis
    unfolding poincare-between-def inversion-zero
    using  $\langle$ of-complex u'  $\neq$  0h $\rangle$   $\langle$ 0h  $\neq$  of-complex v' $\rangle$ 

```



by simp  
qed

lemma *poincare-between-u0v*:

assumes  $u \in \text{unit-disc}$  and  $v \in \text{unit-disc}$  and  $u \neq 0_h$  and  $v \neq 0_h$

shows  $\text{poincare-between } u \ 0_h \ v \longleftrightarrow (\exists k < 0. \text{to-complex } u = \text{cor } k * \text{to-complex } v)$  (is ?P u v)

proof (cases  $u = v$ )

case True

thus ?thesis

using *assms*

using *inf-or-of-complex*[of v]

using *poincare-between-sandwich*[of u  $0_h$ ]

by auto

next

case False

have  $\forall u. u \in \text{unit-disc} \wedge u \neq 0_h \longrightarrow ?P \ u \ v$  (is ?P' v)

proof (rule *wlog-rotation-to-positive-x-axis*)

fix  $\varphi \ v$

let ?M = *moebius-pt* (*moebius-rotation*  $\varphi$ )

assume 1:  $v \in \text{unit-disc}$   $v \neq 0_h$

assume 2: ?P' (?M v)

show ?P' v

proof (rule *allI*, *rule impI*, (*erule conjE*)+)

fix u

assume 3:  $u \in \text{unit-disc}$   $u \neq 0_h$

have  $\text{poincare-between } (?M \ u) \ 0_h \ (?M \ v) \longleftrightarrow \text{poincare-between } u \ 0_h \ v$

using  $\langle u \in \text{unit-disc} \rangle \langle v \in \text{unit-disc} \rangle$

using *unit-disc-fix-moebius-preserve-poincare-between unit-disc-fix-rotation zero-in-unit-disc*

by *fastforce*

thus ?P u v

using 1 2[*rule-format*, of ?M u] 3

using *inf-or-of-complex*[of u] *inf-or-of-complex*[of v]

by auto

qed

next

fix x

assume 1: *is-real* x  $0 < \text{Re } x$   $\text{Re } x < 1$

hence  $x \neq 0$

by auto

show ?P' (*of-complex* x)

proof (rule *allI*, *rule impI*, (*erule conjE*)+)

fix u

assume 2:  $u \in \text{unit-disc}$   $u \neq 0_h$

then obtain u' where  $u = \text{of-complex } u'$

using *inf-or-of-complex*[of u]

by auto

show ?P u (*of-complex* x)

using 1 2  $\langle x \neq 0 \rangle \langle u = \text{of-complex } u' \rangle$

using *poincare-between-rev*[of u  $0_h$  *of-complex* x]

using *poincare-between-x-axis-u0v*[of x u']  $\langle \text{is-real } x \rangle$

apply (*auto simp add: cmod-eq-Re*)

apply (*rule-tac*  $x = \text{Re } u' / \text{Re } x$  in *exI*, *simp add: divide-neg-pos algebra-split-simps*)

using *mult-neg-pos mult-pos-neg*

by *blast*

qed

qed *fact+*

thus ?thesis

using *assms*

by auto

qed

lemma *poincare-between-u0v-polar-form*:

assumes  $x \in \text{unit-disc}$  and  $y \in \text{unit-disc}$  and  $x \neq 0_h$  and  $y \neq 0_h$  and

$\text{to-complex } x = \text{cor } rx * \text{cis } \varphi$   $\text{to-complex } y = \text{cor } ry * \text{cis } \varphi$

shows  $\text{poincare-between } x \ 0_h \ y \longleftrightarrow rx * ry < 0$  (is ?P x y rx ry)

proof—

```

from assms have  $rx \neq 0 \text{ } ry \neq 0$ 
  using inf-or-of-complex[of x] inf-or-of-complex[of y]
  by auto

have  $(\exists k < 0. \text{ cor } rx = \text{ cor } k * \text{ cor } ry) = (rx * ry < 0)$ 
proof
  assume  $\exists k < 0. \text{ cor } rx = \text{ cor } k * \text{ cor } ry$ 
  then obtain k where  $k < 0 \text{ cor } rx = \text{ cor } k * \text{ cor } ry$ 
    by auto
  hence  $rx = k * ry$ 
    using of-real-eq-iff
    by fastforce
  thus  $rx * ry < 0$ 
    using  $\langle k < 0 \rangle \langle rx \neq 0 \rangle \langle ry \neq 0 \rangle$ 
    by (smt divisors-zero mult-nonneg-nonpos mult-nonpos-nonpos zero-less-mult-pos2)
next
  assume  $rx * ry < 0$ 
  hence  $rx = (rx/ry)*ry \text{ } rx / ry < 0$ 
    using  $\langle rx \neq 0 \rangle \langle ry \neq 0 \rangle$ 
    by (auto simp add: divide-less-0-iff algebra-split-simps)
  thus  $\exists k < 0. \text{ cor } rx = \text{ cor } k * \text{ cor } ry$ 
    using  $\langle rx \neq 0 \rangle \langle ry \neq 0 \rangle$ 
    by (rule-tac x=rx / ry in exI, simp)
qed
thus ?thesis
  using assms
  using poincare-between-u0v[OF assms(1-4)]
  by auto
qed

lemma poincare-between-x-axis-0uw:
  fixes x y :: real
  assumes  $-1 < x \text{ and } x < 1 \text{ and } x \neq 0$ 
  assumes  $-1 < y \text{ and } y < 1 \text{ and } y \neq 0$ 
  shows poincare-between  $0_h$  (of-complex x) (of-complex y)  $\longleftrightarrow$ 
     $(x < 0 \wedge y < 0 \wedge y \leq x) \vee (x > 0 \wedge y > 0 \wedge x \leq y)$  (is ?lhs  $\longleftrightarrow$  ?rhs)
proof (cases x = y)
  case True
  thus ?thesis
    using assms
    unfolding poincare-between-def
    by auto
next
  case False
  let ?x = of-complex x and ?y = of-complex y

  have ?x  $\in$  unit-disc ?y  $\in$  unit-disc
    using assms
    by auto

  have distinct: distinct [ $0_h$ , ?x, ?y, inversion ?x]
    using  $\langle x \neq 0 \rangle \langle y \neq 0 \rangle \langle x \neq y \rangle \langle ?x \in \text{unit-disc} \rangle \langle ?y \in \text{unit-disc} \rangle$ 
    using inversion-noteq-unit-disc[of ?x ?y]
    using inversion-noteq-unit-disc[of ?x ?x]
    using inversion-noteq-unit-disc[of ?x 0_h]
    using of-complex-inj[of x y]
    by (metis distinct-length-2-or-more distinct-singleton of-complex-zero-iff of-real-eq-0-iff of-real-eq-iff zero-in-unit-disc)

  let ?cr = cross-ratio  $0_h$  ?x ?y (inversion ?x)
  have Re (to-complex ?cr) =  $x^2 * (x*y - 1) / (x * (y - x))$ 
    using  $\langle x \neq 0 \rangle \langle x \neq y \rangle$ 
    unfolding inversion-def
    by simp (transfer, transfer, auto simp add: vec-cnj-def power2-eq-square field-simps split: if-split-asm)
moreover
{
  fix a b :: real

```

```

  assume  $b \neq 0$ 
  hence  $a < 0 \iff b^2 * a < (0::real)$ 
  by (metis mult.commute mult-eq-0-iff mult-neg-pos mult-pos-pos not-less-iff-gr-or-eq not-real-square-gt-zero power2-eq-square)
}
hence  $x^2 * (x*y - 1) < 0$ 
  using assms
  by (smt minus-mult-minus mult-le-cancel-left1)
moreover
have  $x * (y - x) > 0 \iff ?rhs$ 
  using  $\langle x \neq 0 \rangle \langle y \neq 0 \rangle \langle x \neq y \rangle$ 
  by (smt mult-le-0-iff)
ultimately
have *:  $Re (to-complex ?cr) < 0 \iff ?rhs$ 
  by (simp add: divide-less-0-iff)

show ?thesis
proof
  assume ?lhs
  have is-real (to-complex ?cr)  $Re (to-complex ?cr) < 0$ 
    using  $\langle ?lhs \rangle$  distinct
    unfolding poincare-between-def Let-def
    by auto
  thus ?rhs
    using *
    by simp
next
  assume ?rhs
  hence  $Re (to-complex ?cr) < 0$ 
    using *
    by simp
  moreover
  have  $\{0_h, \text{of-complex } (cor\ x), \text{of-complex } (cor\ y), \text{inversion } (\text{of-complex } (cor\ x))\} \subseteq \text{circline-set } x\text{-axis}$ 
    using  $\langle x \neq 0 \rangle$  is-real-inversion[of cor x]
    using inf-or-of-complex[of inversion ?x]
    by (auto simp del: inversion-of-complex)
  hence is-real (to-complex ?cr)
    using four-points-on-circline-iff-cross-ratio-real[OF distinct]
    by auto
  ultimately
  show ?lhs
    using distinct
    unfolding poincare-between-def Let-def
    by auto
qed
qed

```

lemma poincare-between-0uv:

```

assumes  $u \in \text{unit-disc}$  and  $v \in \text{unit-disc}$  and  $u \neq 0_h$  and  $v \neq 0_h$ 
shows poincare-between  $0_h$   $u$   $v \iff$ 
  (let  $u' = \text{to-complex } u$ ;  $v' = \text{to-complex } v$  in  $\text{Arg } u' = \text{Arg } v' \wedge \text{cmod } u' \leq \text{cmod } v'$ ) (is ?P  $u$   $v$ )
proof (cases  $u = v$ )
  case True
  thus ?thesis
    by simp
next
  case False
  have  $\forall v. v \in \text{unit-disc} \wedge v \neq 0_h \wedge v \neq u \implies (\text{poincare-between } 0_h\ u\ v \iff (\text{let } u' = \text{to-complex } u; v' = \text{to-complex } v \text{ in } \text{Arg } u' = \text{Arg } v' \wedge \text{cmod } u' \leq \text{cmod } v'))$  (is ?P'  $u$ )
  proof (rule wlog-rotation-to-positive-x-axis)
    show  $u \in \text{unit-disc}$   $u \neq 0_h$ 
      by fact+
  next
  fix  $x$ 
  assume *: is-real  $x$   $0 < Re\ x$   $Re\ x < 1$ 
  hence of-complex  $x \in \text{unit-disc}$  of-complex  $x \neq 0_h$  of-complex  $x \in \text{circline-set } x\text{-axis}$ 
    unfolding circline-set-x-axis

```

```

  by (auto simp add: cmod-eq-Re)
show ?P' (of-complex x)
proof safe
  fix v
  assume v ∈ unit-disc v ≠ 0h v ≠ of-complex x poincare-between 0h (of-complex x) v
  hence v ∈ circline-set x-axis
    using poincare-between-poincare-line-uvw[of 0h of-complex x v]
    using poincare-line-0-real-is-x-axis[of of-complex x]
    using ⟨of-complex x ≠ 0h⟩ ⟨v ≠ 0h⟩ ⟨v ≠ of-complex x⟩ ⟨of-complex x ∈ unit-disc⟩ ⟨of-complex x ∈ circline-set
x-axis⟩
  by auto
obtain v' where v = of-complex v'
  using ⟨v ∈ unit-disc⟩
  using inf-or-of-complex[of v]
  by auto
hence **: v = of-complex v' - 1 < Re v' Re v' < 1 Re v' ≠ 0 is-real v'
  using ⟨v ∈ unit-disc⟩ ⟨v ≠ 0h⟩ ⟨v ∈ circline-set x-axis⟩ of-complex-inj[of v']
  unfolding circline-set-x-axis
  by (auto simp add: cmod-eq-Re real-imag-0)
show let u' = to-complex (of-complex x); v' = to-complex v in Arg u' = Arg v' ∧ cmod u' ≤ cmod v'
  using poincare-between-x-axis-0uv[of Re x Re v'] * **
  using ⟨poincare-between 0h (of-complex x) v⟩
  using arg-complex-of-real-positive[of Re x] arg-complex-of-real-negative[of Re x]
  using arg-complex-of-real-positive[of Re v'] arg-complex-of-real-negative[of Re v']
  by (auto simp add: cmod-eq-Re)
next
fix v
assume v ∈ unit-disc v ≠ 0h v ≠ of-complex x
then obtain v' where **: v = of-complex v' v' ≠ 0 v' ≠ x
  using inf-or-of-complex[of v]
  by auto blast
assume let u' = to-complex (of-complex x); v' = to-complex v in Arg u' = Arg v' ∧ cmod u' ≤ cmod v'
hence ***: Re x < 0 ∧ Re v' < 0 ∧ Re v' ≤ Re x ∨ 0 < Re x ∧ 0 < Re v' ∧ Re x ≤ Re v' is-real v'
  using arg-pi-iff[of x] arg-pi-iff[of v']
  using arg-0-iff[of x] arg-0-iff[of v']
  using * **
  by (smt cmod-Re-le-iff to-complex-of-complex)+
have -1 < Re v' Re v' < 1 Re v' ≠ 0 is-real v'
  using ⟨v ∈ unit-disc⟩ ** ⟨is-real v'⟩
  by (auto simp add: cmod-eq-Re complex-eq-if-Re-eq)
thus poincare-between 0h (of-complex x) v
  using poincare-between-x-axis-0uv[of Re x Re v'] * ** ***
  by simp
qed
next
fix φ u
assume u ∈ unit-disc u ≠ 0h
let ?M = moebius-rotation φ
assume *: ?P' (moebius-pt ?M u)
show ?P' u
proof (rule allI, rule impI, (erule conjE)+)
  fix v
  assume v ∈ unit-disc v ≠ 0h v ≠ u
  have moebius-pt ?M v ≠ moebius-pt ?M u
    using ⟨v ≠ u⟩
    by auto
obtain u' v' where v = of-complex v' u = of-complex u' v' ≠ 0 u' ≠ 0
  using inf-or-of-complex[of u] inf-or-of-complex[of v]
  using ⟨v ∈ unit-disc⟩ ⟨u ∈ unit-disc⟩ ⟨v ≠ 0h⟩ ⟨u ≠ 0h⟩
  by auto
thus ?P u v
  using *[rule-format, of moebius-pt ?M v]
  using ⟨moebius-pt ?M v ≠ moebius-pt ?M u⟩
  using unit-disc-fix-moebius-preserve-poincare-between[of ?M 0h u v]
  using ⟨v ∈ unit-disc⟩ ⟨u ∈ unit-disc⟩ ⟨v ≠ 0h⟩ ⟨u ≠ 0h⟩
  using arg-mult-eq[of cis φ u' v']

```

```

    by simp (auto simp add: arg-mult norm-mult)
  qed
qed
thus ?thesis
  using assms False
  by auto
qed

lemma poincare-between-y-axis-0uv:
  fixes x y :: real
  assumes  $-1 < x$  and  $x < 1$  and  $x \neq 0$ 
  assumes  $-1 < y$  and  $y < 1$  and  $y \neq 0$ 
  shows poincare-between  $0_h$  (of-complex (i * x)) (of-complex (i * y))  $\longleftrightarrow$ 
    ( $x < 0 \wedge y < 0 \wedge y \leq x$ )  $\vee$  ( $x > 0 \wedge y > 0 \wedge x \leq y$ ) (is ?lhs  $\longleftrightarrow$  ?rhs)
  using assms
  using poincare-between-0uv[of of-complex (i * x) of-complex (i * y)]
  using arg-pi2-iff[of i * cor x] arg-pi2-iff[of i * cor y]
  using arg-minus-pi2-iff[of i * cor x] arg-minus-pi2-iff[of i * cor y]
  apply (simp add: norm-mult)
  apply (smt (verit, best))
  done

lemma poincare-between-x-axis-uvw:
  fixes x y z :: real
  assumes  $-1 < x$  and  $x < 1$ 
  assumes  $-1 < y$  and  $y < 1$  and  $y \neq x$ 
  assumes  $-1 < z$  and  $z < 1$  and  $z \neq x$ 
  shows poincare-between (of-complex x) (of-complex y) (of-complex z)  $\longleftrightarrow$ 
    ( $y < x \wedge z < x \wedge z \leq y$ )  $\vee$  ( $y > x \wedge z > x \wedge y \leq z$ ) (is ?lhs  $\longleftrightarrow$  ?rhs)
proof (cases  $x = 0 \vee y = 0 \vee z = 0$ )
  case True
  thus ?thesis
  proof (cases  $x = 0$ )
    case True
    thus ?thesis
      using poincare-between-x-axis-0uv assms
      by simp
  next
  case False
  show ?thesis
  proof (cases  $z = 0$ )
    case True
    thus ?thesis
      using poincare-between-x-axis-0uv assms poincare-between-rev
      by (smt norm-of-real of-complex-zero of-real-0 poincare-between-nonstrict(2) unit-disc-iff-cmod-lt-1)
  next
  case False
  have  $y = 0$ 
    using  $\langle x \neq 0 \rangle \langle z \neq 0 \rangle \langle x = 0 \vee y = 0 \vee z = 0 \rangle$ 
    by simp
  have poincare-between (of-complex x)  $0_h$  (of-complex z) = (is-real z  $\wedge x * z < 0$ )
    using  $\langle x \neq 0 \rangle \langle z \neq 0 \rangle$  poincare-between-x-axis-u0v
    by auto
  moreover
  have  $x * z < 0 \longleftrightarrow ?rhs$ 
    using True  $\langle x \neq 0 \rangle \langle z \neq 0 \rangle$ 
    by (smt zero-le-mult-iff)
  ultimately
  show ?thesis
    using  $\langle y = 0 \rangle$ 
    by auto
  qed
qed
qed
next
case False

```

```

thus ?thesis
proof (cases z = y)
  case True
    thus ?thesis
      using assms
      unfolding poincare-between-def
      by auto
next
  case False
let ?x = of-complex x and ?y = of-complex y and ?z = of-complex z

have ?x ∈ unit-disc ?y ∈ unit-disc ?z ∈ unit-disc
  using assms
  by auto

have distinct: distinct [?x, ?y, ?z, inversion ?y]
  using ⟨y ≠ x⟩ ⟨z ≠ x⟩ False ⟨?x ∈ unit-disc⟩ ⟨?y ∈ unit-disc⟩ ⟨?z ∈ unit-disc⟩
  using inversion-noteq-unit-disc[of ?y ?y]
  using inversion-noteq-unit-disc[of ?y ?x]
  using inversion-noteq-unit-disc[of ?y ?z]
  using of-complex-inj[of x y] of-complex-inj[of y z] of-complex-inj[of x z]
  by auto

have cor y * cor x ≠ 1
  using assms
  by (smt minus-mult-minus mult-less-cancel-left2 mult-less-cancel-right2 of-real-1 of-real-eq-iff of-real-mult)

let ?cr = cross-ratio ?x ?y ?z (inversion ?y)
have Re (to-complex ?cr) = (x - y) * (z*y - 1) / ((x*y - 1)*(z - y))
proof-
  have  $\bigwedge y x z. \llbracket y \neq x; z \neq x; z \neq y; \text{cor } y * \text{cor } x \neq 1; x \neq 0; y \neq 0; z \neq 0 \rrbracket \implies$ 
    
$$\frac{(y * y + y * (y * (x * z)) - (y * x + y * (y * (y * z)))) / (y * y + y * (y * (x * z)) - (y * z + y * (y * (y * x))))}{(y + y * (x * z) - (x + y * (y * z))) / (y + y * (x * z) - (z + y * (y * x)))}$$

  by (metis (no-types, opaque-lifting) ab-group-add-class.ab-diff-conv-add-uminus distrib-left mult-divide-mult-cancel-left-if
  mult-minus-right)
  thus ?thesis
    using ⟨y ≠ x⟩ ⟨z ≠ x⟩ False ⟨¬ (x = 0 ∨ y = 0 ∨ z = 0)⟩
    using ⟨cor y * cor x ≠ 1⟩
    unfolding inversion-def
    by (transfer, transfer, auto simp add: vec-cnj-def power2-eq-square field-simps split: if-split-asm)
qed

moreover
have (x*y - 1) < 0
  using assms
  by (smt minus-mult-minus mult-less-cancel-right2 zero-less-mult-iff)
moreover
have (z*y - 1) < 0
  using assms
  by (smt minus-mult-minus mult-less-cancel-right2 zero-less-mult-iff)
moreover
have (x - y) / (z - y) < 0  $\longleftrightarrow$  ?rhs
  using ⟨y ≠ x⟩ ⟨z ≠ x⟩ False ⟨¬ (x = 0 ∨ y = 0 ∨ z = 0)⟩
  by (smt divide-less-cancel divide-nonneg-nonpos divide-nonneg-pos divide-nonpos-nonneg divide-nonpos-nonpos)
ultimately
have *: Re (to-complex ?cr) < 0  $\longleftrightarrow$  ?rhs
  by (smt algebra-split-simps(24) minus-divide-left zero-less-divide-iff zero-less-mult-iff)
show ?thesis
proof
  assume ?lhs
  have is-real (to-complex ?cr) Re (to-complex ?cr) < 0
    using ⟨?lhs⟩ distinct
    unfolding poincare-between-def Let-def
    by auto
  thus ?rhs

```

```

    using *
    by simp
next
assume ?rhs
hence Re (to-complex ?cr) < 0
    using *
    by simp
moreover
have {of-complex (cor x), of-complex (cor y), of-complex (cor z), inversion (of-complex (cor y))} ⊆ circline-set
x-axis
    using ⟨¬ (x = 0 ∨ y = 0 ∨ z = 0)⟩ is-real-inversion[of cor y]
    using inf-or-of-complex[of inversion ?y]
    by (auto simp del: inversion-of-complex)
hence is-real (to-complex ?cr)
    using four-points-on-circline-iff-cross-ratio-real[OF distinct]
    by auto
ultimately
show ?lhs
    using distinct
    unfolding poincare-between-def Let-def
    by auto
qed
qed
qed

```

### 7.1.5 H-betweenness and h-collinearity

For three h-collinear points at least one of the three possible h-betweenness relations must hold.

lemma poincare-collinear3-between:

```

assumes u ∈ unit-disc and v ∈ unit-disc and w ∈ unit-disc
assumes poincare-collinear {u, v, w}
shows poincare-between u v w ∨ poincare-between u w v ∨ poincare-between v u w (is ?P' u v w)
proof (cases u=v)
case True
thus ?thesis
    using assms
    by auto
next
case False
have ∀ w. w ∈ unit-disc ∧ poincare-collinear {u, v, w} → ?P' u v w (is ?P u v)
proof (rule wlog-positive-x-axis[where P=?P])
fix x
assume x: is-real x 0 < Re x Re x < 1
hence x ≠ 0
    using complex.expand[of x 0]
    by auto
hence *: poincare-line 0h (of-complex x) = x-axis
    using x poincare-line-0-real-is-x-axis[of of-complex x]
    unfolding circline-set-x-axis
    by auto
have of-complex x ∈ unit-disc
    using x
    by (auto simp add: cmod-eq-Re)
have of-complex x ≠ 0h
    using ⟨x ≠ 0⟩
    by auto
show ?P 0h (of-complex x)
proof safe
fix w
assume w ∈ unit-disc
assume poincare-collinear {0h, of-complex x, w}
hence w ∈ circline-set x-axis
    using * unique-poincare-line[of 0h of-complex x] ⟨of-complex x ∈ unit-disc⟩ ⟨x ≠ 0⟩ ⟨of-complex x ≠ 0h⟩
    unfolding poincare-collinear-def
    by auto
then obtain w' where w': w = of-complex w' is-real w'

```

```

    using ⟨w ∈ unit-disc⟩
    using inf-or-of-complex[of w]
    unfolding circline-set-x-axis
    by auto
  hence  $-1 < \operatorname{Re} w' \operatorname{Re} w' < 1$ 
    using ⟨w ∈ unit-disc⟩
    by (auto simp add: cmod-eq-Re)
  assume 1:  $\neg \operatorname{poincare-between} (\operatorname{of-complex} x) 0_h w$ 
  hence  $w \neq 0_h \ w' \neq 0$ 
    using w'
    unfolding poincare-between-def
    by auto
  hence  $\operatorname{Re} w' \neq 0$ 
    using w' complex.expand[of w' 0]
    by auto

  have  $\operatorname{Re} w' \geq 0$ 
    using 1 poincare-between-x-axis-u0v[of x w'] ⟨Re x > 0⟩ ⟨is-real x⟩ ⟨x ≠ 0⟩ ⟨w' ≠ 0⟩ w'
    using mult-pos-neg
    by force

  moreover

  assume  $\neg \operatorname{poincare-between} 0_h (\operatorname{of-complex} x) w$ 
  hence  $\operatorname{Re} w' < \operatorname{Re} x$ 
    using poincare-between-x-axis-0uv[of Re x Re w']
    using w' x ⟨-1 < Re w'⟩ ⟨Re w' < 1⟩ ⟨Re w' ≠ 0⟩
    by auto

  ultimately
  show  $\operatorname{poincare-between} 0_h w (\operatorname{of-complex} x)$ 
    using poincare-between-x-axis-0uv[of Re w' Re x]
    using w' x ⟨-1 < Re w'⟩ ⟨Re w' < 1⟩ ⟨Re w' ≠ 0⟩
    by auto
qed
next
show  $u \in \operatorname{unit-disc} \ v \in \operatorname{unit-disc} \ u \neq v$ 
  by fact+
next
fix M u v
assume 1:  $\operatorname{unit-disc-fix} M u \in \operatorname{unit-disc} \ v \in \operatorname{unit-disc} \ u \neq v$ 
let ?Mu = moebius-pt M u and ?Mv = moebius-pt M v
assume 2:  $?P \ ?Mu \ ?Mv$ 
show  $?P \ u \ v$ 
proof safe
  fix w
  assume  $w \in \operatorname{unit-disc} \ \operatorname{poincare-collinear} \ {u, v, w} \ \neg \operatorname{poincare-between} \ u \ v \ w \ \neg \operatorname{poincare-between} \ v \ u \ w$ 
  thus  $\operatorname{poincare-between} \ u \ w \ v$ 
    using 1 2[rule-format, of moebius-pt M w]
    by simp
qed
qed
thus ?thesis
  using assms
  by simp
qed

lemma poincare-collinear3-iff:
  assumes  $u \in \operatorname{unit-disc} \ v \in \operatorname{unit-disc} \ w \in \operatorname{unit-disc}$ 
  shows  $\operatorname{poincare-collinear} \ {u, v, w} \ \longleftrightarrow \ \operatorname{poincare-between} \ u \ v \ w \ \vee \ \operatorname{poincare-between} \ v \ u \ w \ \vee \ \operatorname{poincare-between} \ v \ w \ u$ 
  using assms
  by (metis poincare-collinear3-between insert-commute poincare-between-poincare-collinear poincare-between-rev)

```

## 7.2 Some properties of betweenness

lemma *poincare-between-transitivity*:



```

assumes  $a \in \text{unit-disc}$  and  $x \in \text{unit-disc}$  and  $b \in \text{unit-disc}$  and  $y \in \text{unit-disc}$  and
   $\text{poincare-between } a \ x \ b$  and  $\text{poincare-between } a \ b \ y$ 
shows  $\text{poincare-between } x \ b \ y$ 
proof(cases  $a = b$ )
  case True
    thus ?thesis
    using assms
    using poincare-between-sandwich by blast
next
  case False
have  $\forall x. \forall y. \text{poincare-between } a \ x \ b \wedge \text{poincare-between } a \ b \ y \wedge x \in \text{unit-disc}$ 
   $\wedge y \in \text{unit-disc} \longrightarrow \text{poincare-between } x \ b \ y$  (is ? $P \ a \ b$ )
proof (rule wlog-positive-x-axis[where  $P = ?P$ ])
  show  $a \in \text{unit-disc}$ 
  using assms by simp
next
  show  $b \in \text{unit-disc}$ 
  using assms by simp
next
  show  $a \neq b$ 
  using False by simp
next
  fix  $M \ u \ v$ 
assume *:  $\text{unit-disc-fix } M \ u \in \text{unit-disc}$   $v \in \text{unit-disc}$   $u \neq v$ 
   $\forall x \ y. \text{poincare-between } (\text{moebius-pt } M \ u) \ x \ (\text{moebius-pt } M \ v) \wedge$ 
   $\text{poincare-between } (\text{moebius-pt } M \ u) \ (\text{moebius-pt } M \ v) \ y \wedge$ 
   $x \in \text{unit-disc} \wedge y \in \text{unit-disc} \longrightarrow$ 
   $\text{poincare-between } x \ (\text{moebius-pt } M \ v) \ y$ 
show  $\forall x \ y. \text{poincare-between } u \ x \ v \wedge \text{poincare-between } u \ v \ y \wedge x \in \text{unit-disc} \wedge y \in \text{unit-disc}$ 
   $\longrightarrow \text{poincare-between } x \ v \ y$ 
proof safe
  fix  $x \ y$ 
  assume  $\text{poincare-between } u \ x \ v \ \text{poincare-between } u \ v \ y \ x \in \text{unit-disc} \ y \in \text{unit-disc}$ 

  have  $\text{poincare-between } (\text{moebius-pt } M \ u) \ (\text{moebius-pt } M \ x) \ (\text{moebius-pt } M \ v)$ 
  using  $\langle \text{poincare-between } u \ x \ v \rangle \langle \text{unit-disc-fix } M \rangle \langle x \in \text{unit-disc} \rangle \langle u \in \text{unit-disc} \rangle \langle v \in \text{unit-disc} \rangle$ 
  by simp
  moreover
  have  $\text{poincare-between } (\text{moebius-pt } M \ u) \ (\text{moebius-pt } M \ v) \ (\text{moebius-pt } M \ y)$ 
  using  $\langle \text{poincare-between } u \ v \ y \rangle \langle \text{unit-disc-fix } M \rangle \langle y \in \text{unit-disc} \rangle \langle u \in \text{unit-disc} \rangle \langle v \in \text{unit-disc} \rangle$ 
  by simp
  moreover
  have  $(\text{moebius-pt } M \ x) \in \text{unit-disc}$ 
  using  $\langle \text{unit-disc-fix } M \rangle \langle x \in \text{unit-disc} \rangle$  by simp
  moreover
  have  $(\text{moebius-pt } M \ y) \in \text{unit-disc}$ 
  using  $\langle \text{unit-disc-fix } M \rangle \langle y \in \text{unit-disc} \rangle$  by simp
  ultimately
  have  $\text{poincare-between } (\text{moebius-pt } M \ x) \ (\text{moebius-pt } M \ v) \ (\text{moebius-pt } M \ y)$ 
  using * by blast
  thus  $\text{poincare-between } x \ v \ y$ 
  using  $\langle y \in \text{unit-disc} \rangle \ast \langle x \in \text{unit-disc} \rangle$  by simp
qed
next
  fix  $x$ 
assume  $xx: \text{is-real } x \ 0 < \text{Re } x \ \text{Re } x < 1$ 
hence  $\text{of-complex } x \in \text{unit-disc}$ 
  using cmmod-eq-Re by auto
hence  $\text{of-complex } x \neq \infty_h$ 
  by simp
have  $\text{of-complex } x \neq 0_h$ 
  using  $xx$  by auto
have  $\text{of-complex } x \in \text{circline-set } x\text{-axis}$ 
  using  $xx$  by simp
show  $\forall m \ n. \text{poincare-between } 0_h \ m \ (\text{of-complex } x) \wedge \text{poincare-between } 0_h \ (\text{of-complex } x) \ n \wedge$ 
   $m \in \text{unit-disc} \wedge n \in \text{unit-disc} \longrightarrow \text{poincare-between } m \ (\text{of-complex } x) \ n$ 

```

```

proof safe
  fix  $m\ n$ 
  assume **: poincare-between  $0_h\ m$  (of-complex  $x$ ) poincare-between  $0_h$  (of-complex  $x$ )  $n$ 
     $m \in \text{unit-disc}$   $n \in \text{unit-disc}$ 
  show poincare-between  $m$  (of-complex  $x$ )  $n$ 
  proof(cases  $m = 0_h$ )
    case True
      thus ?thesis
      using ** by auto
    next
      case False
      hence  $m \in \text{circline-set } x\text{-axis}$ 
        using poincare-between-poincare-line-uzv[of  $0_h$  of-complex  $x\ m$ ]
        using poincare-line-0-real-is-x-axis[of of-complex  $x$ ]
        using  $\langle \text{of-complex } x \in \text{unit-disc} \rangle \langle \text{of-complex } x \neq \infty_h \rangle \langle \text{of-complex } x \neq 0_h \rangle$ 
        using  $\langle \text{of-complex } x \in \text{circline-set } x\text{-axis} \rangle \langle m \in \text{unit-disc} \rangle$  **(1)
        by simp
      then obtain  $m'$  where  $m = \text{of-complex } m'$  is-real  $m'$ 
        using inf-or-of-complex[of  $m$ ]  $\langle m \in \text{unit-disc} \rangle$ 
        unfolding circline-set-x-axis
        by auto
      hence  $\text{Re } m' \leq \text{Re } x$ 
        using  $\langle \text{poincare-between } 0_h\ m$  (of-complex  $x$ )  $\rangle\ xx$   $\langle \text{of-complex } x \neq 0_h \rangle$ 
        using False **  $\langle \text{of-complex } x \in \text{unit-disc} \rangle$ 
        using cmod-Re-le-iff poincare-between-0uv by auto

      have  $n \neq 0_h$ 
        using **(2, 4)  $\langle \text{of-complex } x \neq 0_h \rangle \langle \text{of-complex } x \in \text{unit-disc} \rangle$ 
        using poincare-between-sandwich by fastforce
      have  $n \in \text{circline-set } x\text{-axis}$ 
        using poincare-between-poincare-line-uvw[of  $0_h$  of-complex  $x\ n$ ]
        using poincare-line-0-real-is-x-axis[of of-complex  $x$ ]
        using  $\langle \text{of-complex } x \in \text{unit-disc} \rangle \langle \text{of-complex } x \neq \infty_h \rangle \langle \text{of-complex } x \neq 0_h \rangle$ 
        using  $\langle \text{of-complex } x \in \text{circline-set } x\text{-axis} \rangle \langle n \in \text{unit-disc} \rangle$  **(2)
        by simp
      then obtain  $n'$  where  $n = \text{of-complex } n'$  is-real  $n'$ 
        using inf-or-of-complex[of  $n$ ]  $\langle n \in \text{unit-disc} \rangle$ 
        unfolding circline-set-x-axis
        by auto
      hence  $\text{Re } x \leq \text{Re } n'$ 
        using  $\langle \text{poincare-between } 0_h$  (of-complex  $x$ )  $n$   $\rangle\ xx$   $\langle \text{of-complex } x \neq 0_h \rangle$ 
        using False **  $\langle \text{of-complex } x \in \text{unit-disc} \rangle \langle n \neq 0_h \rangle$ 
        using cmod-Re-le-iff poincare-between-0uv
        by (metis Re-complex-of-real arg-0-iff rcis-cmod-Arg rcis-zero-arg to-complex-of-complex)

      have poincare-between (of-complex  $m'$ ) (of-complex  $x$ ) (of-complex  $n'$ )
        using  $\langle \text{Re } x \leq \text{Re } n' \rangle \langle \text{Re } m' \leq \text{Re } x \rangle$ 
        using poincare-between-x-axis-uvw[of  $\text{Re } m'$   $\text{Re } x$   $\text{Re } n'$ ]
        using  $\langle \text{is-real } n' \rangle \langle \text{is-real } m' \rangle \langle n \in \text{unit-disc} \rangle \langle n = \text{of-complex } n' \rangle$ 
        using  $\langle \text{of-complex } m' \rangle \langle m \in \text{unit-disc} \rangle$ 
        by (smt complex-of-real-Re norm-of-real poincare-between-def unit-disc-iff-cmod-lt-1)

      thus ?thesis
        using  $\langle n = \text{of-complex } n' \rangle \langle m = \text{of-complex } m' \rangle$ 
        by auto
    qed
  qed
thus ?thesis
  using assms
by blast
qed

```

### 7.3 Poincare between - sum distances

Another possible definition of the h-betweenness relation is given in terms of h-distances between pairs of points. We prove it as a characterization equivalent to our cross-ratio based definition.

**lemma** *poincare-between-sum-distances-x-axis-u0v*:

**assumes** *of-complex*  $u' \in \text{unit-disc}$  *of-complex*  $v' \in \text{unit-disc}$

**assumes** *is-real*  $u' \neq 0$   $v' \neq 0$

**shows** *poincare-distance* (*of-complex*  $u'$ )  $0_h$  + *poincare-distance*  $0_h$  (*of-complex*  $v'$ ) = *poincare-distance* (*of-complex*  $u'$ ) (*of-complex*  $v'$ )  $\longleftrightarrow$

*is-real*  $v' \wedge \text{Re } u' * \text{Re } v' < 0$  (**is**  $?P$   $u' v' \longleftrightarrow ?Q$   $u' v'$ )

**proof**–

**have** *Re*  $u' \neq 0$

**using**  $\langle \text{is-real } u' \rangle \langle u' \neq 0 \rangle$

**using** *complex-eq-iff-Re-eq*

**by** *simp*

**let**  $?u = \text{cmod } u'$  **and**  $?v = \text{cmod } v'$  **and**  $?uv = \text{cmod } (u' - v')$

**have** *disc*:  $?u^2 < 1$   $?v^2 < 1$

**using** *unit-disc-cmod-square-lt-1*[*OF* *assms*(1)]

**using** *unit-disc-cmod-square-lt-1*[*OF* *assms*(2)]

**by** *auto*

**have** *poincare-distance* (*of-complex*  $u'$ )  $0_h$  + *poincare-distance*  $0_h$  (*of-complex*  $v'$ ) =

$\text{arcosh } (((1 + ?u^2) * (1 + ?v^2) + 4 * ?u * ?v) / ((1 - ?u^2) * (1 - ?v^2)))$  (**is** - = *arcosh*  $?r1$ )

**using** *poincare-distance-formula-zero-sum*[*OF* *assms*(1-2)]

**by** (*simp* *add*: *Let-def*)

**moreover**

**have** *poincare-distance* (*of-complex*  $u'$ ) (*of-complex*  $v'$ ) =

$\text{arcosh } (((1 - ?u^2) * (1 - ?v^2) + 2 * ?u * ?v) / ((1 - ?u^2) * (1 - ?v^2)))$  (**is** - = *arcosh*  $?r2$ )

**using** *disc*

**using** *poincare-distance-formula*[*OF* *assms*(1-2)]

**by** (*subst* *add-divide-distrib*) *simp*

**moreover**

**have** *arcosh*  $?r1 = \text{arcosh } ?r2 \longleftrightarrow ?Q$   $u' v'$

**proof**

**assume** *arcosh*  $?r1 = \text{arcosh } ?r2$

**hence**  $?r1 = ?r2$

**proof** (*subst* (*asm*) *arcosh-eq-iff*)

**show**  $?r1 \geq 1$

**proof**–

**have**  $(1 - ?u^2) * (1 - ?v^2) \leq (1 + ?u^2) * (1 + ?v^2) + 4 * ?u * ?v$

**by** (*simp* *add*: *field-simps*)

**thus** *?thesis*

**using** *disc*

**by** *simp*

**qed**

**next**

**show**  $?r2 \geq 1$

**using** *disc*

**by** *simp*

**qed**

**hence**  $(1 + ?u^2) * (1 + ?v^2) + 4 * ?u * ?v = (1 - ?u^2) * (1 - ?v^2) + 2 * ?uv^2$

**using** *disc*

**by** *auto*

**hence**  $(\text{cmod } (u' - v'))^2 = (\text{cmod } u' + \text{cmod } v')^2$

**by** (*simp* *add*: *field-simps*) *power2-eq-square*)

**hence**  $*$ :  $\text{Re } u' * \text{Re } v' + |\text{Re } u'| * \text{sqrt } ((\text{Im } v')^2 + (\text{Re } v')^2) = 0$

**using**  $\langle \text{is-real } u' \rangle$

**unfolding** *cmod-power2* *cmod-def*

**by** (*simp* *add*: *field-simps*) (*simp* *add*: *power2-eq-square*) *field-simps*)

**hence**  $\text{sqrt } ((\text{Im } v')^2 + (\text{Re } v')^2) = |\text{Re } v'|$

**using**  $\langle \text{Re } u' \neq 0 \rangle \langle v' \neq 0 \rangle$

**by** (*smt* *complex-neq-0* *mult.commute* *mult-cancel-right* *mult-minus-left* *real-sqrt-gt-0-iff*)

**hence**  $\text{Im } v' = 0$

**by** (*smt* *Im-eq-0* *norm-complex-def*)

**moreover**

**hence**  $\text{Re } u' * \text{Re } v' = - |\text{Re } u'| * |\text{Re } v'|$

```

    using *
    by simp
  hence  $Re\ u' * Re\ v' < 0$ 
    using  $\langle Re\ u' \neq 0 \rangle \langle v' \neq 0 \rangle$ 
    by (simp add:  $\langle is-real\ v' \rangle$  complex-eq-if-Re-eq)
  ultimately
  show  $?Q\ u'\ v'$ 
    by simp
next
  assume  $?Q\ u'\ v'$ 
  hence  $is-real\ v'\ Re\ u' * Re\ v' < 0$ 
    by auto
  have  $?r1 = ?r2$ 
  proof (cases  $Re\ u' > 0$ )
    case True
    hence  $Re\ v' < 0$ 
      using  $\langle Re\ u' * Re\ v' < 0 \rangle$ 
      by (smt zero-le-mult-iff)
    show  $?thesis$ 
      using disc  $\langle is-real\ u' \rangle \langle is-real\ v' \rangle$ 
      using  $\langle Re\ u' > 0 \rangle \langle Re\ v' < 0 \rangle$ 
      unfolding cmod-power2 cmod-def
      by simp (simp add: power2-eq-square field-simps)
  next
    case False
    hence  $Re\ u' < 0$ 
      using  $\langle Re\ u' \neq 0 \rangle$ 
      by simp
    hence  $Re\ v' > 0$ 
      using  $\langle Re\ u' * Re\ v' < 0 \rangle$ 
      by (smt zero-le-mult-iff)
    show  $?thesis$ 
      using disc  $\langle is-real\ u' \rangle \langle is-real\ v' \rangle$ 
      using  $\langle Re\ u' < 0 \rangle \langle Re\ v' > 0 \rangle$ 
      unfolding cmod-power2 cmod-def
      by simp (simp add: power2-eq-square field-simps)
  qed
  thus  $arcosh\ ?r1 = arcosh\ ?r2$ 
    by metis
qed
ultimately
show  $?thesis$ 
  by simp
qed

```

Different proof of the previous theorem relying on the cross-ratio definition, and not the distance formula. We suppose that this could be also used to prove the triangle inequality.

**lemma** *poincare-between-sum-distances-x-axis-u0v-different-proof*:

```

  assumes  $of-complex\ u' \in unit-disc\ of-complex\ v' \in unit-disc$ 
  assumes  $is-real\ u'\ u' \neq 0\ v' \neq 0\ is-real\ v'$ 
  shows  $poincare-distance\ (of-complex\ u')\ 0_h + poincare-distance\ 0_h\ (of-complex\ v') = poincare-distance\ (of-complex\ u')\ (of-complex\ v') \iff$ 
     $Re\ u' * Re\ v' < 0$  (is  $?P\ u'\ v' \iff ?Q\ u'\ v'$ )

```

**proof**–

```

  have  $-1 < Re\ u'\ Re\ u' < 1\ Re\ u' \neq 0$ 
    using assms
    by (auto simp add: cmod-eq-Re complex-eq-if-Re-eq)
  have  $-1 < Re\ v'\ Re\ v' < 1\ Re\ v' \neq 0$ 
    using assms
    by (auto simp add: cmod-eq-Re complex-eq-if-Re-eq)

```

```

  have  $|\ln\ (Re\ ((1 - u') / (1 + u')))| + |\ln\ (Re\ ((1 - v') / (1 + v')))| =$ 
     $|\ln\ (Re\ ((1 + u') * (1 - v') / ((1 - u') * (1 + v')))| \iff Re\ u' * Re\ v' < 0$  (is  $|\ln\ ?a1| + |\ln\ ?a2| = |\ln\ ?a3|$ 
 $\iff$  -)

```

**proof**–

```

  have  $1: 0 < ?a1\ \ln\ ?a1 > 0 \iff Re\ u' < 0$ 

```

```

using ⟨Re u' < 1⟩ ⟨Re u' > -1⟩ ⟨is-real u'⟩
using complex-is-Real-iff
by auto
have 2: 0 < ?a2 ln ?a2 > 0 ⟷ Re v' < 0
using ⟨Re v' < 1⟩ ⟨Re v' > -1⟩ ⟨is-real v'⟩
using complex-is-Real-iff
by auto
have 3: 0 < ?a3 ln ?a3 > 0 ⟷ Re v' < Re u'
using ⟨Re u' < 1⟩ ⟨Re u' > -1⟩ ⟨is-real u'⟩
using ⟨Re v' < 1⟩ ⟨Re v' > -1⟩ ⟨is-real v'⟩
using complex-is-Real-iff
by auto (simp add: field-simps)+
show ?thesis
proof
assume *: Re u' * Re v' < 0
show |ln ?a1| + |ln ?a2| = |ln ?a3|
proof (cases Re u' > 0)
case True
hence Re v' < 0
using *
by (smt mult-nonneg-nonneg)
show ?thesis
using 1 2 3 ⟨Re u' > 0⟩ ⟨Re v' < 0⟩
using ⟨Re u' < 1⟩ ⟨Re u' > -1⟩ ⟨is-real u'⟩
using ⟨Re v' < 1⟩ ⟨Re v' > -1⟩ ⟨is-real v'⟩
using complex-is-Real-iff
using ln-div ln-mult
by simp
next
case False
hence Re v' > 0 Re u' < 0
using *
by (smt zero-le-mult-iff)+
show ?thesis
using 1 2 3 ⟨Re u' < 0⟩ ⟨Re v' > 0⟩
using ⟨Re u' < 1⟩ ⟨Re u' > -1⟩ ⟨is-real u'⟩
using ⟨Re v' < 1⟩ ⟨Re v' > -1⟩ ⟨is-real v'⟩
using complex-is-Real-iff
using ln-div ln-mult
by simp
qed
next
assume *: |ln ?a1| + |ln ?a2| = |ln ?a3|
{
assume Re u' > 0 Re v' > 0
hence False
using * 1 2 3
using ⟨Re u' < 1⟩ ⟨Re u' > -1⟩ ⟨is-real u'⟩
using ⟨Re v' < 1⟩ ⟨Re v' > -1⟩ ⟨is-real v'⟩
using complex-is-Real-iff
using ln-mult ln-div
by (cases Re v' < Re u') auto
}
moreover
{
assume Re u' < 0 Re v' < 0
hence False
using * 1 2 3
using ⟨Re u' < 1⟩ ⟨Re u' > -1⟩ ⟨is-real u'⟩
using ⟨Re v' < 1⟩ ⟨Re v' > -1⟩ ⟨is-real v'⟩
using complex-is-Real-iff
using ln-mult ln-div
by (cases Re v' < Re u') auto
}
ultimately
show Re u' * Re v' < 0

```

```

    using ⟨Re u' ≠ 0⟩ ⟨Re v' ≠ 0⟩
    by (smt divisors-zero mult-le-0-iff)
qed
qed
thus ?thesis
using assms
apply (subst poincare-distance-sym, simp, simp)
apply (subst poincare-distance-zero-x-axis, simp, simp add: circline-set-x-axis)
apply (subst poincare-distance-zero-x-axis, simp, simp add: circline-set-x-axis)
apply (subst poincare-distance-x-axis-x-axis, simp, simp, simp add: circline-set-x-axis, simp add: circline-set-x-axis)
apply simp
done
qed

lemma poincare-between-sum-distances:
  assumes u ∈ unit-disc and v ∈ unit-disc and w ∈ unit-disc
  shows poincare-between u v w ↔
    poincare-distance u v + poincare-distance v w = poincare-distance u w (is ?P' u v w)
proof (cases u = v)
  case True
  thus ?thesis
    using assms
    by simp
next
  case False
  have ∀ w. w ∈ unit-disc → (poincare-between u v w ↔ poincare-distance u v + poincare-distance v w = poincare-distance
u w) (is ?P u v)
  proof (rule wlog-positive-x-axis)
    fix x
    assume is-real x 0 < Re x Re x < 1
    have of-complex x ∈ circline-set x-axis
      using ⟨is-real x⟩
      by (auto simp add: circline-set-x-axis)

    have of-complex x ∈ unit-disc
      using ⟨is-real x⟩ ⟨0 < Re x⟩ ⟨Re x < 1⟩
      by (simp add: cmod-eq-Re)

    have x ≠ 0
      using ⟨is-real x⟩ ⟨Re x > 0⟩
      by auto

    show ?P (of-complex x) 0h
  proof (rule allI, rule impI)
    fix w
    assume w ∈ unit-disc
    then obtain w' where w = of-complex w'
      using inf-or-of-complex[of w]
      by auto

    show ?P' (of-complex x) 0h w
  proof (cases w = 0h)
    case True
    thus ?thesis
      by simp
  next
    case False
    hence w' ≠ 0
      using ⟨w = of-complex w'⟩
      by auto

    show ?thesis
      using ⟨is-real x⟩ ⟨x ≠ 0⟩ ⟨w = of-complex w'⟩ ⟨w' ≠ 0⟩
      using ⟨of-complex x ∈ unit-disc⟩ ⟨w ∈ unit-disc⟩
      apply simp
      apply (subst poincare-between-x-axis-u0v, simp-all)

```

```

    apply (subst poincare-between-sum-distances-x-axis-u0v, simp-all)
  done
qed
qed
next
show  $v \in \text{unit-disc } u \in \text{unit-disc}$ 
  using assms
  by auto
next
show  $v \neq u$ 
  using  $\langle u \neq v \rangle$ 
  by simp
next
fix  $M u v$ 
assume *: unit-disc-fix  $M u \in \text{unit-disc } v \in \text{unit-disc } u \neq v$  and
  **:  $?P$  (moebius-pt  $M v$ ) (moebius-pt  $M u$ )
show  $?P v u$ 
proof (rule allI, rule impI)
  fix  $w$ 
  assume  $w \in \text{unit-disc}$ 
  hence moebius-pt  $M w \in \text{unit-disc}$ 
    using  $\langle \text{unit-disc-fix } M \rangle$ 
    by auto
  thus  $?P' v u w$ 
    using  $\langle u \in \text{unit-disc} \rangle \langle v \in \text{unit-disc} \rangle \langle w \in \text{unit-disc} \rangle \langle \text{unit-disc-fix } M \rangle$ 
    using **[rule-format, of moebius-pt  $M w$ ]
    by auto
qed
qed
thus ?thesis
  using assms
  by simp
qed

```

## 7.4 Some more properties of h-betweenness.

Some lemmas proved earlier are proved almost directly using the sum of distances characterization.

**lemma** *unit-disc-fix-moebius-preserve-poincare-between'*:

```

assumes unit-disc-fix  $M$  and  $u \in \text{unit-disc}$  and  $v \in \text{unit-disc}$  and  $w \in \text{unit-disc}$ 
shows poincare-between (moebius-pt  $M u$ ) (moebius-pt  $M v$ ) (moebius-pt  $M w$ )  $\longleftrightarrow$ 
  poincare-between  $u v w$ 
using assms
using poincare-between-sum-distances
by simp

```

**lemma** *conjugate-preserve-poincare-between'*:

```

assumes  $u \in \text{unit-disc } v \in \text{unit-disc } w \in \text{unit-disc}$ 
shows poincare-between (conjugate  $u$ ) (conjugate  $v$ ) (conjugate  $w$ )  $\longleftrightarrow$  poincare-between  $u v w$ 
using assms
using poincare-between-sum-distances
by simp

```

There is a unique point on a ray on the given distance from the given starting point

**lemma** *unique-poincare-distance-on-ray*:

```

assumes  $d \geq 0$   $u \neq v$   $u \in \text{unit-disc } v \in \text{unit-disc}$ 
assumes  $y \in \text{unit-disc}$  poincare-distance  $u y = d$  poincare-between  $u v y$ 
assumes  $z \in \text{unit-disc}$  poincare-distance  $u z = d$  poincare-between  $u v z$ 
shows  $y = z$ 
proof–
  have  $\forall d y z. d \geq 0 \wedge$ 
     $y \in \text{unit-disc} \wedge \text{poincare-distance } u y = d \wedge \text{poincare-between } u v y \wedge$ 
     $z \in \text{unit-disc} \wedge \text{poincare-distance } u z = d \wedge \text{poincare-between } u v z \longrightarrow y = z$  (is  $?P u v$ )
proof (rule wlog-positive-x-axis[where  $P=?P$ ])
  fix  $x$ 
  assume  $x$ : is-real  $x 0 < \text{Re } x \text{Re } x < 1$ 

```

```

hence  $x \neq 0$ 
  using complex.expand[of  $x$  0]
  by auto
hence *: poincare-line  $0_h$  (of-complex  $x$ ) = x-axis
  using  $x$  poincare-line-0-real-is-x-axis[of of-complex  $x$ ]
  unfolding circline-set-x-axis
  by auto
have of-complex  $x \in$  unit-disc
  using  $x$ 
  by (auto simp add: cmod-eq-Re)
have Arg  $x = 0$ 
  using  $x$ 
  using arg-0-iff by blast
show ? $P$   $0_h$  (of-complex  $x$ )
proof safe
  fix  $y z$ 
  assume  $y \in$  unit-disc  $z \in$  unit-disc
  then obtain  $y' z'$  where  $yz$ :  $y =$  of-complex  $y'$   $z =$  of-complex  $z'$ 
    using inf-or-of-complex[of  $y$ ] inf-or-of-complex[of  $z$ ]
    by auto
  assume betw: poincare-between  $0_h$  (of-complex  $x$ )  $y$  poincare-between  $0_h$  (of-complex  $x$ )  $z$ 
  hence  $y \neq 0_h$   $z \neq 0_h$ 
    using  $\langle x \neq 0 \rangle$   $\langle$ of-complex  $x \in$  unit-disc $\rangle$   $\langle y \in$  unit-disc $\rangle$ 
    using poincare-between-sandwich[of  $0_h$  of-complex  $x$ ]
    using of-complex-zero-iff[of  $x$ ]
    by force+

  hence Arg  $y' = 0$  cmod  $y' \geq$  cmod  $x$  Arg  $z' = 0$  cmod  $z' \geq$  cmod  $x$ 
    using poincare-between-0uw[of of-complex  $x$   $y$ ] poincare-between-0uw[of of-complex  $x$   $z$ ]
    using  $\langle$ of-complex  $x \in$  unit-disc $\rangle$   $\langle x \neq 0 \rangle$   $\langle$ Arg  $x = 0$  $\rangle$   $\langle y \in$  unit-disc $\rangle$   $\langle z \in$  unit-disc $\rangle$  betw  $yz$ 
    by (simp-all add: Let-def)
  hence *: is-real  $y'$  is-real  $z'$  Re  $y' > 0$  Re  $z' > 0$ 
    using arg-0-iff[of  $y'$ ] arg-0-iff[of  $z'$ ]  $x \langle y \neq 0_h \rangle$   $\langle z \neq 0_h \rangle$   $yz$ 
    by auto
  assume poincare-distance  $0_h$   $z =$  poincare-distance  $0_h$   $y$   $0 \leq$  poincare-distance  $0_h$   $y$ 
  thus  $y = z$ 
    using *  $yz$   $\langle y \in$  unit-disc $\rangle$   $\langle z \in$  unit-disc $\rangle$ 
    using unique-x-axis-poincare-distance-positive[of poincare-distance  $0_h$   $y$ ]
    by (auto simp add: cmod-eq-Re unit-disc-to-complex-inj)
qed
next
show  $u \in$  unit-disc  $v \in$  unit-disc  $u \neq v$ 
  by fact+
next
fix  $M u v$ 
assume *: unit-disc-fix  $M u \in$  unit-disc  $v \in$  unit-disc  $u \neq v$ 
assume **: ? $P$  (moebius-pt  $M u$ ) (moebius-pt  $M v$ )
show ? $P$   $u v$ 
proof safe
  fix  $d y z$ 
  assume ***:  $0 \leq$  poincare-distance  $u y$ 
     $y \in$  unit-disc poincare-between  $u v$   $y$ 
     $z \in$  unit-disc poincare-between  $u v$   $z$ 
    poincare-distance  $u z =$  poincare-distance  $u y$ 
  let ? $Mu =$  moebius-pt  $M u$  and ? $Mv =$  moebius-pt  $M v$  and ? $My =$  moebius-pt  $M y$  and ? $Mz =$  moebius-pt  $M z$ 
  have ? $Mu \in$  unit-disc ? $Mv \in$  unit-disc ? $My \in$  unit-disc ? $Mz \in$  unit-disc
    using  $\langle u \in$  unit-disc $\rangle$   $\langle v \in$  unit-disc $\rangle$   $\langle y \in$  unit-disc $\rangle$   $\langle z \in$  unit-disc $\rangle$ 
    using  $\langle$ unit-disc-fix  $M$  $\rangle$ 
    by auto
  hence ? $My =$  ? $Mz$ 
    using * ***
    using **[rule-format, of poincare-distance ? $Mu$  ? $My$  ? $My$  ? $Mz$ ]
    by simp
  thus  $y = z$ 
    using bij-moebius-pt[of  $M$ ]
    unfolding bij-def inj-on-def

```



```

    by blast
  qed
qed
thus ?thesis
  using assms
  by auto
qed

end
theory Poincare-Lines-Axis-Intersections
  imports Poincare-Between
begin

```

## 8 Intersection of h-lines with the x-axis in the Poincaré model

### 8.1 Betweenness of x-axis intersection

The intersection point of the h-line determined by points  $u$  and  $v$  and the x-axis is between  $u$  and  $v$ , then  $u$  and  $v$  are in the opposite half-planes (one must be in the upper, and the other one in the lower half-plane).

**lemma** *poincare-between-x-axis-intersection*:

```

assumes  $u \in \text{unit-disc}$  and  $v \in \text{unit-disc}$  and  $z \in \text{unit-disc}$  and  $u \neq v$ 
assumes  $u \notin \text{circline-set } x\text{-axis}$  and  $v \notin \text{circline-set } x\text{-axis}$ 
assumes  $z \in \text{circline-set } (\text{poincare-line } u\ v) \cap \text{circline-set } x\text{-axis}$ 
shows  $\text{poincare-between } u\ z\ v \longleftrightarrow \text{Arg } (\text{to-complex } u) * \text{Arg } (\text{to-complex } v) < 0$ 
proof–
have  $\forall u\ v. u \in \text{unit-disc} \wedge v \in \text{unit-disc} \wedge u \neq v \wedge$ 
 $u \notin \text{circline-set } x\text{-axis} \wedge v \notin \text{circline-set } x\text{-axis} \wedge$ 
 $z \in \text{circline-set } (\text{poincare-line } u\ v) \cap \text{circline-set } x\text{-axis} \longrightarrow$ 
 $(\text{poincare-between } u\ z\ v \longleftrightarrow \text{Arg } (\text{to-complex } u) * \text{Arg } (\text{to-complex } v) < 0)$  (is ?P z)
proof (rule wlog-real-zero)
show ?P  $0_h$ 
proof ((rule allI)+, rule impI, (erule conjE)+)
  fix  $u\ v$ 
  assume *:  $u \in \text{unit-disc}$   $v \in \text{unit-disc}$   $u \neq v$ 
 $u \notin \text{circline-set } x\text{-axis}$   $v \notin \text{circline-set } x\text{-axis}$ 
 $0_h \in \text{circline-set } (\text{poincare-line } u\ v) \cap \text{circline-set } x\text{-axis}$ 
  obtain  $u'\ v'$  where  $uv: u = \text{of-complex } u'\ v = \text{of-complex } v'$ 
  using * inf-or-of-complex[of u] inf-or-of-complex[of v]
  by auto

hence  $u \neq 0_h$   $v \neq 0_h$   $u' \neq 0$   $v' \neq 0$ 
  using *
  by auto

hence  $\text{Arg } u' \neq 0$   $\text{Arg } v' \neq 0$ 
  using * arg-0-iff[of u'] arg-0-iff[of v']
  unfolding circline-set-x-axis uv
  by auto

have poincare-collinear  $\{0_h, u, v\}$ 
  using *
  unfolding poincare-collinear-def
  by (rule-tac  $x=\text{poincare-line } u\ v$  in exI, simp)
have  $(\exists k < 0. u' = \text{cor } k * v') \longleftrightarrow (\text{Arg } u' * \text{Arg } v' < 0)$  (is ?lhs  $\longleftrightarrow$  ?rhs)
proof
  assume ?lhs
  then obtain  $k$  where  $k < 0$   $u' = \text{cor } k * v'$ 
  by auto
  thus ?rhs
  using arg-mult-real-negative[of k v'] arg-uminus-opposite-sign[of v']
  using  $\langle u' \neq 0 \rangle \langle v' \neq 0 \rangle \langle \text{Arg } u' \neq 0 \rangle \langle \text{Arg } v' \neq 0 \rangle$ 
  by (auto simp add: mult-neg-pos mult-pos-neg)
next
  assume ?rhs

```

```

obtain ru rv φ where polar: u' = cor ru * cis φ v' = cor rv * cis φ
  using ⟨poincare-collinear {0h, u, v}⟩ poincare-collinear-zero-polar-form[of u' v'] uv * ⟨u' ≠ 0⟩ ⟨v' ≠ 0⟩
  by auto
have ru * rv < 0
  using polar ⟨?rhs⟩ ⟨u' ≠ 0⟩ ⟨v' ≠ 0⟩
  using arg-mult-real-negative[of ru cis φ] arg-mult-real-positive[of ru cis φ]
  using arg-mult-real-negative[of rv cis φ] arg-mult-real-positive[of rv cis φ]
  apply (cases ru > 0)
  apply (cases rv > 0, simp, simp add: mult-pos-neg)
  apply (cases rv > 0, simp add: mult-neg-pos, simp)
  done
thus ?lhs
  using polar
  by (rule-tac x=ru / rv in exI, auto simp add: divide-less-0-iff mult-less-0-iff)
qed
thus poincare-between u 0h v = (Arg (to-complex u) * Arg (to-complex v) < 0)
  using poincare-between-u0v[of u v] * ⟨u ≠ 0h⟩ ⟨v ≠ 0h⟩ uv
  by simp
qed
next
fix a z
assume 1: is-real a cmod a < 1 z ∈ unit-disc
assume 2: ?P (moebius-pt (blaschke a) z)
show ?P z
proof ((rule allI)+, rule impI, (erule conjE)+)
  fix u v
  let ?M = moebius-pt (blaschke a)
  let ?Mu = ?M u
  let ?Mv = ?M v
  assume *: u ∈ unit-disc v ∈ unit-disc u ≠ v u ∉ circline-set x-axis v ∉ circline-set x-axis
  hence u ≠ ∞h v ≠ ∞h
  by auto

  have **:  $\bigwedge x y :: \text{real}. x * y < 0 \longleftrightarrow \text{sgn}(x * y) < 0$ 
  by simp

  assume z ∈ circline-set (poincare-line u v) ∩ circline-set x-axis
  thus poincare-between u z v = (Arg (to-complex u) * Arg (to-complex v) < 0)
    using * 1 2[rule-format, of ?Mu ?Mv] ⟨cmod a < 1⟩ ⟨is-real a⟩ blaschke-unit-disc-fix[of a]
    using inversion-noteq-unit-disc[of of-complex a u] ⟨u ≠ ∞h⟩
    using inversion-noteq-unit-disc[of of-complex a v] ⟨v ≠ ∞h⟩
    apply auto
    apply (subst (asm) **, subst **, subst (asm) sgn-mult, subst sgn-mult, simp)
    apply (subst (asm) **, subst (asm) **, subst (asm) sgn-mult, subst (asm) sgn-mult, simp)
    done
  qed
next
show z ∈ unit-disc by fact
next
show is-real (to-complex z)
  using assms inf-or-of-complex[of z]
  by (auto simp add: circline-set-x-axis)
qed
thus ?thesis
  using assms
  by simp
qed

```

## 8.2 Check if an h-line intersects the x-axis

lemma x-axis-intersection-equation:

```

assumes
  H = mk-circline A B C D and
  (A, B, C, D) ∈ hermitean-nonzero
shows of-complex z ∈ circline-set x-axis ∩ circline-set H  $\longleftrightarrow$ 
  A*z2 + 2*Re B*z + D = 0 ∧ is-real z (is ?lhs  $\longleftrightarrow$  ?rhs)

```

```

proof-
  have ?lhs  $\longleftrightarrow A*z^2 + (B + cnj B)*z + D = 0 \wedge z = cnj z$ 
    using assms
    using circline-equation-x-axis[of z]
    using circline-equation[of H A B C D z]
    using hermitean-elems
    by (auto simp add: power2-eq-square field-simps)
  thus ?thesis
    using eq-cnj-iff-real[of z]
    using hermitean-elems[of A B C D]
    by (simp add: complex-add-cnj complex-eq-if-Re-eq)
qed

```

Check if an h-line intersects x-axis within the unit disc - this could be generalized to checking if an arbitrary circline intersects the x-axis, but we do not need that.

**definition** *intersects-x-axis-cmat* :: *complex-mat*  $\Rightarrow$  *bool* **where**  
 [*simp*]: *intersects-x-axis-cmat* *H* = (*let* (*A, B, C, D*) = *H* *in*  $A = 0 \vee (Re B)^2 > (Re A)^2$ )

**lift-definition** *intersects-x-axis-clmat* :: *circline-mat*  $\Rightarrow$  *bool* **is** *intersects-x-axis-cmat*  
**done**

**lift-definition** *intersects-x-axis* :: *circline*  $\Rightarrow$  *bool* **is** *intersects-x-axis-clmat*

**proof** (*transfer*)

```

  fix H1 H2
  assume hh: hermitean H1  $\wedge$  H1  $\neq$  mat-zero and hermitean H2  $\wedge$  H2  $\neq$  mat-zero
  obtain A1 B1 C1 D1 A2 B2 C2 D2 where *: H1 = (A1, B1, C1, D1) H2 = (A2, B2, C2, D2)
    by (cases H1, cases H2, auto)
  assume circline-eq-cmat H1 H2
  then obtain k where k: k  $\neq$  0  $\wedge$  H2 = cor k *sm H1
    by auto
  show intersects-x-axis-cmat H1 = intersects-x-axis-cmat H2
  proof-
    have k  $\neq$  0  $\implies (Re A1)^2 < (Re B1)^2 \longleftrightarrow (k * Re A1)^2 < (k * Re B1)^2$ 
      by (smt mult-strict-left-mono power2-eq-square semiring-normalization-rules(13) zero-less-power2)
    thus ?thesis
      using * k
      by auto
  qed
qed

```

**lemma** *intersects-x-axis-mk-circline*:

**assumes** *is-real A* **and**  $A \neq 0 \vee B \neq 0$   
**shows** *intersects-x-axis* (*mk-circline A B (cnj B) A*)  $\longleftrightarrow A = 0 \vee (Re B)^2 > (Re A)^2$

**proof-**

```

  let ?H = (A, B, (cnj B), A)
  have hermitean ?H
    using  $\langle$ is-real A $\rangle$ 
    unfolding hermitean-def mat-adj-def mat-cnj-def
    using eq-cnj-iff-real
    by auto
  moreover
  have ?H  $\neq$  mat-zero
    using assms
    by auto
  ultimately
  show ?thesis
    by (transfer, transfer, auto simp add: Let-def)
qed

```

**lemma** *intersects-x-axis-iff*:

**assumes** *is-poincare-line H*  
**shows**  $(\exists x \in \text{unit-disc. } x \in \text{circline-set } H \cap \text{circline-set } x\text{-axis}) \longleftrightarrow \text{intersects-x-axis } H$

**proof-**

```

  obtain Ac B C Dc where *: H = mk-circline Ac B C Dc hermitean (Ac, B, C, Dc) (Ac, B, C, Dc) \neq mat-zero
    using ex-mk-circline[of H]
    by auto

```

```

hence (cmod B)2 > (cmod Ac)2 Ac = Dc
using assms
using is-poincare-line-mk-circline
by auto

hence H = mk-circline (Re Ac) B (cnj B) (Re Ac) hermitean (cor (Re Ac), B, (cnj B), cor (Re Ac)) (cor (Re Ac),
B, (cnj B), cor (Re Ac)) ≠ mat-zero
using hermitean-elems[of Ac B C Dc] *
by auto
then obtain A where
*: H = mk-circline (cor A) B (cnj B) (cor A) (cor A, B, (cnj B), cor A) ∈ hermitean-nonzero
by auto

show ?thesis
proof (cases A = 0)
case True
thus ?thesis
using *
using x-axis-intersection-equation[OF *(1-2), of 0]
using intersects-x-axis-mk-circline[of cor A B]
by auto
next
case False
show ?thesis
proof
assume ∃ x ∈ unit-disc. x ∈ circline-set H ∩ circline-set x-axis
then obtain x where **: of-complex x ∈ unit-disc of-complex x ∈ circline-set H ∩ circline-set x-axis
by (metis inf-or-of-complex inf-notin-unit-disc)
hence is-real x
unfolding circline-set-x-axis
using of-complex-inj
by auto
hence eq: A * (Re x)2 + 2 * Re B * Re x + A = 0
using **
using x-axis-intersection-equation[OF *(1-2), of Re x]
by simp
hence (2 * Re B)2 - 4 * A * A ≥ 0
using discriminant-iff[of A - 2 * Re B A]
using discrim-def[of A 2 * Re B A] False
by auto
hence (Re B)2 ≥ (Re A)2
by (simp add: power2-eq-square)
moreover
have (Re B)2 ≠ (Re A)2
proof (rule ccontr)
assume ¬ ?thesis
hence Re B = Re A ∨ Re B = - Re A
using power2-eq-iff by blast
hence A * (Re x)2 + A * 2 * Re x + A = 0 ∨ A * (Re x)2 - A * 2 * Re x + A = 0
using eq
by auto
hence A * ((Re x)2 + 2 * Re x + 1) = 0 ∨ A * ((Re x)2 - 2 * Re x + 1) = 0
by (simp add: field-simps)
hence (Re x)2 + 2 * Re x + 1 = 0 ∨ (Re x)2 - 2 * Re x + 1 = 0
using ⟨A ≠ 0⟩
by simp
hence (Re x + 1)2 = 0 ∨ (Re x - 1)2 = 0
by (simp add: power2-sum power2-diff field-simps)
hence Re x = -1 ∨ Re x = 1
by auto
thus False
using ⟨is-real x⟩ ⟨of-complex x ∈ unit-disc⟩
by (auto simp add: cmod-eq-Re)
qed
ultimately
show intersects-x-axis H

```

```

    using intersects-x-axis-mk-circline
    using *
    by auto
next
assume intersects-x-axis H
hence  $(\text{Re } B)^2 > (\text{Re } A)^2$ 
    using * False
    using intersects-x-axis-mk-circline
    by simp
hence discr:  $(2 * \text{Re } B)^2 - 4 * A * A > 0$ 
    by (simp add: power2-eq-square)
then obtain x1 x2 where
    eqs:  $A * x1^2 + 2 * \text{Re } B * x1 + A = 0$   $A * x2^2 + 2 * \text{Re } B * x2 + A = 0$   $x1 \neq x2$ 
    using discriminant-pos-ex[OF  $\langle A \neq 0 \rangle$ , of  $2 * \text{Re } B$  A]
    using discrim-def[of  $A$   $2 * \text{Re } B$  A]
    by auto
hence  $x1 * x2 = 1$ 
    using viette2[OF  $\langle A \neq 0 \rangle$ , of  $2 * \text{Re } B$  A  $x1$   $x2$ ] discr  $\langle A \neq 0 \rangle$ 
    by auto
have  $\text{abs } x1 \neq 1$   $\text{abs } x2 \neq 1$ 
    using eqs discr  $\langle x1 * x2 = 1 \rangle$ 
    by (auto simp add: abs-if power2-eq-square)
hence  $\text{abs } x1 < 1 \vee \text{abs } x2 < 1$ 
    using  $\langle x1 * x2 = 1 \rangle$ 
    by (smt mult-le-cancel-left1 mult-minus-right)
thus  $\exists x \in \text{unit-disc. } x \in \text{circline-set } H \cap \text{circline-set } x\text{-axis}$ 
    using x-axis-intersection-equation[OF  $*(1-2)$ , of  $x1$ ]
    using x-axis-intersection-equation[OF  $*(1-2)$ , of  $x2$ ]
    using eqs
    by auto
qed
qed
qed

```

### 8.3 Check if a Poincaré line intersects the y-axis

**definition** *intersects-y-axis-cmat* :: *complex-mat*  $\Rightarrow$  *bool* **where**  
*[simp]*: *intersects-y-axis-cmat*  $H = (\text{let } (A, B, C, D) = H \text{ in } A = 0 \vee (\text{Im } B)^2 > (\text{Re } A)^2)$

**lift-definition** *intersects-y-axis-clmat* :: *circline-mat*  $\Rightarrow$  *bool* **is** *intersects-y-axis-cmat*  
**done**

**lift-definition** *intersects-y-axis* :: *circline*  $\Rightarrow$  *bool* **is** *intersects-y-axis-clmat*

**proof** (*transfer*)

```

fix H1 H2
assume hh: hermitean H1  $\wedge$  H1  $\neq$  mat-zero and hermitean H2  $\wedge$  H2  $\neq$  mat-zero
obtain A1 B1 C1 D1 A2 B2 C2 D2 where *: H1 = (A1, B1, C1, D1) H2 = (A2, B2, C2, D2)
    by (cases H1, cases H2, auto)
assume circline-eq-cmat H1 H2
then obtain k where k:  $k \neq 0 \wedge H2 = \text{cor } k *_{sm} H1$ 
    by auto
show intersects-y-axis-cmat H1 = intersects-y-axis-cmat H2
proof–
    have  $k \neq 0 \implies (\text{Re } A1)^2 < (\text{Im } B1)^2 \iff (k * \text{Re } A1)^2 < (k * \text{Im } B1)^2$ 
        by (smt mult-strict-left-mono power2-eq-square semiring-normalization-rules(13) zero-less-power2)
    thus ?thesis
        using * k
        by auto
    qed
qed

```

**lemma** *intersects-x-axis-intersects-y-axis* [*simp*]:

**shows** *intersects-x-axis* (*moebius-circline* (*moebius-rotation* ( $\pi/2$ )) *H*)  $\iff$  *intersects-y-axis* *H*  
**unfolding** *moebius-rotation-def* *moebius-similarity-def*  
**by** *simp* (*transfer, transfer, auto simp add: mat-adj-def mat-cnj-def*)

```

lemma intersects-y-axis-iff:
  assumes is-poincare-line H
  shows  $(\exists y \in \text{unit-disc. } y \in \text{circline-set } H \cap \text{circline-set } y\text{-axis}) \longleftrightarrow \text{intersects-y-axis } H$  (is ?lhs  $\longleftrightarrow$  ?rhs)
proof-
  let ?R = moebius-rotation (pi / 2)
  let ?H' = moebius-circline ?R H
  have 1: is-poincare-line ?H'
    using assms
    using unit-circle-fix-preserve-is-poincare-line[OF - assms, of ?R]
    by simp

  show ?thesis
proof
  assume ?lhs
  then obtain y where  $y \in \text{unit-disc } y \in \text{circline-set } H \cap \text{circline-set } y\text{-axis}$ 
    by auto
  hence moebius-pt ?R y  $\in \text{unit-disc} \wedge \text{moebius-pt } ?R y \in \text{circline-set } ?H' \cap \text{circline-set } x\text{-axis}$ 
    using rotation-pi-2-y-axis
  by (metis Int-iff circline-set-moebius-circline-E moebius-circline-comp-inv-left moebius-pt-comp-inv-left unit-disc-fix-discI
unit-disc-fix-rotation)
  thus ?rhs
    using intersects-x-axis-iff[OF 1]
    using intersects-x-axis-intersects-y-axis[of H]
    by auto
next
  assume intersects-y-axis H
  hence intersects-x-axis ?H'
    using intersects-x-axis-intersects-y-axis[of H]
    by simp
  then obtain x where *:  $x \in \text{unit-disc } x \in \text{circline-set } ?H' \cap \text{circline-set } x\text{-axis}$ 
    using intersects-x-axis-iff[OF 1]
    by auto
  let ?y = moebius-pt (-?R) x
  have ?y  $\in \text{unit-disc} \wedge ?y \in \text{circline-set } H \cap \text{circline-set } y\text{-axis}$ 
    using * rotation-pi-2-y-axis[symmetric]
  by (metis Int-iff circline-set-moebius-circline-E moebius-pt-comp-inv-left moebius-rotation-uminus uminus-moebius-def
unit-disc-fix-discI unit-disc-fix-rotation)
  thus ?lhs
    by auto
qed
qed

```

## 8.4 Intersection point of a Poincaré line with the x-axis in the unit disc

**definition** *calc-x-axis-intersection-cvec* :: *complex*  $\Rightarrow$  *complex*  $\Rightarrow$  *complex-vec* **where**

```

[simp]: calc-x-axis-intersection-cvec A B =
  (let discr = (Re B)2 - (Re A)2 in
    (-Re(B) + sgn (Re B) * sqrt(discr), A))

```

**definition** *calc-x-axis-intersection-cmat-cvec* :: *complex-mat*  $\Rightarrow$  *complex-vec* **where** [simp]:

```

calc-x-axis-intersection-cmat-cvec H =
  (let (A, B, C, D) = H in
    if A  $\neq$  0 then
      calc-x-axis-intersection-cvec A B
    else
      (0, 1)
  )

```

**lift-definition** *calc-x-axis-intersection-clmat-hcoords* :: *circline-mat*  $\Rightarrow$  *complex-homo-coords* **is** *calc-x-axis-intersection-cmat-cvec*  
**by** (auto split: if-split-asm)

**lift-definition** *calc-x-axis-intersection* :: *circline*  $\Rightarrow$  *complex-homo* **is** *calc-x-axis-intersection-clmat-hcoords*

**proof** transfer

```
fix H1 H2
```

```
assume *: hermitean H1  $\wedge$  H1  $\neq$  mat-zero hermitean H2  $\wedge$  H2  $\neq$  mat-zero
```

```

obtain  $A1\ B1\ C1\ D1\ A2\ B2\ C2\ D2$  where  $hh: H1 = (A1, B1, C1, D1)\ H2 = (A2, B2, C2, D2)$ 
  by (cases H1, cases H2, auto)
assume circline-eq-cmat H1 H2
then obtain  $k$  where  $k \neq 0\ H2 = cor\ k\ *_{sm}\ H1$ 
  by auto

```

```

have calc-x-axis-intersection-cvec A1 B1  $\approx_v$  calc-x-axis-intersection-cvec A2 B2

```

```

  using  $hh\ k$ 
  apply simp
  apply (rule-tac x=cor k in exI)
  apply auto
  apply (simp add: sgn-mult power-mult-distrib)
  apply (subst right-diff-distrib[symmetric])
  apply (subst real-sqrt-mult)
  by (simp add: real-sgn-eq right-diff-distrib)

```

```

thus calc-x-axis-intersection-cmat-cvec H1  $\approx_v$ 
  calc-x-axis-intersection-cmat-cvec H2

```

```

  using  $hh\ k$ 
  by (auto simp del: calc-x-axis-intersection-cvec-def)

```

qed

**lemma** *calc-x-axis-intersection-in-unit-disc:*

**assumes** *is-poincare-line H intersects-x-axis H*

**shows** *calc-x-axis-intersection H  $\in$  unit-disc*

**proof** (*cases is-line H*)

**case** *True*

**thus** *?thesis*

**using** *assms*

**unfolding** *unit-disc-def disc-def*

**by** *simp (transfer, transfer, auto simp add: vec-cnj-def)*

**next**

**case** *False*

**thus** *?thesis*

**using** *assms*

**unfolding** *unit-disc-def disc-def*

**proof** (*simp, transfer, transfer*)

**fix**  $H$

**assume**  $hh: hermitean\ H \wedge H \neq mat-zero$

**then obtain**  $A\ B\ D$  **where**  $*$ :  $H = (A, B, cnj\ B, D)$  *is-real A is-real D*

**using** *hermitean-elems*

**by** (*cases H*) *blast*

**assume** *is-poincare-line-cmat H*

**hence**  $*$ :  $H = (A, B, cnj\ B, A)$  *is-real A*

**using**  $*$

**by** *auto*

**assume**  $\neg$  *circline-A0-cmat H*

**hence**  $A \neq 0$

**using**  $*$

**by** *simp*

**assume** *intersects-x-axis-cmat H*

**hence**  $(Re\ B)^2 > (Re\ A)^2$

**using**  $*$   $\langle A \neq 0 \rangle$

**by** (*auto simp add: power2-eq-square complex.expand*)

**hence**  $Re\ B \neq 0$

**by** *auto*

**have**  $Re\ A \neq 0$

**using**  $\langle is-real\ A \rangle \langle A \neq 0 \rangle$

**by** (*auto simp add: complex.expand*)

**have**  $\sqrt{(Re\ B)^2 - (Re\ A)^2} < \sqrt{(Re\ B)^2}$

```

    using ⟨Re A ≠ 0⟩
    by (subst real-sqrt-less-iff) auto
  also have ... = sgn (Re B) * (Re B)
    by (smt mult-minus-right nonzero-eq-divide-eq real-sgn-eq real-sqrt-abs)
  finally
  have 1: sqrt((Re B)2 - (Re A)2) < sgn (Re B) * (Re B)
    .

  have 2: (Re B)2 - (Re A)2 < sgn (Re B) * (Re B) * sqrt((Re B)2 - (Re A)2)
    using ⟨(Re B)2 > (Re A)2⟩
    using mult-strict-right-mono[OF 1, of sqrt ((Re B)2 - (Re A)2)]
    by simp

  have 3: (Re B)2 - 2*sgn (Re B)*Re B*sqrt((Re B)2 - (Re A)2) + (Re B)2 - (Re A)2 < (Re A)2
    using mult-strict-left-mono[OF 2, of 2]
    by (simp add: field-simps)

  have (sgn (Re B))2 = 1
    using ⟨Re B ≠ 0⟩
    by (simp add: sgn-if)

  hence (-Re B + sgn (Re B) * sqrt((Re B)2 - (Re A)2))2 < (Re A)2
    using ⟨(Re B)2 > (Re A)2⟩ 3
    by (simp add: power2-diff field-simps)

  thus in-ocircline-cmat-cvec unit-circle-cmat (calc-x-axis-intersection-cmat-cvec H)
    using * ⟨(Re B)2 > (Re A)2⟩
    by (auto simp add: vec-cnj-def power2-eq-square split: if-split-asm)
qed
qed

```

**lemma** *calc-x-axis-intersection:*

```

  assumes is-poincare-line H and intersects-x-axis H
  shows calc-x-axis-intersection H ∈ circline-set H ∩ circline-set x-axis
proof (cases is-line H)
  case True
  thus ?thesis
    using assms
    unfolding circline-set-def
    by (simp (transfer, transfer, auto simp add: vec-cnj-def))
next
  case False
  thus ?thesis
    using assms
    unfolding circline-set-def
proof (simp, transfer, transfer)
  fix H
  assume hh: hermitean H ∧ H ≠ mat-zero
  then obtain A B D where *: H = (A, B, cnj B, D) is-real A is-real D
    using hermitean-elems
    by (cases H) blast
  assume is-poincare-line-cmat H
  hence *: H = (A, B, cnj B, A) is-real A
    using *
    by auto
  assume ¬ circline-A0-cmat H
  hence A ≠ 0
    using *
    by auto

  assume intersects-x-axis-cmat H
  hence (Re B)2 > (Re A)2
    using * ⟨A ≠ 0⟩
    by (auto simp add: power2-eq-square complex.expand)

```



hence  $\text{Re } B \neq 0$

by *auto*

show *on-circline-cmat-cvec H (calc-x-axis-intersection-cmat-cvec H)  $\wedge$*   
*on-circline-cmat-cvec x-axis-cmat (calc-x-axis-intersection-cmat-cvec H) (is ?P1  $\wedge$  ?P2)*

**proof**

show *on-circline-cmat-cvec H (calc-x-axis-intersection-cmat-cvec H)*

**proof** (*cases circline-A0-cmat H*)

case *True*

thus *?thesis*

using \* *⟨is-poincare-line-cmat H⟩ ⟨intersects-x-axis-cmat H⟩*

by (*simp add: vec-cnj-def*)

next

case *False*

let *?x = calc-x-axis-intersection-cvec A B*

let *?nom = fst ?x and ?den = snd ?x*

have *x: ?x = (?nom, ?den)*

by *simp*

hence *on-circline-cmat-cvec H (calc-x-axis-intersection-cvec A B)*

**proof** (*subst \*, subst x, subst on-circline-cmat-cvec-circline-equation*)

have  $(\text{sgn}(\text{Re } B))^2 = 1$

using *⟨Re B  $\neq 0$ ⟩ sgn-pos zero-less-power2* by *fastforce*

have  $(\text{sqrt}((\text{Re } B)^2 - (\text{Re } A)^2))^2 = (\text{Re } B)^2 - (\text{Re } A)^2$

using *⟨(Re B)<sup>2</sup> > (Re A)<sup>2</sup>⟩*

by *simp*

have  $(-(\text{Re } B) + \text{sgn}(\text{Re } B) * \text{sqrt}((\text{Re } B)^2 - (\text{Re } A)^2))^2 =$   
 $(-(\text{Re } B))^2 + (\text{sgn}(\text{Re } B) * \text{sqrt}((\text{Re } B)^2 - (\text{Re } A)^2))^2 - 2 * (\text{Re } B) * \text{sgn}(\text{Re } B) * \text{sqrt}((\text{Re } B)^2 - (\text{Re } A)^2)$

by (*simp add: power2-diff*)

also have  $\dots = (\text{Re } B) * (\text{Re } B) + (\text{sgn}(\text{Re } B) * \text{sqrt}((\text{Re } B)^2 - (\text{Re } A)^2))^2 - 2 * (\text{Re } B) * \text{sgn}(\text{Re } B) * \text{sqrt}((\text{Re } B)^2 - (\text{Re } A)^2)$

by (*simp add: power2-eq-square*)

also have  $\dots = (\text{Re } B) * (\text{Re } B) + (\text{sgn}(\text{Re } B))^2 * (\text{sqrt}((\text{Re } B)^2 - (\text{Re } A)^2))^2 - 2 * (\text{Re } B) * \text{sgn}(\text{Re } B) * \text{sqrt}((\text{Re } B)^2 - (\text{Re } A)^2)$

by (*simp add: power-mult-distrib*)

also have  $\dots = (\text{Re } B) * (\text{Re } B) + (\text{Re } B)^2 - (\text{Re } A)^2 - 2 * (\text{Re } B) * \text{sgn}(\text{Re } B) * \text{sqrt}((\text{Re } B)^2 - (\text{Re } A)^2)$

using *⟨(sqrt((Re B)<sup>2</sup> - (Re A)<sup>2</sup>))<sup>2</sup> = (Re B)<sup>2</sup> - (Re A)<sup>2</sup>⟩, ⟨(sgn(Re B))<sup>2</sup> = 1⟩*

by *simp*

finally have  $(-(\text{Re } B) + \text{sgn}(\text{Re } B) * \text{sqrt}((\text{Re } B)^2 - (\text{Re } A)^2))^2 =$   
 $(\text{Re } B) * (\text{Re } B) + (\text{Re } B)^2 - (\text{Re } A)^2 - 2 * (\text{Re } B) * \text{sgn}(\text{Re } B) * \text{sqrt}((\text{Re } B)^2 - (\text{Re } A)^2)$

by *simp*

have *is-real ?nom is-real ?den*

using *⟨is-real A⟩*

by *simp+*

hence *cnj (?nom) = ?nom cnj (?den) = ?den*

by (*simp add: eq-cnj-iff-real*) +

hence  $A * ?nom * (\text{cnj } (?nom)) + B * ?den * (\text{cnj } (?nom)) + (\text{cnj } B) * (\text{cnj } (?den)) * ?nom + A * ?den * (\text{cnj } (?den))$   
 $= A * ?nom * ?nom + B * ?den * ?nom + (\text{cnj } B) * ?den * ?nom + A * ?den * ?den$

by *auto*

also have  $\dots = A * ?nom * ?nom + (B + (\text{cnj } B)) * ?den * ?nom + A * ?den * ?den$

by (*simp add: field-simps*)

also have  $\dots = A * ?nom * ?nom + 2 * (\text{Re } B) * ?den * ?nom + A * ?den * ?den$

by (*simp add: complex-add-cnj*)

also have  $\dots = A * ?nom^2 + 2 * (\text{Re } B) * ?den * ?nom + A * ?den * ?den$

by (*simp add: power2-eq-square*)

also have  $\dots = A * (-(\text{Re } B) + \text{sgn}(\text{Re } B) * \text{sqrt}((\text{Re } B)^2 - (\text{Re } A)^2))^2$   
 $+ 2 * (\text{Re } B) * A * (-(\text{Re } B) + \text{sgn}(\text{Re } B) * \text{sqrt}((\text{Re } B)^2 - (\text{Re } A)^2)) + A * A * A$

unfolding *calc-x-axis-intersection-cvec-def*

by *auto*

also have  $\dots = A * ((\text{Re } B) * (\text{Re } B) + (\text{Re } B)^2 - (\text{Re } A)^2 - 2 * (\text{Re } B) * \text{sgn}(\text{Re } B) * \text{sqrt}((\text{Re } B)^2 - (\text{Re } A)^2))$   
 $+ 2 * (\text{Re } B) * A * (-(\text{Re } B) + \text{sgn}(\text{Re } B) * \text{sqrt}((\text{Re } B)^2 - (\text{Re } A)^2)) + A * A * A$

using *⟨(-(Re B) + sgn(Re B) \* sqrt((Re B)<sup>2</sup> - (Re A)<sup>2</sup>))<sup>2</sup> =*

$(\text{Re } B) * (\text{Re } B) + (\text{Re } B)^2 - (\text{Re } A)^2 - 2 * (\text{Re } B) * \text{sgn}(\text{Re } B) * \text{sqrt}((\text{Re } B)^2 - (\text{Re } A)^2)⟩$

by *simp*

```

also have ... = A*((Re B)*(Re B) + (Re B)2 - A2 - 2*(Re B)*sgn(Re B)*sqrt((Re B)2 - (Re A)2))
  + 2*(Re B)*A*(-(Re B) + sgn(Re B)*sqrt((Re B)2 - (Re A)2)) + A*A*A
  using ⟨is-real A⟩
  by simp
also have ... = 0
  apply (simp add:field-simps)
  by (simp add: power2-eq-square)
  finally have A*?nom*(cnj (?nom)) + B*?den*(cnj (?nom)) + (cnj B)*(cnj (?den))*?nom + A*?den*(cnj
(?den)) = 0
  by simp
  thus circline-equation A B (cnj B) A ?nom ?den
  by simp
qed
thus ?thesis
  using * ⟨is-poincare-line-cmat H⟩ ⟨intersects-x-axis-cmat H⟩
  by (simp add: vec-cnj-def)
qed
next
show on-circline-cmat-cvec x-axis-cmat (calc-x-axis-intersection-cmat-cvec H)
  using * ⟨is-poincare-line-cmat H⟩ ⟨intersects-x-axis-cmat H⟩ ⟨is-real A⟩
  using eq-cnj-iff-real[of A]
  by (simp add: vec-cnj-def)
qed
qed
qed

```

**lemma** *unique-calc-x-axis-intersection:*

**assumes** *is-poincare-line H and H ≠ x-axis*

**assumes** *x ∈ unit-disc and x ∈ circline-set H ∩ circline-set x-axis*

**shows** *x = calc-x-axis-intersection H*

**proof**—

**have** \*: *intersects-x-axis H*

**using** *assms*

**using** *intersects-x-axis-iff[OF assms(1)]*

**by auto**

**show** *x = calc-x-axis-intersection H*

**using** *calc-x-axis-intersection[OF assms(1) \*]*

**using** *calc-x-axis-intersection-in-unit-disc[OF assms(1) \*]*

**using** *assms*

**using** *unique-is-poincare-line[of x calc-x-axis-intersection H H x-axis]*

**by auto**

**qed**

## 8.5 Check if an h-line intersects the positive part of the x-axis

**definition** *intersects-x-axis-positive-cmat :: complex-mat ⇒ bool where*

[simp]: *intersects-x-axis-positive-cmat H = (let (A, B, C, D) = H in Re A ≠ 0 ∧ Re B / Re A < -1)*

**lift-definition** *intersects-x-axis-positive-clmat :: circline-mat ⇒ bool is intersects-x-axis-positive-cmat*

**done**

**lift-definition** *intersects-x-axis-positive :: circline ⇒ bool is intersects-x-axis-positive-clmat*

**proof** (*transfer*)

**fix** *H1 H2*

**assume** *hh: hermitean H1 ∧ H1 ≠ mat-zero and hermitean H2 ∧ H2 ≠ mat-zero*

**obtain** *A1 B1 C1 D1 A2 B2 C2 D2 where \*: H1 = (A1, B1, C1, D1) H2 = (A2, B2, C2, D2)*

**by** (*cases H1, cases H2, auto*)

**assume** *circline-eq-cmat H1 H2*

**then obtain** *k where k ≠ 0 ∧ H2 = cor k \*<sub>sm</sub> H1*

**by auto**

**thus** *intersects-x-axis-positive-cmat H1 = intersects-x-axis-positive-cmat H2*

**using** \*

**by simp**

**qed**

**lemma** *intersects-x-axis-positive-mk-circline:*

**assumes** *is-real A and  $A \neq 0 \vee B \neq 0$*   
**shows** *intersects-x-axis-positive (mk-circline A B (cnj B) A)  $\longleftrightarrow$  Re B / Re A  $< -1$*   
**proof** –  
**let**  $?H = (A, B, (cnj B), A)$   
**have** *hermitean ?H*  
**using**  $\langle is-real A \rangle$   
**unfolding** *hermitean-def mat-adj-def mat-cnj-def*  
**using** *eq-cnj-iff-real*  
**by** *auto*  
**moreover**  
**have**  $?H \neq mat-zero$   
**using** *assms*  
**by** *auto*  
**ultimately**  
**show** *?thesis*  
**by** (*transfer, transfer, auto simp add: Let-def*)  
**qed**

**lemma** *intersects-x-axis-positive-intersects-x-axis [simp]:*  
**assumes** *intersects-x-axis-positive H*  
**shows** *intersects-x-axis H*  
**proof** –  
**have**  $\bigwedge a aa. [\text{Re } a \neq 0; \text{Re } aa / \text{Re } a < -1; \neg (\text{Re } a)^2 < (\text{Re } aa)^2] \implies aa = 0 \wedge a = 0$   
**by** (*smt less-divide-eq-1-pos one-less-power pos2 power2-minus power-divide zero-less-power2*)  
**thus** *?thesis*  
**using** *assms*  
**apply** *transfer*  
**apply** *transfer*  
**apply** (*auto simp add: hermitean-def mat-adj-def mat-cnj-def*)  
**done**  
**qed**

**lemma** *add-less-abs-positive-iff:*  
**fixes**  $a b :: real$   
**assumes**  $abs\ b < abs\ a$   
**shows**  $a + b > 0 \longleftrightarrow a > 0$   
**using** *assms*  
**by** *auto*

**lemma** *calc-x-axis-intersection-positive-abs':*  
**fixes**  $A B :: real$   
**assumes**  $B^2 > A^2$  **and**  $A \neq 0$   
**shows**  $abs\ (sgn(B) * sqrt(B^2 - A^2) / A) < abs(-B/A)$   
**proof** –  
**from** *assms* **have**  $B \neq 0$   
**by** *auto*  
  
**have**  $B^2 - A^2 < B^2$   
**using**  $\langle A \neq 0 \rangle$   
**by** *auto*  
**hence**  $sqrt\ (B^2 - A^2) < abs\ B$   
**using** *real-sqrt-less-iff[ $of\ B^2 - A^2\ B^2$ ]*  
**by** *simp*  
**thus** *?thesis*  
**using** *assms  $\langle B \neq 0 \rangle$*   
**by** (*simp add: abs-mult divide-strict-right-mono*)  
**qed**

**lemma** *calc-intersect-x-axis-positive-lemma:*  
**assumes**  $B^2 > A^2$  **and**  $A \neq 0$   
**shows**  $(-B + sgn\ B * sqrt(B^2 - A^2)) / A > 0 \longleftrightarrow -B/A > 1$   
**proof** –  
**have**  $(-B + sgn\ B * sqrt(B^2 - A^2)) / A = -B / A + (sgn\ B * sqrt(B^2 - A^2)) / A$   
**using** *assms*  
**by** (*simp add: field-simps*)

**moreover**  
**have**  $-B / A + (\text{sgn } B * \text{sqrt}(B^2 - A^2)) / A > 0 \iff -B / A > 0$   
**using** *add-less-abs-positive-iff*[*OF calc-x-axis-intersection-positive-abs'*[*OF assms*]]  
**by** *simp*  
**moreover**  
**hence**  $(B/A)^2 > 1$   
**using** *assms*  
**by** (*simp add: power-divide*)  
**hence**  $B/A > 1 \vee B/A < -1$   
**by** (*smt one-power2 pos2 power2-minus power-0 power-strict-decreasing zero-power2*)  
**hence**  $-B / A > 0 \iff -B / A > 1$   
**by** *auto*  
**ultimately**  
**show** *?thesis*  
**using** *assms*  
**by** *auto*  
**qed**

**lemma** *intersects-x-axis-positive-iff'*:  
**assumes** *is-poincare-line H*  
**shows** *intersects-x-axis-positive H*  $\iff$   
 $\text{calc-x-axis-intersection } H \in \text{unit-disc} \wedge \text{calc-x-axis-intersection } H \in \text{circline-set } H \cap \text{positive-x-axis}$  (**is** *?lhs*  $\iff$   
*?rhs*)

**proof**

**let** *?x = calc-x-axis-intersection H*  
**assume** *?lhs*  
**hence**  $?x \in \text{circline-set } x\text{-axis } ?x \in \text{circline-set } H ?x \in \text{unit-disc}$   
**using** *calc-x-axis-intersection-in-unit-disc*[*OF assms*] *calc-x-axis-intersection*[*OF assms*]  
**by** *auto*  
**moreover**  
**have**  $\text{Re } (\text{to-complex } ?x) > 0$   
**using**  $\langle ?lhs \rangle$  *assms*  
**proof** (*transfer, transfer*)  
**fix** *H*  
**assume** *hh: hermitean H*  $\wedge$   $H \neq \text{mat-zero}$   
**obtain** *A B C D* **where**  $*$ :  $H = (A, B, C, D)$   
**by** (*cases H, auto*)  
**assume** *intersects-x-axis-positive-cmat H*  
**hence**  $*$ :  $\text{Re } B / \text{Re } A < -1$   $\text{Re } A \neq 0$   
**using**  $*$   
**by** *auto*  
**have**  $(\text{Re } B)^2 > (\text{Re } A)^2$   
**using**  $*$   
**by** (*smt divide-less-eq-1-neg divide-minus-left less-divide-eq-1-pos real-sqrt-abs real-sqrt-less-iff right-inverse-eq*)  
**have** *is-real A*  $A \neq 0$   
**using** *hh hermitean-elems*  $*$   $\langle \text{Re } A \neq 0 \rangle$  *complex.expand*[*of A 0*]  
**by** *auto*  
**have**  $(\text{cmod } B)^2 > (\text{cmod } A)^2$   
**using**  $\langle (\text{Re } B)^2 > (\text{Re } A)^2 \rangle$   $\langle \text{is-real } A \rangle$   
**by** (*smt cmod-power2 power2-less-0 zero-power2*)  
**have**  $*$ :  $0 < (-\text{Re } B + \text{sgn } (\text{Re } B) * \text{sqrt}((\text{Re } B)^2 - (\text{Re } A)^2)) / \text{Re } A$   
**using** *calc-intersect-x-axis-positive-lemma*[*of Re A Re B*]  $*$   $\langle (\text{Re } B)^2 > (\text{Re } A)^2 \rangle$   
**by** *auto*  
  
**assume** *is-poincare-line-cmat H*  
**hence**  $A = D$   
**using**  $*$  *hh*  
**by** *simp*

**have**  $\text{Re } ((\text{cor } (\text{sgn } (\text{Re } B)) * \text{cor } (\text{sqrt}((\text{Re } B)^2 - (\text{Re } A)^2)) - \text{cor } (\text{Re } B)) / A) = (\text{sgn } (\text{Re } B) * \text{sqrt}((\text{Re } B)^2 - (\text{Re } A)^2) - \text{Re } B) / \text{Re } D$   
**using**  $\langle \text{is-real } A \rangle$   $\langle A = D \rangle$   
**by** (*metis (no-types, lifting) Re-complex-of-real complex-of-real-Re of-real-diff of-real-divide of-real-mult*)  
**thus**  $0 < \text{Re } (\text{to-complex-cvec } (\text{calc-x-axis-intersection-cmat-cvec } H))$   
**using**  $*$  *hh*  $*$   $*$   $\langle (\text{cmod } B)^2 > (\text{cmod } A)^2 \rangle$   $\langle (\text{Re } B)^2 > (\text{Re } A)^2 \rangle$   $\langle A \neq 0 \rangle$   $\langle A = D \rangle$   
**by** *simp*

```

qed
ultimately
show ?rhs
  unfolding positive-x-axis-def
  by auto
next
let ?x = calc-x-axis-intersection H
assume ?rhs
hence Re (to-complex ?x) > 0 ?x ≠ ∞h ?x ∈ circline-set x-axis ?x ∈ unit-disc ?x ∈ circline-set H
  unfolding positive-x-axis-def
  by auto
hence intersects-x-axis H
  using intersects-x-axis-iff[OF assms]
  by auto
thus ?lhs
  using ⟨Re (to-complex ?x) > 0⟩ assms
proof (transfer, transfer)
  fix H
  assume hh: hermitean H ∧ H ≠ mat-zero
  obtain A B C D where *: H = (A, B, C, D)
  by (cases H, auto)
  assume 0 < Re (to-complex-cvec (calc-x-axis-intersection-cmat-cvec H)) intersects-x-axis-cmat H is-poincare-line-cmat
H
  hence **: A ≠ 0 0 < Re ((cor (sgn (Re B)) * cor (sqrt ((Re B)2 - (Re A)2)) - cor (Re B)) / A) A = D is-real
A (Re B)2 > (Re A)2
  using * hh hermitean-elems
  by (auto split: if-split-asm)

  have Re A ≠ 0
  using complex.expand[of A 0] ⟨A ≠ 0⟩ ⟨is-real A⟩
  by auto

  have Re ((cor (sgn (Re B)) * cor (sqrt ((Re B)2 - (Re D)2)) - cor (Re B)) / D) = (sgn (Re B) * sqrt ((Re B)2
- (Re D)2) - Re B) / Re D
  using ⟨is-real A⟩ ⟨A = D⟩
  by (metis (no-types, lifting) Re-complex-of-real complex-of-real-Re of-real-diff of-real-divide of-real-mult)

  thus intersects-x-axis-positive-cmat H
  using * ** ⟨Re A ≠ 0⟩
  using calc-intersect-x-axis-positive-lemma[of Re A Re B]
  by simp
qed
qed

lemma intersects-x-axis-positive-iff:
  assumes is-poincare-line H and H ≠ x-axis
  shows intersects-x-axis-positive H ⟷
    (∃ x. x ∈ unit-disc ∧ x ∈ circline-set H ∩ positive-x-axis) (is ?lhs ⟷ ?rhs)
proof
  assume ?lhs
  thus ?rhs
  using intersects-x-axis-positive-iff'[OF assms(1)]
  by auto
next
  assume ?rhs
  then obtain x where x ∈ unit-disc x ∈ circline-set H ∩ positive-x-axis
  by auto
  thus ?lhs
  using unique-calc-x-axis-intersection[OF assms, of x]
  using intersects-x-axis-positive-iff'[OF assms(1)]
  unfolding positive-x-axis-def
  by auto
qed

```

## 8.6 Check if an h-line intersects the positive part of the y-axis

**definition** *intersects-y-axis-positive-cmat* :: *complex-mat*  $\Rightarrow$  *bool* **where**  
*[simp]*: *intersects-y-axis-positive-cmat*  $H = (\text{let } (A, B, C, D) = H \text{ in } \text{Re } A \neq 0 \wedge \text{Im } B / \text{Re } A < -1)$

**lift-definition** *intersects-y-axis-positive-clmat* :: *circline-mat*  $\Rightarrow$  *bool* **is** *intersects-y-axis-positive-cmat* **done**

**lift-definition** *intersects-y-axis-positive* :: *circline*  $\Rightarrow$  *bool* **is** *intersects-y-axis-positive-clmat*

**proof** (*transfer*)

**fix**  $H1\ H2$

**assume**  $hh: \text{hermitean } H1 \wedge H1 \neq \text{mat-zero}$  **and**  $\text{hermitean } H2 \wedge H2 \neq \text{mat-zero}$

**obtain**  $A1\ B1\ C1\ D1\ A2\ B2\ C2\ D2$  **where**  $*$ :  $H1 = (A1, B1, C1, D1)\ H2 = (A2, B2, C2, D2)$

**by** (*cases*  $H1$ , *cases*  $H2$ , *auto*)

**assume** *circline-eq-cmat*  $H1\ H2$

**then obtain**  $k$  **where**  $k \neq 0 \wedge H2 = \text{cor } k *_{sm} H1$

**by** *auto*

**thus** *intersects-y-axis-positive-cmat*  $H1 = \text{intersects-y-axis-positive-cmat } H2$

**using**  $*$

**by** *simp*

**qed**

**lemma** *intersects-x-axis-positive-intersects-y-axis-positive* [*simp*]:

**shows** *intersects-x-axis-positive* (*moebius-circline* (*moebius-rotation*  $(-\pi/2)$ )  $H$ )  $\longleftrightarrow$  *intersects-y-axis-positive*  $H$

**using** *hermitean-elems*

**unfolding** *moebius-rotation-def* *moebius-similarity-def*

**by** *simp* (*transfer*, *transfer*, *auto* *simp* *add: mat-adj-def mat-cnj-def*)

**lemma** *intersects-y-axis-positive-iff*:

**assumes** *is-poincare-line*  $H\ H \neq \text{y-axis}$

**shows**  $(\exists y \in \text{unit-disc. } y \in \text{circline-set } H \cap \text{positive-y-axis}) \longleftrightarrow \text{intersects-y-axis-positive } H$  (**is**  $?lhs \longleftrightarrow ?rhs$ )

**proof** –

**let**  $?R = \text{moebius-rotation } (-\pi / 2)$

**let**  $?H' = \text{moebius-circline } ?R\ H$

**have**  $1: \text{is-poincare-line } ?H'$

**using** *assms*

**using** *unit-circle-fix-preserve-is-poincare-line*[*OF - assms(1), of ?R*]

**by** *simp*

**have**  $2: \text{moebius-circline } ?R\ H \neq \text{x-axis}$

**proof** (*rule ccontr*)

**assume**  $\neg ?thesis$

**hence**  $H = \text{moebius-circline } (\text{moebius-rotation } (\pi/2))\ \text{x-axis}$

**using** *moebius-circline-comp-inv-left*[*of ?R H*]

**by** *auto*

**thus** *False*

**using**  $\langle H \neq \text{y-axis} \rangle$

**by** *auto*

**qed**

**show**  $?thesis$

**proof**

**assume**  $?lhs$

**then obtain**  $y$  **where**  $y \in \text{unit-disc } y \in \text{circline-set } H \cap \text{positive-y-axis}$

**by** *auto*

**hence** *moebius-pt*  $?R\ y \in \text{unit-disc}$  *moebius-pt*  $?R\ y \in \text{circline-set } ?H' \cap \text{positive-x-axis}$

**using** *rotation-minus-pi-2-positive-y-axis*

**by** *auto*

**thus**  $?rhs$

**using** *intersects-x-axis-positive-iff*[*OF 1 2*]

**using** *intersects-x-axis-positive-intersects-y-axis-positive*[*of H*]

**by** *auto*

**next**

**assume** *intersects-y-axis-positive*  $H$

**hence** *intersects-x-axis-positive*  $?H'$

**using** *intersects-x-axis-positive-intersects-y-axis-positive*[*of H*]

**by** *simp*

```

then obtain  $x$  where  $*$ :  $x \in \text{unit-disc} \wedge x \in \text{circline-set } ?H' \cap \text{positive-x-axis}$ 
  using intersects-x-axis-positive-iff[OF 1 2]
  by auto
let  $?y = \text{moebius-pt } (-?R) x$ 
have  $?y \in \text{unit-disc} \wedge ?y \in \text{circline-set } H \cap \text{positive-y-axis}$ 
  using  $*$  rotation-minus-pi-2-positive-y-axis[symmetric]
  by (metis Int-iff circline-set-moebius-circline-E image-eqI moebius-pt-comp-inv-image-left moebius-rotation-uminus
uminus-moebius-def unit-disc-fix-discI unit-disc-fix-rotation)
  thus  $?lhs$ 
  by auto
qed
qed

```

## 8.7 Position of the intersection point in the unit disc

Check if the intersection point of one h-line with the x-axis is located more outward the edge of the disc than the intersection point of another h-line.

```

definition outward-cmat :: complex-mat  $\Rightarrow$  complex-mat  $\Rightarrow$  bool where
[simp]: outward-cmat  $H1 H2 = (\text{let } (A1, B1, C1, D1) = H1; (A2, B2, C2, D2) = H2$ 
   $\text{in } -\text{Re } B1 / \text{Re } A1 \leq -\text{Re } B2 / \text{Re } A2)$ 

```

```

lift-definition outward-clmat :: circline-mat  $\Rightarrow$  circline-mat  $\Rightarrow$  bool is outward-cmat
done

```

```

lift-definition outward :: circline  $\Rightarrow$  circline  $\Rightarrow$  bool is outward-clmat

```

```

apply transfer
apply simp
apply (case-tac circline-mat1, case-tac circline-mat2, case-tac circline-mat3, case-tac circline-mat4)
apply simp
apply (erule-tac exE) $+$ 
apply (erule-tac conjE) $+$ 
apply simp
done

```

**lemma** *outward-mk-circline*:

```

assumes is-real  $A1$  and is-real  $A2$  and  $A1 \neq 0 \vee B1 \neq 0$  and  $A2 \neq 0 \vee B2 \neq 0$ 
shows outward (mk-circline  $A1 B1 (\text{cnj } B1) A1$ ) (mk-circline  $A2 B2 (\text{cnj } B2) A2$ )  $\longleftrightarrow -\text{Re } B1 / \text{Re } A1 \leq -\text{Re } B2 / \text{Re } A2$ 

```

**proof** –

```

let  $?H1 = (A1, B1, (\text{cnj } B1), A1)$ 
let  $?H2 = (A2, B2, (\text{cnj } B2), A2)$ 
have hermitean  $?H1$  hermitean  $?H2$ 
  using  $\langle \text{is-real } A1 \rangle \langle \text{is-real } A2 \rangle$ 
  unfolding hermitean-def mat-adj-def mat-cnj-def
  using eq-cnj-iff-real
  by auto

```

**moreover**

```

have  $?H1 \neq \text{mat-zero} \wedge ?H2 \neq \text{mat-zero}$ 
  using assms
  by auto

```

**ultimately**

```

show  $?thesis$ 
  by (transfer, transfer, auto simp add: Let-def)

```

**qed**

**lemma** *calc-x-axis-intersection-fun-mono*:

```

fixes  $x1 x2 :: \text{real}$ 
assumes  $x1 > 1$  and  $x2 > x1$ 
shows  $x1 - \text{sqrt}(x1^2 - 1) > x2 - \text{sqrt}(x2^2 - 1)$ 
using assms

```

**proof** –

```

have  $*$ :  $\text{sqrt}(x1^2 - 1) + \text{sqrt}(x2^2 - 1) > 0$ 
  using assms
  by (smt one-less-power pos2 real-sqrt-gt-zero)

```

```

have  $\text{sqrt}(x1^2 - 1) < x1$ 
  using real-sqrt-less-iff[of x1^2 - 1 x1^2]  $\langle x1 > 1 \rangle$ 

```

```

    by auto
  moreover
  have  $\sqrt{x_2^2 - 1} < x_2$ 
    using real-sqrt-less-iff[of  $x_2^2 - 1$   $x_2^2$ ]  $\langle x_1 > 1 \rangle \langle x_2 > x_1 \rangle$ 
    by auto
  ultimately
  have  $\sqrt{x_1^2 - 1} + \sqrt{x_2^2 - 1} < x_1 + x_2$ 
    by simp
  hence  $(x_1 + x_2) / (\sqrt{x_1^2 - 1} + \sqrt{x_2^2 - 1}) > 1$ 
    using *
    using less-divide-eq-1-pos[of  $\sqrt{x_1^2 - 1} + \sqrt{x_2^2 - 1}$   $x_1 + x_2$ ]
    by simp
  hence  $(x_2^2 - x_1^2) / (\sqrt{x_1^2 - 1} + \sqrt{x_2^2 - 1}) > x_2 - x_1$ 
    using  $\langle x_2 > x_1 \rangle$ 
    using mult-less-cancel-left-pos[of  $x_2 - x_1$   $1$   $(x_2 + x_1) / (\sqrt{x_1^2 - 1} + \sqrt{x_2^2 - 1})$ ]
    by (simp add: power2-eq-square field-simps)
  moreover
  have  $(x_2^2 - x_1^2) = (\sqrt{x_1^2 - 1} + \sqrt{x_2^2 - 1}) * ((\sqrt{x_2^2 - 1} - \sqrt{x_1^2 - 1}))$ 
    using  $\langle x_1 > 1 \rangle \langle x_2 > x_1 \rangle$ 
    by (simp add: field-simps)
  ultimately
  have  $\sqrt{x_2^2 - 1} - \sqrt{x_1^2 - 1} > x_2 - x_1$ 
    using *
    by simp
  thus ?thesis
    by simp
qed

```

lemma *calc-x-axis-intersection-mono*:

```

  fixes  $a_1 b_1 a_2 b_2 :: real$ 
  assumes  $-b_1/a_1 > 1$  and  $a_1 \neq 0$  and  $-b_2/a_2 \geq -b_1/a_1$  and  $a_2 \neq 0$ 
  shows  $(-b_1 + \text{sgn } b_1 * \sqrt{b_1^2 - a_1^2}) / a_1 \geq (-b_2 + \text{sgn } b_2 * \sqrt{b_2^2 - a_2^2}) / a_2$  (is ?lhs  $\geq$  ?rhs)

```

proof-

```

  have ?lhs =  $-b_1/a_1 - \sqrt{(-b_1/a_1)^2 - 1}$ 
  proof (cases  $b_1 > 0$ )
  case True
  hence  $a_1 < 0$ 
    using assms
    by (smt divide-neg-pos)
  thus ?thesis
    using  $\langle b_1 > 0 \rangle \langle a_1 < 0 \rangle$ 
    by (simp add: real-sqrt-divide field-simps)

```

next

```

  case False
  hence  $b_1 < 0$ 
    using assms
    by (cases b_1 = 0) auto
  hence  $a_1 > 0$ 
    using assms
    by (smt divide-pos-neg)
  thus ?thesis
    using  $\langle b_1 < 0 \rangle \langle a_1 > 0 \rangle$ 
    by (simp add: real-sqrt-divide field-simps)

```

qed

moreover

```

  have ?rhs =  $-b_2/a_2 - \sqrt{(-b_2/a_2)^2 - 1}$ 
  proof (cases  $b_2 > 0$ )
  case True
  hence  $a_2 < 0$ 
    using assms
    by (smt divide-neg-pos)
  thus ?thesis
    using  $\langle b_2 > 0 \rangle \langle a_2 < 0 \rangle$ 
    by (simp add: real-sqrt-divide field-simps)

```



```

next
  case False
  hence  $b2 < 0$ 
    using assms
    by (cases b2 = 0) auto
  hence  $a2 > 0$ 
    using assms
    by (smt divide-pos-neg)
  thus ?thesis
    using  $\langle b2 < 0 \rangle \langle a2 > 0 \rangle$ 
    by (simp add: real-sqrt-divide field-simps)
qed

ultimately

show ?thesis
  using calc-x-axis-intersection-fun-mono[of  $-b1/a1 -b2/a2$ ]
  using assms
  by (cases -b1/a1=-b2/a2, auto)
qed

lemma outward:
  assumes is-poincare-line H1 and is-poincare-line H2
  assumes intersects-x-axis-positive H1 and intersects-x-axis-positive H2
  assumes outward H1 H2
  shows  $Re (to-complex (calc-x-axis-intersection H1)) \geq Re (to-complex (calc-x-axis-intersection H2))$ 
proof-
  have intersects-x-axis H1 intersects-x-axis H2
    using assms
    by auto
  thus ?thesis
    using assms
  proof (transfer, transfer)
    fix H1 H2
    assume hh: hermitean H1  $\wedge$  H1  $\neq$  mat-zero hermitean H2  $\wedge$  H2  $\neq$  mat-zero
    obtain A1 B1 C1 D1 A2 B2 C2 D2 where  $*$ :  $H1 = (A1, B1, C1, D1) H2 = (A2, B2, C2, D2)$ 
      by (cases H1, cases H2, auto)
    have is-real A1 is-real A2
      using hermitean-elems * hh
      by auto
    assume 1: intersects-x-axis-positive-cmat H1 intersects-x-axis-positive-cmat H2
    assume 2: intersects-x-axis-cmat H1 intersects-x-axis-cmat H2
    assume 3: is-poincare-line-cmat H1 is-poincare-line-cmat H2
    assume 4: outward-cmat H1 H2
    have  $A1 \neq 0 A2 \neq 0$ 
      using  $\langle is-real A1 \rangle \langle is-real A2 \rangle 1$  complex.expand[of  $A1 0$ ] complex.expand[of  $A2 0$ ]
      by auto
    hence  $(sgn (Re B2) * sqrt ((Re B2)^2 - (Re A2)^2) - Re B2) / Re A2$ 
       $\leq (sgn (Re B1) * sqrt ((Re B1)^2 - (Re A1)^2) - Re B1) / Re A1$ 
      using calc-x-axis-intersection-mono[of  $Re B1 Re A1 Re B2 Re A2$ ]
      using 1 4  $*$ 
      by simp
    moreover
    have  $(sgn (Re B2) * sqrt ((Re B2)^2 - (Re A2)^2) - Re B2) / Re A2 =$ 
       $Re ((cor (sgn (Re B2)) * cor (sqrt ((Re B2)^2 - (Re A2)^2)) - cor (Re B2)) / A2)$ 
      using  $\langle is-real A2 \rangle \langle A2 \neq 0 \rangle$ 
      by (simp add: Re-divide-real)
    moreover
    have  $(sgn (Re B1) * sqrt ((Re B1)^2 - (Re A1)^2) - Re B1) / Re A1 =$ 
       $Re ((cor (sgn (Re B1)) * cor (sqrt ((Re B1)^2 - (Re A1)^2)) - cor (Re B1)) / A1)$ 
      using  $\langle is-real A1 \rangle \langle A1 \neq 0 \rangle$ 
      by (simp add: Re-divide-real)
    ultimately
  show  $Re (to-complex-cvec (calc-x-axis-intersection-cmat-cvec H2))$ 
     $\leq Re (to-complex-cvec (calc-x-axis-intersection-cmat-cvec H1))$ 
    using 2 3  $\langle A1 \neq 0 \rangle \langle A2 \neq 0 \rangle * \langle is-real A1 \rangle \langle is-real A2 \rangle$ 

```

by (*simp del: is-poincare-line-cmat-def intersects-x-axis-cmat-def*)  
qed  
qed

## 8.8 Ideal points and x-axis intersection

**lemma** *ideal-points-intersects-x-axis:*

**assumes** *is-poincare-line H and ideal-points H = {i1, i2} and H ≠ x-axis*  
**shows** *intersects-x-axis H ↔ Im (to-complex i1) \* Im (to-complex i2) < 0*  
**using** *assms*

**proof**–

**have** *i1 ≠ i2*  
**using** *assms(1) assms(2) ex-poincare-line-points ideal-points-different(1)*  
**by** *blast*

**have** *calc-ideal-points H = {i1, i2}*  
**using** *assms*  
**using** *ideal-points-unique*  
**by** *auto*

**have**  $\forall i1 \in \text{calc-ideal-points } H.$

$\forall i2 \in \text{calc-ideal-points } H.$

$\text{is-poincare-line } H \wedge H \neq \text{x-axis} \wedge i1 \neq i2 \longrightarrow (\text{Im (to-complex } i1) * \text{Im (to-complex } i2) < 0 \longleftrightarrow \text{intersects-x-axis } H)$

**proof** (*transfer, transfer, (rule ballI)+, rule impI, (erule conjE)+, case-tac H, case-tac i1, case-tac i2*)

**fix** *i11 i12 i21 i22 A B C D H i1 i2*

**assume** *H: H = (A, B, C, D) hermitean H H ≠ mat-zero*

**assume** *line: is-poincare-line-cmat H*

**assume** *i1: i1 = (i11, i12) i1 ∈ calc-ideal-points-cmat-cvec H*

**assume** *i2: i2 = (i21, i22) i2 ∈ calc-ideal-points-cmat-cvec H*

**assume** *different: ¬ i1 ≈<sub>v</sub> i2*

**assume** *not-x-axis: ¬ circline-eq-cmat H x-axis-cmat*

**have** *is-real A is-real D C = conj B*

**using** *H hermitean-elems*

**by** *auto*

**have**  $(\text{cmod } A)^2 < (\text{cmod } B)^2 \wedge A = D$

**using** *line H*

**by** *auto*

**let** *?discr = sqrt ((cmod B)<sup>2</sup> - (Re D)<sup>2</sup>)*

**let** *?den = (cmod B)<sup>2</sup>*

**let** *?i1 = B \* (- D - i \* ?discr)*

**let** *?i2 = B \* (- D + i \* ?discr)*

**have** *i11 = ?i1 ∨ i11 = ?i2 i12 = ?den*

*i21 = ?i1 ∨ i21 = ?i2 i22 = ?den*

**using** *i1 i2 H line*

**by** (*auto split: if-split-asm*)

**hence** *i: i11 = ?i1 ∧ i21 = ?i2 ∨ i11 = ?i2 ∧ i21 = ?i1*

**using**  $\langle \neg i1 \approx_v i2 \rangle i1 i2$

**by** *auto*

**have**  $\text{Im (i11 / i12) * Im (i21 / i22) = Im (?i1 / ?den) * Im (?i2 / ?den)}$

**using**  $i \langle i12 = ?den \rangle \langle i22 = ?den \rangle$

**by** *auto*

**also** **have**  $\dots = \text{Im (?i1) * Im (?i2) / ?den}^2$

**by** *simp*

**also** **have**  $\dots = (\text{Im } B * (\text{Im } B * (\text{Re } D * \text{Re } D)) - \text{Re } B * (\text{Re } B * ((\text{cmod } B)^2 - (\text{Re } D)^2))) / \text{cmod } B \wedge 4$

**using**  $\langle (\text{cmod } B)^2 > (\text{cmod } A)^2 \rangle \langle A = D \rangle$

**using**  $\langle \text{is-real } D \rangle \text{cmod-eq-Re[of } A \rangle$

**by** (*auto simp add: field-simps*)

**also** **have**  $\dots = ((\text{Im } B)^2 * (\text{Re } D)^2 - (\text{Re } B)^2 * ((\text{Re } B)^2 + (\text{Im } B)^2 - (\text{Re } D)^2)) / \text{cmod } B \wedge 4$

**proof**–

**have**  $\text{cmod } B * \text{cmod } B = \text{Re } B * \text{Re } B + \text{Im } B * \text{Im } B$

**by** (*metis cmod-power2 power2-eq-square*)

```

    thus ?thesis
      by (simp add: power2-eq-square)
  qed
  also have ... = (((Re D)2 - (Re B)2) * ((Re B)2 + (Im B)2) / cmod B ^ 4
    by (simp add: power2-eq-square field-simps)
  finally have Im-product: Im (i11 / i12) * Im (i21 / i22) = ((Re D)2 - (Re B)2) * ((Re B)2 + (Im B)2) / cmod
    B ^ 4
    .

show Im (to-complex-cvec i1) * Im (to-complex-cvec i2) < 0 ⟷ intersects-x-axis-cmat H
proof safe
  assume opposite: Im (to-complex-cvec i1) * Im (to-complex-cvec i2) < 0
  show intersects-x-axis-cmat H
  proof-
    have ((Re D)2 - (Re B)2) * ((Re B)2 + (Im B)2) / cmod B ^ 4 < 0
      using Im-product opposite i1 i2
      by simp
    hence ((Re D)2 - (Re B)2) * ((Re B)2 + (Im B)2) < 0
      by (simp add: divide-less-0-iff)
    hence (Re D)2 < (Re B)2
      by (simp add: mult-less-0-iff not-sum-power2-lt-zero)
    thus ?thesis
      using H ⟨A = D⟩ ⟨is-real D⟩
      by auto
  qed
next
  have *: (∀ k. k * Im B = 1 ⟶ k = 0) ⟶ Im B = 0
    apply (safe, erule-tac x=1 / Im B in allE)
    using divide-cancel-left by fastforce
  assume intersects-x-axis-cmat H
  hence Re D = 0 ∨ (Re D)2 < (Re B)2
    using H ⟨A = D⟩
    by auto
  hence (Re D)2 < (Re B)2
    using ⟨is-real D⟩ line H ⟨C = cnj B⟩
    using not-x-axis *
    by (auto simp add: complex-eq-iff)
  hence ((Re D)2 - (Re B)2) * ((Re B)2 + (Im B)2) < 0
  by (metis add-cancel-left-left diff-less-eq mult-eq-0-iff mult-less-0-iff power2-eq-square power2-less-0 sum-squares-gt-zero-iff)
  thus Im (to-complex-cvec i1) * Im (to-complex-cvec i2) < 0
    using Im-product i1 i2
    using divide-eq-0-iff divide-less-0-iff prod.simps(2) to-complex-cvec-def zero-complex.simps(1) zero-less-norm-iff
    by fastforce
  qed
qed
thus ?thesis
  using assms ⟨calc-ideal-points H = {i1, i2}⟩ ⟨i1 ≠ i2⟩
  by auto
qed

end
theory Poincare-Perpendicular
  imports Poincare-Lines-Axis-Intersections
begin

```

## 9 H-perpendicular h-lines in the Poincaré model

**definition** *perpendicular-to-x-axis-cmat* :: complex-mat ⇒ bool **where**  
 [simp]: *perpendicular-to-x-axis-cmat* H ⟷ (let (A, B, C, D) = H in is-real B)

**lift-definition** *perpendicular-to-x-axis-clmat* :: circline-mat ⇒ bool **is** *perpendicular-to-x-axis-cmat*  
 done

**lift-definition** *perpendicular-to-x-axis* :: circline ⇒ bool **is** *perpendicular-to-x-axis-clmat*  
 by transfer auto

**lemma** *perpendicular-to-x-axis*:  
**assumes** *is-poincare-line*  $H$   
**shows** *perpendicular-to-x-axis*  $H \longleftrightarrow$  *perpendicular x-axis*  $H$   
**using** *assms*  
**unfolding** *perpendicular-def*  
**proof** (*transfer*, *transfer*)  
**fix**  $H$   
**assume** *hh*: *hermitean*  $H \wedge H \neq$  *mat-zero is-poincare-line-cmat*  $H$   
**obtain**  $A B C D$  **where**  $*$ :  $H = (A, B, C, D)$   
**by** (*cases*  $H$ , *auto*)  
**hence** *is-real*  $A \pmod{B}^2 > \pmod{A}^2$   $H = (A, B, \text{cnj } B, A)$   
**using** *hermitean-elems*[*of*  $A B C D$ ] *hh*  
**by** *auto*  
**thus** *perpendicular-to-x-axis-cmat*  $H =$   
*(cos-angle-cmat (of-circline-cmat x-axis-cmat) (of-circline-cmat H) = 0)*  
**using** *cmode-square*[*of*  $B$ ] *cmode-square*[*of*  $A$ ]  
**by** *simp*  
**qed**

**lemma** *perpendicular-to-x-axis-y-axis*:  
**assumes** *perpendicular-to-x-axis* (*poincare-line*  $0_h$  (*of-complex*  $z$ ))  $z \neq 0$   
**shows** *is-imag*  $z$   
**using** *assms*  
**by** (*transfer*, *transfer*, *simp*)

**lemma** *wlog-perpendicular-axes*:  
**assumes** *in-disc*:  $u \in$  *unit-disc*  $v \in$  *unit-disc*  $z \in$  *unit-disc*  
**assumes** *perpendicular*: *is-poincare-line*  $H1$  *is-poincare-line*  $H2$  *perpendicular*  $H1 H2$   
**assumes**  $z \in$  *circline-set*  $H1 \cap$  *circline-set*  $H2$   $u \in$  *circline-set*  $H1$   $v \in$  *circline-set*  $H2$   
**assumes** *axes*:  $\bigwedge x y. \llbracket \text{is-real } x; 0 \leq \text{Re } x; \text{Re } x < 1; \text{is-imag } y; 0 \leq \text{Im } y; \text{Im } y < 1 \rrbracket \implies P 0_h$  (*of-complex*  $x$ )  
(*of-complex*  $y$ )  
**assumes** *moebius*:  $\bigwedge M u v w. \llbracket \text{unit-disc-fix } M; u \in$  *unit-disc*;  $v \in$  *unit-disc*;  $w \in$  *unit-disc*;  $P$  (*moebius-pt*  $M u$ )  
(*moebius-pt*  $M v$ ) (*moebius-pt*  $M w$ )  $\rrbracket \implies P u v w$   
**assumes** *conjugate*:  $\bigwedge u v w. \llbracket u \in$  *unit-disc*;  $v \in$  *unit-disc*;  $w \in$  *unit-disc*;  $P$  (*conjugate*  $u$ ) (*conjugate*  $v$ ) (*conjugate*  $w$ )  $\rrbracket \implies P u v w$   
**shows**  $P z u v$   
**proof** –  
**have**  $\forall v H1 H2. \text{is-poincare-line } H1 \wedge \text{is-poincare-line } H2 \wedge \text{perpendicular } H1 H2 \wedge$   
 $z \in \text{circline-set } H1 \cap \text{circline-set } H2 \wedge u \in \text{circline-set } H1 \wedge v \in \text{circline-set } H2 \wedge v \in \text{unit-disc} \longrightarrow P$   
 $z u v$  (**is**  $?P z u$ )  
**proof** (*rule wlog-x-axis*[**where**  $P = ?P$ ])  
**fix**  $x$   
**assume**  $x$ : *is-real*  $x$   $\text{Re } x \geq 0$   $\text{Re } x < 1$   
**have** *of-complex*  $x \in$  *unit-disc*  
**using**  $x$   
**by** (*simp add*: *cmode-eq-Re*)  
**show**  $?P 0_h$  (*of-complex*  $x$ )  
**proof** *safe*  
**fix**  $v H1 H2$   
**assume**  $v \in$  *unit-disc*  
**then obtain**  $y$  **where**  $y$ :  $v =$  *of-complex*  $y$   
**using** *inf-or-of-complex*[*of*  $v$ ]  
**by** *auto*  
**assume**  $1$ : *is-poincare-line*  $H1$  *is-poincare-line*  $H2$  *perpendicular*  $H1 H2$   
**assume**  $2$ :  $0_h \in$  *circline-set*  $H1$   $0_h \in$  *circline-set*  $H2$  *of-complex*  $x \in$  *circline-set*  $H1$   $v \in$  *circline-set*  $H2$   
**show**  $P 0_h$  (*of-complex*  $x$ )  $v$   
**proof** (*cases of-complex*  $x = 0_h$ )  
**case** *True*  
**show**  $P 0_h$  (*of-complex*  $x$ )  $v$   
**proof** (*cases*  $v = 0_h$ )  
**case** *True*

```

thus ?thesis
  using ⟨of-complex  $x = 0_h$ ⟩
  using axes[of 0 0]
  by simp
next
case False
show ?thesis
proof (rule wlog-rotation-to-positive-y-axis)
  show  $v \in \text{unit-disc } v \neq 0_h$ 
    by fact+
next
  fix y
  assume is-imag y  $0 < \text{Im } y$   $\text{Im } y < 1$ 
  thus  $P 0_h$  (of-complex x) (of-complex y)
    using x axes[of x y]
    by simp
next
  fix  $\varphi$  u
  assume  $u \in \text{unit-disc } u \neq 0_h$ 
     $P 0_h$  (of-complex x) (moebius-pt (moebius-rotation  $\varphi$ ) u)
  thus  $P 0_h$  (of-complex x) u
    using ⟨of-complex  $x = 0_h$ ⟩
    using moebius[of moebius-rotation  $\varphi$   $0_h$   $0_h$  u]
    by simp
  qed
qed
next
case False
hence *: poincare-line  $0_h$  (of-complex x) = x-axis
  using x poincare-line-0-real-is-x-axis[of of-complex x]
  unfolding circline-set-x-axis
  by auto
hence H1 = x-axis
  using unique-poincare-line[of  $0_h$  of-complex x H1] 1 2
  using ⟨of-complex  $x \in \text{unit-disc}$ ⟩ False
  by simp
have is-imag y
proof (cases  $y = 0$ )
  case True
  thus ?thesis
    by simp
next
  case False
  hence  $0_h \neq$  of-complex y
    using of-complex-zero-iff[of y]
    by metis
  hence H2 = poincare-line  $0_h$  (of-complex y)
    using 1 2 ⟨ $v \in \text{unit-disc}$ ⟩
    using unique-poincare-line[of  $0_h$  of-complex y H2] y
    by simp
  thus ?thesis
    using 1 ⟨H1 = x-axis⟩
    using perpendicular-to-x-axis-y-axis[of y] False
    using perpendicular-to-x-axis[of H2]
    by simp
  qed
show  $P 0_h$  (of-complex x) v
proof (cases  $\text{Im } y \geq 0$ )
  case True
  thus ?thesis
    using axes[of x y] x y ⟨is-imag y⟩ ⟨ $v \in \text{unit-disc}$ ⟩
    by (simp add: cmod-eq-Im)
next
  case False
  show ?thesis
  proof (rule conjugate)

```

```

    have  $Im (cnj y) < 1$ 
      using  $\langle v \in unit-disc \rangle y \langle is-imag y \rangle eq-minus-cnj-iff-imag[of y]$ 
      by (simp add: cmod-eq-Im)
    thus  $P (conjugate 0_h) (conjugate (of-complex x)) (conjugate v)$ 
      using  $\langle is-real x \rangle eq-cnj-iff-real[of x] y \langle is-imag y \rangle$ 
      using axes[OF x, of cnj y] False
      by simp
    show  $0_h \in unit-disc$  of-complex  $x \in unit-disc$   $v \in unit-disc$ 
      by (simp, fact+)
  qed
qed
qed
qed
next
show  $z \in unit-disc$   $u \in unit-disc$ 
  by fact+
next
fix  $M u v$ 
assume *:  $unit-disc$ -fix  $M u \in unit-disc$   $v \in unit-disc$ 
assume **:  $?P (moebius-pt M u) (moebius-pt M v)$ 
show  $?P u v$ 
proof safe
  fix  $w H1 H2$ 
  assume ***:  $is-poincare-line H1 is-poincare-line H2$  perpendicular  $H1 H2$ 
     $u \in circline-set H1$   $u \in circline-set H2$ 
     $v \in circline-set H1$   $w \in circline-set H2$   $w \in unit-disc$ 
  thus  $P u v w$ 
    using moebius[of M u v w] *
    using **[rule-format, of moebius-circline M H1 moebius-circline M H2 moebius-pt M w]
    by simp
  qed
qed
thus ?thesis
  using assms
  by blast
qed

lemma wlog-perpendicular-foot:
  assumes  $in-disc: u \in unit-disc$   $v \in unit-disc$   $w \in unit-disc$   $z \in unit-disc$ 
  assumes perpendicular:  $u \neq v$   $is-poincare-line H$  perpendicular (poincare-line u v)  $H$ 
  assumes  $z \in circline-set (poincare-line u v) \cap circline-set H$   $w \in circline-set H$ 
  assumes axes:  $\bigwedge u v w. \llbracket is-real u; 0 < Re u; Re u < 1; is-real v; -1 < Re v; Re v < 1; Re u \neq Re v; is-imag w; 0 \leq Im w; Im w < 1 \rrbracket \implies P 0_h (of-complex u) (of-complex v) (of-complex w)$ 
  assumes moebius:  $\bigwedge M z u v w. \llbracket unit-disc$ -fix  $M; u \in unit-disc; v \in unit-disc; w \in unit-disc; z \in unit-disc; P (moebius-pt M z) (moebius-pt M u) (moebius-pt M v) (moebius-pt M w) \rrbracket \implies P z u v w$ 
  assumes conjugate:  $\bigwedge z u v w. \llbracket u \in unit-disc; v \in unit-disc; w \in unit-disc; P (conjugate z) (conjugate u) (conjugate v) (conjugate w) \rrbracket \implies P z u v w$ 
  assumes perm:  $P z v u w \implies P z u v w$ 
  shows  $P z u v w$ 
proof -
  obtain  $m n$  where  $mn: m = u \vee m = v$   $n = u \vee n = v$   $m \neq n$   $m \neq z$ 
    using  $\langle u \neq v \rangle$ 
    by auto

  have  $n \in circline-set (poincare-line z m)$ 
    using  $\langle z \in circline-set (poincare-line u v) \cap circline-set H \rangle$ 
    using mn
    using unique-poincare-line[of z m poincare-line u v, symmetric] in-disc
    by auto

  have  $\forall n. n \in unit-disc \wedge m \neq n \wedge n \in circline-set (poincare-line z m) \wedge m \neq z \longrightarrow P z m n w$  (is ?Q z m w)
  proof (rule wlog-perpendicular-axes[where P=?Q])
    show  $is-poincare-line (poincare-line u v)$ 
      using  $\langle u \neq v \rangle$ 
      by auto
  next

```

```

show is-poincare-line H
  by fact
next
show  $m \in \text{unit-disc}$   $m \in \text{circline-set (poincare-line } u \ v)$ 
  using  $mn \text{ in-disc}$ 
  by auto
next
show  $w \in \text{unit-disc}$   $z \in \text{unit-disc}$ 
  by fact+
next
show  $z \in \text{circline-set (poincare-line } u \ v) \cap \text{circline-set } H$ 
  by fact
next
show perpendicular (poincare-line  $u \ v$ )  $H$ 
  by fact
next
show  $w \in \text{circline-set } H$ 
  by fact
next
fix  $x \ y$ 
assume  $xy$ :  $is\text{-real } x \ 0 \leq \text{Re } x \ \text{Re } x < 1$   $is\text{-imag } y \ 0 \leq \text{Im } y \ \text{Im } y < 1$ 
show  $?Q \ 0_h \ (\text{of-complex } x) \ (\text{of-complex } y)$ 
proof safe
  fix  $n$ 
  assume  $n \in \text{unit-disc}$   $\text{of-complex } x \neq n$ 
  assume  $n \in \text{circline-set (poincare-line } 0_h \ (\text{of-complex } x))$   $\text{of-complex } x \neq 0_h$ 
  hence  $n \in \text{circline-set } x\text{-axis}$ 
  using  $\text{poincare-line-0-real-is-}x\text{-axis[of of-complex } x] \ xy$ 
  by (auto simp add:  $\text{circline-set-}x\text{-axis}$ )
  then obtain  $n'$  where  $n'$ :  $n = \text{of-complex } n'$ 
  using  $\text{inf-or-of-complex[of } n] \ \langle n \in \text{unit-disc} \rangle$ 
  by auto
  hence  $is\text{-real } n'$ 
  using  $\langle n \in \text{circline-set } x\text{-axis} \rangle$ 
  using  $\text{of-complex-inj}$ 
  unfolding  $\text{circline-set-}x\text{-axis}$ 
  by auto
  hence  $-1 < \text{Re } n' \ \text{Re } n' < 1$ 
  using  $\langle n \in \text{unit-disc} \rangle \ n'$ 
  by (auto simp add:  $\text{cmod-eq-Re}$ )

  have  $\text{Re } n' \neq \text{Re } x$ 
  using  $\text{complex.expand[of } n' \ x] \ \langle is\text{-real } n' \rangle \ \langle is\text{-real } x \rangle \ \langle \text{of-complex } x \neq n \rangle \ n'$ 
  by auto

  have  $\text{Re } x > 0$ 
  using  $xy \ \langle \text{of-complex } x \neq 0_h \rangle$ 
  by (cases  $\text{Re } x = 0$ , auto simp add:  $\text{complex.expand}$ )

  show  $P \ 0_h \ (\text{of-complex } x) \ n \ (\text{of-complex } y)$ 
  using  $\text{axes[of } x \ n' \ y] \ xy \ n' \ \langle \text{Re } x > 0 \rangle \ \langle is\text{-real } n' \rangle \ \langle -1 < \text{Re } n' \rangle \ \langle \text{Re } n' < 1 \rangle \ \langle \text{Re } n' \neq \text{Re } x \rangle$ 
  by simp
qed
next
fix  $M \ u \ v \ w$ 
assume 1:  $\text{unit-disc-fix } M \ u \in \text{unit-disc}$   $v \in \text{unit-disc}$   $w \in \text{unit-disc}$ 
assume 2:  $?Q \ (\text{moebius-pt } M \ u) \ (\text{moebius-pt } M \ v) \ (\text{moebius-pt } M \ w)$ 
show  $?Q \ u \ v \ w$ 
proof safe
  fix  $n$ 
  assume  $n \in \text{unit-disc}$   $v \neq n$   $n \in \text{circline-set (poincare-line } u \ v)$   $v \neq u$ 
  thus  $P \ u \ v \ n \ w$ 
  using  $\text{moebius[of } M \ v \ n \ w \ u] \ 1 \ 2[\text{rule-format, of moebius-pt } M \ n]$ 
  by fastforce
qed
next

```

```

fix u v w
assume 1: u ∈ unit-disc v ∈ unit-disc w ∈ unit-disc
assume 2: ?Q (conjugate u) (conjugate v) (conjugate w)
show ?Q u v w
proof safe
  fix n
  assume n ∈ unit-disc v ≠ n n ∈ circline-set (poincare-line u v) v ≠ u
  thus P u v n w
    using conjugate[of v n w u] 1 2[rule-format, of conjugate n]
    using conjugate-inj
    by auto
qed
qed
thus ?thesis
  using mn in-disc ⟨n ∈ circline-set (poincare-line z m)⟩ perm
  by auto
qed

lemma perpendicular-to-x-axis-intersects-x-axis:
assumes is-poincare-line H perpendicular-to-x-axis H
shows intersects-x-axis H
using assms hermitean-elems
by (transfer, transfer, auto simp add: cmod-eq-Re)

lemma perpendicular-intersects:
assumes is-poincare-line H1 is-poincare-line H2
assumes perpendicular H1 H2
shows ∃ z. z ∈ unit-disc ∧ z ∈ circline-set H1 ∩ circline-set H2 (is ?P' H1 H2)
proof –
  have ∀ H2. is-poincare-line H2 ∧ perpendicular H1 H2 → ?P' H1 H2 (is ?P H1)
  proof (rule wlog-line-x-axis)
    show ?P x-axis
    proof safe
      fix H2
      assume is-poincare-line H2 perpendicular x-axis H2
      thus ∃ z. z ∈ unit-disc ∧ z ∈ circline-set x-axis ∩ circline-set H2
        using perpendicular-to-x-axis[of H2]
        using perpendicular-to-x-axis-intersects-x-axis[of H2]
        using intersects-x-axis-iff[of H2]
        by auto
    qed
  next
    fix M
    assume unit-disc-fix M
    assume *: ?P (moebius-circline M H1)
    show ?P H1
    proof safe
      fix H2
      assume is-poincare-line H2 perpendicular H1 H2
      then obtain z where z ∈ unit-disc z ∈ circline-set (moebius-circline M H1) ∧ z ∈ circline-set (moebius-circline
M H2)
        using *[rule-format, of moebius-circline M H2] ⟨unit-disc-fix M⟩
        by auto
      thus ∃ z. z ∈ unit-disc ∧ z ∈ circline-set H1 ∩ circline-set H2
        using ⟨unit-disc-fix M⟩
        by (rule-tac x=moebius-pt (–M) z in exI)
          (metis IntI add.inverse-inverse circline-set-moebius-circline-iff moebius-pt-comp-inv-left uminus-moebius-def
unit-disc-fix-discI unit-disc-fix-moebius-uminus)
    qed
  next
    show is-poincare-line H1
    by fact
  qed
thus ?thesis
using assms

```



by auto  
qed

**definition** *calc-perpendicular-to-x-axis-cmat* :: *complex-vec*  $\Rightarrow$  *complex-mat* **where**

```
[simp]: calc-perpendicular-to-x-axis-cmat z =
  (let (z1, z2) = z
    in if z1*cnj z2 + z2*cnj z1 = 0 then
      (0, 1, 1, 0)
    else
      let A = z1*cnj z2 + z2*cnj z1;
          B = -(z1*cnj z1 + z2*cnj z2)
          in (A, B, B, A)
  )
```

**lift-definition** *calc-perpendicular-to-x-axis-clmat* :: *complex-homo-coords*  $\Rightarrow$  *circline-mat* **is** *calc-perpendicular-to-x-axis-cmat*  
by (auto simp add: hermitean-def mat-adj-def mat-cnj-def Let-def split: if-split-asm)

**lift-definition** *calc-perpendicular-to-x-axis* :: *complex-homo*  $\Rightarrow$  *circline* **is** *calc-perpendicular-to-x-axis-clmat*

**proof** (transfer)

```
fix z w
assume z  $\neq$  vec-zero w  $\neq$  vec-zero
obtain z1 z2 w1 w2 where zw: z = (z1, z2) w = (w1, w2)
  by (cases z, cases w, auto)
assume z  $\approx_v$  w
then obtain k where *: k  $\neq$  0 w1 = k*z1 w2 = k*z2
  using zw
  by auto
have w1 * cnj w2 + w2 * cnj w1 = (k * cnj k) * (z1 * cnj z2 + z2 * cnj z1)
  using *
  by (auto simp add: field-simps)
moreover
have w1 * cnj w1 + w2 * cnj w2 = (k * cnj k) * (z1 * cnj z1 + z2 * cnj z2)
  using *
  by (auto simp add: field-simps)
ultimately
show circline-eq-cmat (calc-perpendicular-to-x-axis-cmat z) (calc-perpendicular-to-x-axis-cmat w)
  using zw *
  apply (auto simp add: Let-def)
  apply (rule-tac x=Re (k * cnj k) in exI, auto simp add: complex.expand field-simps)
  done
```

qed

**lemma** *calc-perpendicular-to-x-axis*:

```
assumes z  $\neq$  of-complex 1 z  $\neq$  of-complex (-1)
shows z  $\in$  circline-set (calc-perpendicular-to-x-axis z)  $\wedge$ 
  is-poincare-line (calc-perpendicular-to-x-axis z)  $\wedge$ 
  perpendicular-to-x-axis (calc-perpendicular-to-x-axis z)
using assms
unfolding circline-set-def perpendicular-def
```

**proof** (simp, transfer, transfer)

```
fix z :: complex-vec
obtain z1 z2 where z: z = (z1, z2)
  by (cases z, auto)
assume **:  $\neg$  z  $\approx_v$  of-complex-cvec 1  $\neg$  z  $\approx_v$  of-complex-cvec (-1)
show on-circline-cmat-cvec (calc-perpendicular-to-x-axis-cmat z) z  $\wedge$ 
  is-poincare-line-cmat (calc-perpendicular-to-x-axis-cmat z)  $\wedge$ 
  perpendicular-to-x-axis-cmat (calc-perpendicular-to-x-axis-cmat z)
proof (cases z1*cnj z2 + z2*cnj z1 = 0)
  case True
  thus ?thesis
    using z
    by (simp add: vec-cnj-def hermitean-def mat-adj-def mat-cnj-def mult.commute)
```

**next**

```
case False
hence z2  $\neq$  0
```

```

using z
by auto
hence  $Re(z2 * cnj z2) \neq 0$ 
using  $\langle z2 \neq 0 \rangle$ 
by (auto simp add: complex.expand)

have  $z1 \neq -z2 \wedge z1 \neq z2$ 
proof (rule ccontr)
  assume  $\neg ?thesis$ 
  hence  $z \approx_v \text{of-complex-cvec } 1 \vee z \approx_v \text{of-complex-cvec } (-1)$ 
  using  $z \langle z2 \neq 0 \rangle$ 
  by auto
  thus False
  using **
  by auto
qed

let ?A =  $z1 * cnj z2 + z2 * cnj z1$  and ?B =  $-(z1 * cnj z1 + z2 * cnj z2)$ 
have  $Re(z1 * cnj z1 + z2 * cnj z2) \geq 0$ 
by auto
hence  $Re ?B \leq 0$ 
by (smt uminus-complex.simps(1))
hence  $abs(Re ?B) = -Re ?B$ 
by auto
also have  $\dots = (Re z1)^2 + (Im z1)^2 + (Re z2)^2 + (Im z2)^2$ 
by (simp add: power2-eq-square[symmetric])
also have  $\dots > abs(Re ?A)$ 
proof (cases  $Re ?A \geq 0$ )
  case False
  have  $(Re z1 + Re z2)^2 + (Im z1 + Im z2)^2 > 0$ 
  using  $\langle z1 \neq -z2 \wedge z1 \neq z2 \rangle$ 
  by (metis add.commute add.inverse-unique complex-neq-0 plus-complex.code plus-complex.simps)
  thus ?thesis
  using False
  by (simp add: power2-sum power2-eq-square field-simps)
next
  case True
  have  $(Re z1 - Re z2)^2 + (Im z1 - Im z2)^2 > 0$ 
  using  $\langle z1 \neq -z2 \wedge z1 \neq z2 \rangle$ 
  by (meson complex-eq-iff right-minus-eq sum-power2-gt-zero-iff)
  thus ?thesis
  using True
  by (simp add: power2-sum power2-eq-square field-simps)
qed
finally
have  $abs(Re ?B) > abs(Re ?A)$ 
.
moreover
have  $cmod ?B = abs(Re ?B) \text{ cmod } ?A = abs(Re ?A)$ 
by (simp-all add: cmod-eq-Re)
ultimately
have  $(cmod ?B)^2 > (cmod ?A)^2$ 
by (smt power2-le-imp-le)
thus ?thesis
using z False
by (simp-all add: Let-def hermitean-def mat-adj-def mat-cnj-def cmod-eq-Re vec-cnj-def field-simps)
qed
qed

lemma ex-perpendicular:
  assumes is-poincare-line  $H z \in \text{unit-disc}$ 
  shows  $\exists H'. \text{is-poincare-line } H' \wedge \text{perpendicular } H H' \wedge z \in \text{circline-set } H' \text{ (is } ?P' H z)$ 
proof -
  have  $\forall z. z \in \text{unit-disc} \longrightarrow ?P' H z \text{ (is } ?P H)$ 
  proof (rule wlog-line-x-axis)
    show ?P x-axis

```

```

proof safe
  fix z
  assume z ∈ unit-disc
  then have z ≠ of-complex 1 z ≠ of-complex (-1)
  by auto
  thus ?P' x-axis z
  using ⟨z ∈ unit-disc⟩
  using calc-perpendicular-to-x-axis[of z] perpendicular-to-x-axis
  by (rule-tac x = calc-perpendicular-to-x-axis z in exI, auto)
qed
next
fix M
assume unit-disc-fix M
assume *: ?P (moebius-circline M H)
show ?P H
proof safe
  fix z
  assume z ∈ unit-disc
  hence moebius-pt M z ∈ unit-disc
  using ⟨unit-disc-fix M⟩
  by auto
then obtain H' where *: is-poincare-line H' perpendicular (moebius-circline M H) H' moebius-pt M z ∈ circline-set
H'
  using *
  by auto
have h: H = moebius-circline (-M) (moebius-circline M H)
  by auto
show ?P' H z
  using * ⟨unit-disc-fix M⟩
  apply (subst h)
  apply (rule-tac x=moebius-circline (-M) H' in exI)
  apply (simp del: moebius-circline-comp-inv-left)
  done
qed
qed fact
thus ?thesis
  using assms
  by simp
qed

lemma ex-perpendicular-foot:
assumes is-poincare-line H z ∈ unit-disc
shows ∃ H'. is-poincare-line H' ∧ z ∈ circline-set H' ∧ perpendicular H H' ∧
  (∃ z' ∈ unit-disc. z' ∈ circline-set H' ∩ circline-set H)
using assms
using ex-perpendicular[OF assms]
using perpendicular-intersects[of H]
by blast

lemma Pythagoras:
assumes in-disc: u ∈ unit-disc v ∈ unit-disc w ∈ unit-disc v ≠ w
assumes distinct[u, v, w] → perpendicular (poincare-line u v) (poincare-line u w)
shows cosh (poincare-distance v w) = cosh (poincare-distance u v) * cosh (poincare-distance u w) (is ?P' u v w)
proof (cases distinct [u, v, w])
case False
thus ?thesis
  using in-disc
  by (auto simp add: poincare-distance-sym)
next
case True
have distinct [u, v, w] → ?P' u v w (is ?P u v w)
proof (rule wlog-perpendicular-axes[where P=?P])
  show is-poincare-line (poincare-line u v) is-poincare-line (poincare-line u w)
  using ⟨distinct [u, v, w]⟩
  by simp-all
next

```

```

show perpendicular (poincare-line u v) (poincare-line u w)
  using True assms
  by simp
next
show u ∈ unit-disc v ∈ unit-disc w ∈ unit-disc
  by fact+
next
show v ∈ circline-set (poincare-line u v) w ∈ circline-set (poincare-line u w)
  u ∈ circline-set (poincare-line u v) ∩ circline-set (poincare-line u w)
  using ⟨distinct [u, v, w]⟩
  by auto
next
fix x y
assume x: is-real x 0 ≤ Re x Re x < 1
assume y: is-imag y 0 ≤ Im y Im y < 1

have of-complex x ∈ unit-disc of-complex y ∈ unit-disc
  using x y
  by (simp-all add: cmod-eq-Re cmod-eq-Im)

show ?P 0h (of-complex x) (of-complex y)
proof
  assume distinct [0h, of-complex x, of-complex y]
  hence x ≠ 0 y ≠ 0
    by auto

  let ?den1 = 1 - (cmod x)2 and ?den2 = 1 - (cmod y)2
  have ?den1 > 0 ?den2 > 0
    using x y
    by (simp-all add: cmod-eq-Re cmod-eq-Im abs-square-less-1)

  let ?d1 = 1 + 2 * (cmod x)2 / ?den1
  have cosh (poincare-distance 0h (of-complex x)) = ?d1
    using ⟨?den1 > 0⟩
    using poincare-distance-formula[of 0h of-complex x] ⟨of-complex x ∈ unit-disc⟩
    by simp

  moreover

  let ?d2 = 1 + 2 * (cmod y)2 / ?den2
  have cosh (poincare-distance 0h (of-complex y)) = ?d2
    using ⟨?den2 > 0⟩ ⟨of-complex y ∈ unit-disc⟩
    using poincare-distance-formula[of 0h of-complex y]
    by simp

  moreover

  let ?den = ?den1 * ?den2
  let ?d3 = 1 + 2 * (cmod (x - y))2 / ?den
  have cosh (poincare-distance (of-complex x) (of-complex y)) = ?d3
    using ⟨of-complex x ∈ unit-disc⟩ ⟨of-complex y ∈ unit-disc⟩
    using ⟨?den1 > 0⟩ ⟨?den2 > 0⟩
    using poincare-distance-formula[of of-complex x of-complex y]
    by simp
  moreover
  have ?d1 * ?d2 = ?d3
  proof-
    have ?d3 = ((1 - (cmod x)2) * (1 - (cmod y)2) + 2 * (cmod (x - y))2) / ?den
      using ⟨?den1 > 0⟩ ⟨?den2 > 0⟩
      by (subst add-num-frac, simp, simp)
    also have ... = (Re ((1 - x * cnj x) * (1 - y * cnj y) + 2 * (x - y) * cnj (x - y))) / ?den
      using ⟨is-real x⟩ ⟨is-imag y⟩
      by ((subst cmod-square)+, simp)
    also have ... = Re (1 + x * cnj x * y * cnj y
      + x * cnj x - 2 * y * cnj x - 2 * x * cnj y + y * cnj y) / ?den
      by (simp add: field-simps)
    also have ... = Re ((1 + y * cnj y) * (1 + x * cnj x)) / ?den

```

```

    using ⟨is-real x⟩ ⟨is-imag y⟩
    by (simp add: field-simps)
  finally
  show ?thesis
    using ⟨?den1 > 0⟩ ⟨?den2 > 0⟩
    apply (subst add-num-frac, simp)
    apply (subst add-num-frac, simp)
    apply simp
    apply (subst cmod-square)+
    apply (simp add: field-simps)
  done
qed
ultimately
show ?P' 0h (of-complex x) (of-complex y)
  by simp
qed
next
fix M u v w
assume 1: unit-disc-fix M u ∈ unit-disc v ∈ unit-disc w ∈ unit-disc
assume 2: ?P (moebius-pt M u) (moebius-pt M v) (moebius-pt M w)
show ?P u v w
  using 1 2
  by auto
next
fix u v w
assume 1: u ∈ unit-disc v ∈ unit-disc w ∈ unit-disc
assume 2: ?P (conjugate u) (conjugate v) (conjugate w)
show ?P u v w
  using 1 2
  by (auto simp add: conjugate-inj)
qed
thus ?thesis
  using True
  by simp
qed
end

```

## 10 Poincaré disc model types

In this section we introduce datatypes that represent objects in the Poincaré disc model. The types are defined as subtypes (e.g., the h-points are defined as elements of  $\mathbb{C}P^1$  that lie within the unit disc). The functions on those types are defined by lifting the functions defined on the carrier type (e.g., h-distance is defined by lifting the distance function defined for elements of  $\mathbb{C}P^1$ ).

```

theory Poincare
imports Poincare-Lines Poincare-Between Poincare-Distance Poincare-Circles
begin

```

### 10.1 H-points

```

typedef p-point = {z. z ∈ unit-disc}
  using zero-in-unit-disc
  by (rule-tac x=0h in exI, simp)

```

```

setup-lifting type-definition-p-point

```

Point zero

```

lift-definition p-zero :: p-point is 0h
  by (rule zero-in-unit-disc)

```

Constructing h-points from complex numbers

```

lift-definition p-of-complex :: complex ⇒ p-point is λ z. if cmod z < 1 then of-complex z else 0h
  by auto

```

## 10.2 H-lines

```
typedef p-line = {H. is-poincare-line H}
  by (rule-tac x=x-axis in exI, simp)
```

setup-lifting type-definition-p-line

```
lift-definition p-incident :: p-line  $\Rightarrow$  p-point  $\Rightarrow$  bool is on-circline
  done
```

Set of h-points on an h-line

```
definition p-points :: p-line  $\Rightarrow$  p-point set where
  p-points l = {p. p-incident l p}
```

x-axis is an example of an h-line

```
lift-definition p-x-axis :: p-line is x-axis
  by simp
```

Constructing the unique h-line from two h-points

```
lift-definition p-line :: p-point  $\Rightarrow$  p-point  $\Rightarrow$  p-line is poincare-line
proof-
  fix u v
  show is-poincare-line (poincare-line u v)
  proof (cases u  $\neq$  v)
    case True
      thus ?thesis
        by simp
  next
```

This branch must work only for formal reasons.

```
  case False
    thus ?thesis
      by (transfer, transfer, auto simp add: hermitean-def mat-adj-def mat-cnj-def split: if-split-asm)
qed
qed
```

Next we show how to lift some lemmas. This could be done for all the lemmas that we have proved earlier, but we do not do that.

If points are different then the constructed line contains the starting points

```
lemma p-on-line:
  assumes z  $\neq$  w
  shows p-incident (p-line z w) z
        p-incident (p-line z w) w
  using assms
  by (transfer, simp)+
```

There is a unique h-line passing through the two different given h-points

```
lemma
  assumes u  $\neq$  v
  shows  $\exists! l. \{u, v\} \subseteq p\text{-points } l$ 
  using assms
  apply (rule-tac a=p-line u v in exII, auto simp add: p-points-def p-on-line)
  apply (transfer, simp add: unique-poincare-line)
  done
```

The unique h-line through zero and a non-zero h-point on the x-axis is the x-axis

```
lemma
  assumes p-zero  $\in$  p-points l u  $\in$  p-points l u  $\neq$  p-zero u  $\in$  p-points p-x-axis
  shows l = p-x-axis
  using assms
  unfolding p-points-def
  apply simp
  apply transfer
  using is-poincare-line-0-real-is-x-axis inf-notin-unit-disc
  unfolding circline-set-def
  by blast
```

### 10.3 H-collinearity

**lift-definition**  $p\text{-collinear} :: p\text{-point set} \Rightarrow \text{bool}$  **is** *poincare-collinear*  
**done**

### 10.4 H-isometries

H-isometries are functions that map the unit disc onto itself

**typedef**  $p\text{-isometry} = \{f. \text{unit-disc-fix-}f\}$   
**by** (*rule-tac x=id in exI, simp add: unit-disc-fix-f-def, rule-tac x=id-moebius in exI, simp*)

**setup-lifting** *type-definition-p-isometry*

Action of an h-isometry on an h-point

**lift-definition**  $p\text{-isometry-pt} :: p\text{-isometry} \Rightarrow p\text{-point} \Rightarrow p\text{-point}$  **is**  $\lambda f p. f p$   
**using** *unit-disc-fix-f-unit-disc*  
**by** *auto*

Action of an h-isometry on an h-line

**lift-definition**  $p\text{-isometry-line} :: p\text{-isometry} \Rightarrow p\text{-line} \Rightarrow p\text{-line}$  **is**  $\lambda f l. \text{unit-disc-fix-f-circline } f l$   
**proof**–  
**fix**  $f H$   
**assume** *unit-disc-fix-f is-poincare-line H*  
**then obtain**  $M$  **where** *unit-disc-fix M* **and**  $*$ :  $f = \text{moebius-pt } M \vee f = \text{moebius-pt } M \circ \text{conjugate}$   
**unfolding** *unit-disc-fix-f-def*  
**by** *auto*  
**show** *is-poincare-line (unit-disc-fix-f-circline f H)*  
**using**  $*$   
**proof**  
**assume**  $f = \text{moebius-pt } M$   
**thus** *?thesis*  
**using**  $\langle \text{unit-disc-fix } M \rangle \langle \text{is-poincare-line } H \rangle$   
**using** *unit-disc-fix-f-circline-direct[of M f H]*  
**by** *auto*  
**next**  
**assume**  $f = \text{moebius-pt } M \circ \text{conjugate}$   
**thus** *?thesis*  
**using**  $\langle \text{unit-disc-fix } M \rangle \langle \text{is-poincare-line } H \rangle$   
**using** *unit-disc-fix-f-circline-indirect[of M f H]*  
**by** *auto*  
**qed**  
**qed**

An example lemma about h-isometries.

H-isometries preserve h-collinearity

**lemma**  $p\text{-collinear-p-isometry-pt}$  [*simp*]:  
**shows**  $p\text{-collinear} (p\text{-isometry-pt } M \text{ ' } A) \longleftrightarrow p\text{-collinear } A$   
**proof**–  
**have**  $*$ :  $\forall M A. ((\lambda x. \text{moebius-pt } M (\text{conjugate } x)) \text{ ' } A = \text{moebius-pt } M \text{ ' } (\text{conjugate } \text{ ' } A))$   
**by** *auto*  
**show** *?thesis*  
**by** *transfer (auto simp add: unit-disc-fix-f-def \*)*  
**qed**

### 10.5 H-distance and h-congruence

**lift-definition**  $p\text{-dist} :: p\text{-point} \Rightarrow p\text{-point} \Rightarrow \text{real}$  **is** *poincare-distance*  
**done**

**definition**  $p\text{-congruent} :: p\text{-point} \Rightarrow p\text{-point} \Rightarrow p\text{-point} \Rightarrow p\text{-point} \Rightarrow \text{bool}$  **where**  
[*simp*]:  $p\text{-congruent } u v u' v' \longleftrightarrow p\text{-dist } u v = p\text{-dist } u' v'$

**lemma**

**assumes**  $p\text{-dist } u v = p\text{-dist } u' v'$

```

assumes  $p\text{-dist } v \ w = p\text{-dist } v' \ w'$ 
assumes  $p\text{-dist } u \ w = p\text{-dist } u' \ w'$ 
shows  $\exists f. p\text{-isometry-pt } f \ u = u' \wedge p\text{-isometry-pt } f \ v = v' \wedge p\text{-isometry-pt } f \ w = w'$ 
using assms
apply transfer
using unit-disc-fix-f-congruent-triangles
by auto

```

We prove that unit disc equipped with Poincaré distance is a metric space, i.e. an instantiation of *metric-space* locale.

```

instantiation p-point :: metric-space
begin
definition dist-p-point = p-dist
definition (uniformity-p-point :: (p-point × p-point) filter) = (INF  $e \in \{0 < ..\}$ . principal  $\{(x, y). \text{dist-class.dist } x \ y < e\}$ )
definition open-p-point (U :: p-point set) = ( $\forall x \in U. \text{eventually } (\lambda(x', y). x' = x \longrightarrow y \in U) \text{ uniformity}$ )
instance
proof
  fix  $x \ y :: p\text{-point}$ 
  show (dist-class.dist  $x \ y = 0$ ) = ( $x = y$ )
    unfolding dist-p-point-def
    by (transfer, simp add: poincare-distance-eq-0-iff)
next
  fix  $x \ y \ z :: p\text{-point}$ 
  show dist-class.dist  $x \ y \leq \text{dist-class.dist } x \ z + \text{dist-class.dist } y \ z$ 
    unfolding dist-p-point-def
    apply transfer
    using poincare-distance-triangle-inequality poincare-distance-sym
    by metis
qed (simp-all add: open-p-point-def uniformity-p-point-def)
end

```

## 10.6 H-betweenness

```

lift-definition p-between :: p-point  $\Rightarrow$  p-point  $\Rightarrow$  p-point  $\Rightarrow$  bool is poincare-between
  done

```

**end**

## 11 Poincaré model satisfies Tarski axioms

```

theory Poincare-Tarski
  imports Poincare Poincare-Lines-Axis-Intersections Tarski
begin

```

### 11.1 Pasch axiom

```

lemma Pasch-fun-mono:
  fixes  $r1 \ r2 :: \text{real}$ 
  assumes  $0 < r1$  and  $r1 \leq r2$  and  $r2 < 1$ 
  shows  $r1 + 1/r1 \geq r2 + 1/r2$ 
proof (cases  $r1 = r2$ )
  case True
  thus ?thesis
    by simp
next
  case False
  hence  $r2 - r1 > 0$ 
    using assms
    by simp

  have  $r1 * r2 < 1$ 
    using assms
    by (smt mult-le-cancel-left1)
  hence  $1 / (r1 * r2) > 1$ 

```



```

using assms
by simp
hence  $(r2 - r1) / (r1 * r2) > (r2 - r1)$ 
using  $\langle r2 - r1 > 0 \rangle$ 
using mult-less-cancel-left-pos[of  $r2 - r1$  1 1 /  $(r1 * r2)$ ]
by simp
hence  $1 / r1 - 1 / r2 > r2 - r1$ 
using assms
by (simp add: field-simps)
thus ?thesis
by simp
qed

```

Pasch axiom, non-degenerative case.

**lemma** *Pasch-nondeg*:

```

assumes  $x \in \text{unit-disc}$  and  $y \in \text{unit-disc}$  and  $z \in \text{unit-disc}$  and  $u \in \text{unit-disc}$  and  $v \in \text{unit-disc}$ 
assumes distinct  $[x, y, z, u, v]$ 
assumes  $\neg \text{poincare-collinear } \{x, y, z\}$ 
assumes poincare-between  $x$   $u$   $z$  and poincare-between  $y$   $v$   $z$ 
shows  $\exists a. a \in \text{unit-disc} \wedge \text{poincare-between } u$   $a$   $y \wedge \text{poincare-between } x$   $a$   $v$ 

```

**proof** –

```

have  $\forall y z u. \text{distinct } [x, y, z, u, v] \wedge \neg \text{poincare-collinear } \{x, y, z\} \wedge y \in \text{unit-disc} \wedge z \in \text{unit-disc} \wedge u \in \text{unit-disc} \wedge$ 
 $\text{poincare-between } x$   $u$   $z \wedge \text{poincare-between } y$   $v$   $z \longrightarrow (\exists a. a \in \text{unit-disc} \wedge \text{poincare-between } u$   $a$   $y \wedge$ 
poincare-between  $x$   $a$   $v)$  (is ?P  $x$   $v$ )

```

**proof** (rule *wlog-positive-x-axis*[where  $P = ?P$ ])

**fix**  $v$

**assume**  $v$ : *is-real*  $v$   $0 < \text{Re } v$   $\text{Re } v < 1$

**hence** *of-complex*  $v \in \text{unit-disc}$

**by** (*auto simp add: cmod-eq-Re*)

**show** ?P  $0_h$  (*of-complex*  $v$ )

**proof** *safe*

**fix**  $y z u$

**assume** *distinct*: *distinct*  $[0_h, y, z, u, \text{of-complex } v]$

**assume** *in-disc*:  $y \in \text{unit-disc}$   $z \in \text{unit-disc}$   $u \in \text{unit-disc}$

**then obtain**  $y' z' u'$

**where**  $*$ :  $y = \text{of-complex } y'$   $z = \text{of-complex } z'$   $u = \text{of-complex } u'$

**using** *inf-or-of-complex inf-notin-unit-disc*

**by** *metis*

**have**  $y' \neq 0$   $z' \neq 0$   $u' \neq 0$   $v \neq 0$   $y' \neq z'$   $y' \neq u'$   $z' \neq u'$   $y \neq z$   $y \neq u$   $z \neq u$

**using** *of-complex-inj distinct \**

**by** *auto*

**note** *distinct = distinct this*

**assume**  $\neg \text{poincare-collinear } \{0_h, y, z\}$

**hence** *nondeg-yz*:  $y' * \text{cnj } z' \neq \text{cnj } y' * z'$

**using**  $*$  *poincare-collinear-zero-iff*[of  $y' z'$ ] *in-disc distinct*

**by** *auto*

**assume** *poincare-between*  $0_h$   $u$   $z$

**hence**  $\text{Arg } u' = \text{Arg } z'$  *cmod*  $u' \leq \text{cmod } z'$

**using**  $*$  *poincare-between-0uv*[of  $u z$ ] *distinct in-disc*

**by** *auto*

**then obtain**  $\varphi$   $r_u$   $r_z$  **where**

*uz-polar*:  $u' = \text{cor } r_u * \text{cis } \varphi$   $z' = \text{cor } r_z * \text{cis } \varphi$   $0 < r_u$   $r_u \leq r_z$   $0 < r_z$  **and**

$\varphi = \text{Arg } u'$   $\varphi = \text{Arg } z'$

**using**  $*$   $\langle u' \neq 0 \rangle$   $\langle z' \neq 0 \rangle$

**by** (*smt cmod-cis norm-le-zero-iff*)

**obtain**  $\vartheta$   $r_y$  **where**

*y-polar*:  $y' = \text{cor } r_y * \text{cis } \vartheta$   $r_y > 0$  **and**  $\vartheta = \text{Arg } y'$

**using**  $\langle y' \neq 0 \rangle$

```

by (smt cmod-cis norm-le-zero-iff)

from in-disc * ⟨u' = cor ru * cis φ⟩ ⟨z' = cor rz * cis φ⟩ ⟨y' = cor ry * cis ϑ⟩
have ru < 1 rz < 1 ry < 1
  by (auto simp: norm-mult)

note polar = this y-polar uz-polar

have nondeg: cis ϑ * cis (- φ) ≠ cis (- ϑ) * cis φ
  using nondeg-yz polar
  by simp

let ?yz = poincare-line y z
let ?v = calc-x-axis-intersection ?yz

assume poincare-between y (of-complex v) z

hence of-complex v ∈ circline-set ?yz
  using in-disc ⟨of-complex v ∈ unit-disc⟩
  using distinct poincare-between-poincare-collinear[of y of-complex v z]
  using unique-poincare-line[of y z]
  by (auto simp add: poincare-collinear-def)
moreover
have of-complex v ∈ circline-set x-axis
  using ⟨is-real v⟩
  unfolding circline-set-x-axis
  by auto
moreover
have ?yz ≠ x-axis
proof (rule ccontr)
  assume ¬ ?thesis
  hence {0h, y, z} ⊆ circline-set (poincare-line y z)
    unfolding circline-set-def
    using distinct poincare-line[of y z]
    by auto
  hence poincare-collinear {0h, y, z}
    unfolding poincare-collinear-def
    using distinct
    by force
  thus False
    using ⟨¬ poincare-collinear {0h, y, z}⟩
    by simp
qed
ultimately
have ?v = of-complex v intersects-x-axis ?yz
  using unique-calc-x-axis-intersection[of poincare-line y z of-complex v]
  using intersects-x-axis-iff[of ?yz]
  using distinct ⟨of-complex v ∈ unit-disc⟩
  by (metis IntI is-poincare-line-poincare-line)+

have intersects-x-axis-positive ?yz
  using ⟨Re v > 0⟩ ⟨of-complex v ∈ unit-disc⟩
  using ⟨of-complex v ∈ circline-set ?yz⟩ ⟨of-complex v ∈ circline-set x-axis⟩
  using intersects-x-axis-positive-iff[of ?yz] ⟨y ≠ z⟩ ⟨?yz ≠ x-axis⟩
  unfolding positive-x-axis-def
  by force

have y ∉ circline-set x-axis
proof (rule ccontr)
  assume ¬ ?thesis
  moreover
  hence poincare-line y (of-complex v) = x-axis
    using distinct ⟨of-complex v ∈ circline-set x-axis⟩
    using in-disc ⟨of-complex v ∈ unit-disc⟩
    using unique-poincare-line[of y of-complex v x-axis]
    by simp

```

**moreover**  
**have**  $z \in \text{circline-set } (\text{poincare-line } y \text{ (of-complex } v))$   
**using**  $\langle \text{of-complex } v \in \text{circline-set } ?yz \rangle$   
**using**  $\text{unique-poincare-line}[\text{of } y \text{ of-complex } v \text{ poincare-line } y \ z]$   
**using**  $\text{in-disc } \langle \text{of-complex } v \in \text{unit-disc} \rangle \text{ distinct}$   
**using**  $\text{poincare-line}[\text{of } y \ z]$   
**unfolding**  $\text{circline-set-def}$   
**by**  $(\text{metis distinct-length-2-or-more is-poincare-line-poincare-line mem-Collect-eq})$   
**ultimately**  
**have**  $y \in \text{circline-set } x\text{-axis } z \in \text{circline-set } x\text{-axis}$   
**by**  $\text{auto}$   
**hence**  $\text{poincare-collinear } \{0_h, y, z\}$   
**unfolding**  $\text{poincare-collinear-def}$   
**by**  $\text{force}$   
**thus**  $\text{False}$   
**using**  $\langle \neg \text{poincare-collinear } \{0_h, y, z\} \rangle$   
**by**  $\text{simp}$   
**qed**

**moreover**

**have**  $z \notin \text{circline-set } x\text{-axis}$   
**proof**  $(\text{rule } \text{ccontr})$   
**assume**  $\neg ?thesis$   
**moreover**  
**hence**  $\text{poincare-line } z \text{ (of-complex } v) = x\text{-axis}$   
**using**  $\text{distinct } \langle \text{of-complex } v \in \text{circline-set } x\text{-axis} \rangle$   
**using**  $\text{in-disc } \langle \text{of-complex } v \in \text{unit-disc} \rangle$   
**using**  $\text{unique-poincare-line}[\text{of } z \text{ of-complex } v \ x\text{-axis}]$   
**by**  $\text{simp}$   
**moreover**  
**have**  $y \in \text{circline-set } (\text{poincare-line } z \text{ (of-complex } v))$   
**using**  $\langle \text{of-complex } v \in \text{circline-set } ?yz \rangle$   
**using**  $\text{unique-poincare-line}[\text{of } z \text{ of-complex } v \ \text{poincare-line } y \ z]$   
**using**  $\text{in-disc } \langle \text{of-complex } v \in \text{unit-disc} \rangle \text{ distinct}$   
**using**  $\text{poincare-line}[\text{of } y \ z]$   
**unfolding**  $\text{circline-set-def}$   
**by**  $(\text{metis distinct-length-2-or-more is-poincare-line-poincare-line mem-Collect-eq})$   
**ultimately**  
**have**  $y \in \text{circline-set } x\text{-axis } z \in \text{circline-set } x\text{-axis}$   
**by**  $\text{auto}$   
**hence**  $\text{poincare-collinear } \{0_h, y, z\}$   
**unfolding**  $\text{poincare-collinear-def}$   
**by**  $\text{force}$   
**thus**  $\text{False}$   
**using**  $\langle \neg \text{poincare-collinear } \{0_h, y, z\} \rangle$   
**by**  $\text{simp}$   
**qed**

**ultimately**

**have**  $\varphi * \vartheta < 0$   
**using**  $\langle \text{poincare-between } y \text{ (of-complex } v) \ z \rangle$   
**using**  $\text{poincare-between-x-axis-intersection}[\text{of } y \ z \ \text{of-complex } v]$   
**using**  $\text{in-disc } \langle \text{of-complex } v \in \text{unit-disc} \rangle \text{ distinct}$   
**using**  $\langle \text{of-complex } v \in \text{circline-set } ?yz \rangle \langle \text{of-complex } v \in \text{circline-set } x\text{-axis} \rangle$   
**using**  $\langle \varphi = \text{Arg } z' \rangle \langle \vartheta = \text{Arg } y' \rangle *$   
**by**  $(\text{simp add: field-simps})$

**have**  $\varphi \neq \pi \ \varphi \neq 0$   
**using**  $\langle z \notin \text{circline-set } x\text{-axis} \rangle * \text{polar cis-pi}$   
**unfolding**  $\text{circline-set-x-axis}$   
**by**  $\text{auto}$

**have**  $\vartheta \neq \pi \ \vartheta \neq 0$   
**using**  $\langle y \notin \text{circline-set } x\text{-axis} \rangle * \text{polar cis-pi}$

```

unfolding circline-set-x-axis
by auto

have phi-sin:  $\varphi > 0 \iff \sin \varphi > 0$   $\varphi < 0 \iff \sin \varphi < 0$ 
  using  $\langle \varphi = \text{Arg } z' \rangle \langle \varphi \neq 0 \rangle \langle \varphi \neq \pi \rangle$ 
  using Arg-bounded[of z']
  by (smt sin-gt-zero sin-le-zero sin-pi-minus sin-0-iff-canon sin-ge-zero) +

have theta-sin:  $\vartheta > 0 \iff \sin \vartheta > 0$   $\vartheta < 0 \iff \sin \vartheta < 0$ 
  using  $\langle \vartheta = \text{Arg } y' \rangle \langle \vartheta \neq 0 \rangle \langle \vartheta \neq \pi \rangle$ 
  using Arg-bounded[of y']
  by (smt sin-gt-zero sin-le-zero sin-pi-minus sin-0-iff-canon sin-ge-zero) +

have  $\sin \varphi * \sin \vartheta < 0$ 
  using  $\langle \varphi * \vartheta < 0 \rangle$  phi-sin theta-sin
  by (simp add: mult-less-0-iff)

have  $\sin (\varphi - \vartheta) \neq 0$ 
proof (rule ccontr)
  assume  $\neg ?thesis$ 
  hence  $\sin (\varphi - \vartheta) = 0$ 
    by simp
  have  $-2 * \pi < \varphi - \vartheta$   $\varphi - \vartheta < 2 * \pi$ 
    using  $\langle \varphi = \text{Arg } z' \rangle \langle \vartheta = \text{Arg } y' \rangle$  Arg-bounded[of z'] Arg-bounded[of y']  $\langle \varphi \neq \pi \rangle \langle \vartheta \neq \pi \rangle$ 
    by auto
  hence  $\varphi - \vartheta = -\pi \vee \varphi - \vartheta = 0 \vee \varphi - \vartheta = \pi$ 
    using  $\langle \sin (\varphi - \vartheta) = 0 \rangle$ 
    by (smt sin-0-iff-canon sin-periodic-pi2)
  moreover
  {
    assume  $\varphi - \vartheta = -\pi$ 
    hence  $\varphi = \vartheta - \pi$ 
      by simp
    hence False
      using nondeg-yz
      using  $\langle y' = \text{cor } r y * \text{cis } \vartheta \rangle \langle z' = \text{cor } r z * \text{cis } \varphi \rangle \langle r z > 0 \rangle \langle r y > 0 \rangle$ 
      by auto
  }
  moreover
  {
    assume  $\varphi - \vartheta = 0$ 
    hence  $\varphi = \vartheta$ 
      by simp
    hence False
      using  $\langle y' = \text{cor } r y * \text{cis } \vartheta \rangle \langle z' = \text{cor } r z * \text{cis } \varphi \rangle \langle r z > 0 \rangle \langle r y > 0 \rangle$ 
      using nondeg-yz
      by auto
  }
  moreover
  {
    assume  $\varphi - \vartheta = \pi$ 
    hence  $\varphi = \vartheta + \pi$ 
      by simp
    hence False
      using  $\langle y' = \text{cor } r y * \text{cis } \vartheta \rangle \langle z' = \text{cor } r z * \text{cis } \varphi \rangle \langle r z > 0 \rangle \langle r y > 0 \rangle$ 
      using nondeg-yz
      by auto
  }
  ultimately
  show False
    by auto
qed

have  $u \notin \text{circline-set } x\text{-axis}$ 
proof–
  have  $\neg \text{is-real } u'$ 

```

```

    using * polar in-disc
    using ⟨ $\varphi \neq 0$ ⟩ ⟨ $\varphi = \text{Arg } u'$ ⟩ ⟨ $\varphi \neq \pi$ ⟩ phi-sin(1) phi-sin(2)
    by (metis is-real-arg2)
  moreover
  have  $u \neq \infty_h$ 
    using in-disc
    by auto
  ultimately
  show ?thesis
    using * of-complex-inj[of  $u'$ ]
    unfolding circline-set-x-axis
    by auto
qed

let ?yu = poincare-line  $y u$ 
have nondeg-yu:  $y' * \text{cnj } u' \neq \text{cnj } u' * u'$ 
  using nondeg-yz polar ⟨ $ru > 0$ ⟩ ⟨ $rz > 0$ ⟩ distinct
  by auto

{
  fix  $r :: \text{real}$ 
  assume  $r > 0$ 

  have den:  $\text{cor } ry * \text{cis } \vartheta * \text{cnj } 1 * \text{cnj } (\text{cor } r * \text{cis } \varphi) * 1 - \text{cor } r * \text{cis } \varphi * \text{cnj } 1 * \text{cnj } (\text{cor } ry * \text{cis } \vartheta) * 1 \neq 0$ 
    using ⟨ $0 < r$ ⟩ ⟨ $0 < ry$ ⟩ nondeg
    by auto

  let ?A =  $2 * r * ry * \sin(\varphi - \vartheta)$ 
  let ?B =  $i * (r * \text{cis } \varphi * (1 + ry^2) - ry * \text{cis } \vartheta * (1 + r^2))$ 
  let ?ReB =  $ry * (1 + r^2) * \sin \vartheta - r * (1 + ry^2) * \sin \varphi$ 

  have Re ( $i * (r * \text{cis } (-\varphi) * ry * \text{cis } \vartheta) - ry * \text{cis } (-\vartheta) * r * \text{cis } (\varphi)$ ) = ?A
    by (simp add: sin-diff field-simps)
  moreover
  have  $\text{cor } ry * \text{cis } (-\vartheta) * (\text{cor } ry * \text{cis } \vartheta) = ry^2$   $\text{cor } r * \text{cis } (-\varphi) * (\text{cor } r * \text{cis } \varphi) = r^2$ 
    by (metis cis-inverse cis-neq-zero divide-complex-def cor-squared nonzero-mult-div-cancel-right power2-eq-square
    semiring-normalization-rules(15))+
  ultimately
  have 1: poincare-line-cvec-cmat (of-complex-cvec (cor  $ry * \text{cis } \vartheta$ )) (of-complex-cvec (cor  $r * \text{cis } \varphi$ )) = (?A, ?B,
  cnj ?B, ?A)
    using den
    unfolding poincare-line-cvec-cmat-def of-complex-cvec-def Let-def prod.case
    by (simp add: field-simps)

  have 2: is-real ?A
    by simp
  let ?mix =  $\text{cis } \vartheta * \text{cis } (-\varphi) - \text{cis } (-\vartheta) * \text{cis } \varphi$ 
  have is-imag ?mix
    using eq-minus-cnj-iff-imag[of ?mix]
    by simp
  hence Im ?mix  $\neq 0$ 
    using nondeg
    using complex.expand[of ?mix 0]
    by auto
  hence 3: Re ?A  $\neq 0$ 
    using ⟨ $r > 0$ ⟩ ⟨ $ry > 0$ ⟩
    by (simp add: sin-diff field-simps)

  have ?A  $\neq 0$ 
    using 2 3
    by auto
  hence 4: cor ?A  $\neq 0$ 
    using 2 3
    by (metis zero-complex.simps(1))

```

```

have 5: ?ReB / ?A = (sin ϑ) / (2 * sin(φ - ϑ)) * (1/r + r) - (sin φ) / (2 * sin(φ - ϑ)) * (1/ry + ry)
using ⟨ry > 0⟩ ⟨r > 0⟩
apply (subst diff-divide-distrib)
apply (subst add-frac-num, simp)
apply (subst add-frac-num, simp)
apply (simp add: power2-eq-square mult.commute)
apply (simp add: field-simps)
done

have poincare-line-cvec-cmat (of-complex-cvec (cor ry * cis ϑ)) (of-complex-cvec (cor r * cis φ)) = (?A, ?B, cnj
?B, ?A) ∧
  is-real ?A ∧ Re ?A ≠ 0 ∧ ?A ≠ 0 ∧ cor ?A ≠ 0 ∧
  Re ?B = ?ReB ∧
  ?ReB / ?A = (sin ϑ) / (2 * sin(φ - ϑ)) * (1/r + r) - (sin φ) / (2 * sin(φ - ϑ)) * (1/ry + ry)
using 1 2 3 4 5
by auto
}
note ** = this

let ?Ayz = 2 * rz * ry * sin(φ - ϑ)
let ?Byz = i * (rz * cis φ * (1 + ry2) - ry * cis ϑ * (1 + rz2))
let ?ReByz = ry * (1 + rz2) * sin ϑ - rz * (1 + ry2) * sin φ
let ?Kz = (sin ϑ) / (2 * sin(φ - ϑ)) * (1/rz + rz) - (sin φ) / (2 * sin(φ - ϑ)) * (1/ry + ry)
have yz: poincare-line-cvec-cmat (of-complex-cvec (cor ry * cis ϑ)) (of-complex-cvec (cor rz * cis φ)) = (?Ayz,
?Byz, cnj ?Byz, ?Ayz)
  is-real ?Ayz Re ?Ayz ≠ 0 ?Ayz ≠ 0 cor ?Ayz ≠ 0 Re ?Byz = ?ReByz and Kz: ?ReByz / ?Ayz = ?Kz
using **[OF ⟨0 < rz⟩]
by auto

let ?Ayu = 2 * ru * ry * sin(φ - ϑ)
let ?Byu = i * (ru * cis φ * (1 + ry2) - ry * cis ϑ * (1 + ru2))
let ?ReByu = ry * (1 + ru2) * sin ϑ - ru * (1 + ry2) * sin φ
let ?Ku = (sin ϑ) / (2 * sin(φ - ϑ)) * (1/ru + ru) - (sin φ) / (2 * sin(φ - ϑ)) * (1/ry + ry)
have yu: poincare-line-cvec-cmat (of-complex-cvec (cor ry * cis ϑ)) (of-complex-cvec (cor ru * cis φ)) = (?Ayu,
?Byu, cnj ?Byu, ?Ayu)
  is-real ?Ayu Re ?Ayu ≠ 0 ?Ayu ≠ 0 cor ?Ayu ≠ 0 Re ?Byu = ?ReByu and Ku: ?ReByu / ?Ayu = ?Ku
using **[OF ⟨0 < ru⟩]
by auto

have ?Ayz ≠ 0
using ⟨sin(φ - ϑ) ≠ 0⟩ ⟨ry > 0⟩ ⟨rz > 0⟩
by auto

have Re ?Byz / ?Ayz < -1
using ⟨intersects-x-axis-positive ?yz⟩
  * ⟨y' = cor ry * cis ϑ⟩ ⟨z' = cor rz * cis φ⟩ ⟨u' = cor ru * cis φ⟩
apply simp
apply (transfer fixing: ry rz ru ϑ φ)
apply (transfer fixing: ry rz ru ϑ φ)
proof-
assume intersects-x-axis-positive-cmat (poincare-line-cvec-cmat (of-complex-cvec (cor ry * cis ϑ)) (of-complex-cvec
(cor rz * cis φ)))
  thus (ry * sin ϑ * (1 + rz2) - rz * sin φ * (1 + ry2)) / (2 * rz * ry * sin(φ - ϑ)) < - 1
  using yz
  by simp
qed

have ?ReByz / ?Ayz ≥ ?ReByu / ?Ayu
proof (cases sin φ > 0)
  case True
  hence sin ϑ < 0
  using ⟨sin φ * sin ϑ < 0⟩
  by (smt mult-nonneg-nonneg)

have ?ReByz < 0
proof-

```

```

have  $ry * (1 + rz^2) * \sin \vartheta < 0$ 
  using  $\langle ry > 0 \rangle \langle rz > 0 \rangle$ 
  using  $\langle \sin \vartheta < 0 \rangle$ 
  by (smt mult-pos-neg mult-pos-pos zero-less-power)
moreover
have  $rz * (1 + ry^2) * \sin \varphi > 0$ 
  using  $\langle ry > 0 \rangle \langle rz > 0 \rangle$ 
  using  $\langle \sin \varphi > 0 \rangle$ 
  by (smt mult-pos-neg mult-pos-pos zero-less-power)
ultimately
show ?thesis
  by simp
qed
have  $?Ayz > 0$ 
  using  $\langle Re ?Byz / ?Ayz < -1 \rangle \langle Re ?Byz = ?ReByz \rangle \langle ?ReByz < 0 \rangle$ 
  by (smt divide-less-0-iff)
hence  $\sin(\varphi - \vartheta) > 0$ 
  using  $\langle ry > 0 \rangle \langle rz > 0 \rangle$ 
  by (smt mult-pos-pos zero-less-mult-pos)

have  $1 / ru + ru \geq 1 / rz + rz$ 
  using Pasch-fun-mono[of ru rz]  $\langle 0 < ru \rangle \langle ru \leq rz \rangle \langle rz < 1 \rangle$ 
  by simp
hence  $\sin \vartheta * (1 / ru + ru) \leq \sin \vartheta * (1 / rz + rz)$ 
  using  $\langle \sin \vartheta < 0 \rangle$ 
  by auto
thus ?thesis
  using  $\langle ru > 0 \rangle \langle rz > 0 \rangle \langle ru \leq rz \rangle \langle rz < 1 \rangle \langle ?Ayz > 0 \rangle \langle \sin(\varphi - \vartheta) > 0 \rangle$ 
  using divide-right-mono[of  $\sin \vartheta * (1 / ru + ru)$   $\sin \vartheta * (1 / rz + rz)$   $2 * \sin(\varphi - \vartheta)$ ]
  by (subst Kz, subst Ku) simp
next
assume  $\neg \sin \varphi > 0$ 
hence  $\sin \varphi < 0$ 
  using  $\langle \sin \varphi * \sin \vartheta < 0 \rangle$ 
  by (cases  $\sin \varphi = 0$ , simp-all)
hence  $\sin \vartheta > 0$ 
  using  $\langle \sin \varphi * \sin \vartheta < 0 \rangle$ 
  by (smt mult-nonpos-nonpos)
have  $?ReByz > 0$ 
proof -
  have  $ry * (1 + rz^2) * \sin \vartheta > 0$ 
    using  $\langle ry > 0 \rangle \langle rz > 0 \rangle$ 
    using  $\langle \sin \vartheta > 0 \rangle$ 
    by (smt mult-pos-neg mult-pos-pos zero-less-power)
  moreover
  have  $rz * (1 + ry^2) * \sin \varphi < 0$ 
    using  $\langle ry > 0 \rangle \langle rz > 0 \rangle$ 
    using  $\langle \sin \varphi < 0 \rangle$ 
    by (smt mult-pos-neg mult-pos-pos zero-less-power)
  ultimately
  show ?thesis
    by simp
qed
have  $?Ayz < 0$ 
  using  $\langle Re ?Byz / ?Ayz < -1 \rangle \langle ?Ayz \neq 0 \rangle \langle Re ?Byz = ?ReByz \rangle \langle ?ReByz > 0 \rangle$ 
  by (smt divide-less-0-iff)
hence  $\sin(\varphi - \vartheta) < 0$ 
  using  $\langle ry > 0 \rangle \langle rz > 0 \rangle$ 
  by (smt mult-nonneg-nonneg)

have  $1 / ru + ru \geq 1 / rz + rz$ 
  using Pasch-fun-mono[of ru rz]  $\langle 0 < ru \rangle \langle ru \leq rz \rangle \langle rz < 1 \rangle$ 
  by simp
hence  $\sin \vartheta * (1 / ru + ru) \geq \sin \vartheta * (1 / rz + rz)$ 
  using  $\langle \sin \vartheta > 0 \rangle$ 
  by auto

```

```

thus ?thesis
  using ⟨ru > 0⟩ ⟨rz > 0⟩ ⟨ru ≤ rz⟩ ⟨rz < 1⟩ ⟨?Ayz < 0⟩ ⟨sin (φ - ϑ) < 0⟩
  using divide-right-mono-neg[of sin ϑ * (1 / rz + rz) sin ϑ * (1 / ru + ru) 2 * sin (φ - ϑ)]
  by (subst Kz, subst Ku) simp
qed

have intersects-x-axis-positive ?yu
  using * ⟨y' = cor ry * cis ϑ⟩ ⟨z' = cor rz * cis φ⟩ ⟨u' = cor ru * cis φ⟩
  apply simp
  apply (transfer fixing: ry rz ru ϑ φ)
  apply (transfer fixing: ry rz ru ϑ φ)
proof-
  have Re ?Byu / ?Ayu < -1
    using ⟨Re ?Byz / ?Ayz < -1⟩ ⟨?ReByz / ?Ayz ≥ ?ReByu / ?Ayu⟩
    by (subst (asm) ⟨Re ?Byz = ?ReByz⟩, subst ⟨Re ?Byu = ?ReByu⟩) simp
  thus intersects-x-axis-positive-cmat (poincare-line-cvec-cmat (of-complex-cvec (cor ry * cis ϑ)) (of-complex-cvec
    (cor ru * cis φ)))
    using yu
    by simp
qed

let ?a = calc-x-axis-intersection ?yu
have ?a ∈ positive-x-axis ?a ∈ circline-set ?yu ?a ∈ unit-disc
  using ⟨intersects-x-axis-positive ?yu⟩
  using intersects-x-axis-positive-iff'[of ?yu] ⟨y ≠ u⟩
  by auto

then obtain a' where a': ?a = of-complex a' is-real a' Re a' > 0 Re a' < 1
unfolding positive-x-axis-def circline-set-x-axis
by (auto simp add: cmod-eq-Re)

have intersects-x-axis ?yz intersects-x-axis ?yu
  using ⟨intersects-x-axis-positive ?yz⟩ ⟨intersects-x-axis-positive ?yu⟩
  by auto

show ∃ a. a ∈ unit-disc ∧ poincare-between u a y ∧ poincare-between 0h a (of-complex v)
proof (rule-tac x=?a in exI, safe)
  show poincare-between u ?a y
    using poincare-between-x-axis-intersection[of y u ?a]
    using calc-x-axis-intersection[OF is-poincare-line-poincare-line[OF ⟨y ≠ u⟩] ⟨intersects-x-axis ?yu⟩]
    using calc-x-axis-intersection-in-unit-disc[OF is-poincare-line-poincare-line[OF ⟨y ≠ u⟩] ⟨intersects-x-axis ?yu⟩]
    using in-disc ⟨y ≠ u⟩ ⟨y ∉ circline-set x-axis⟩ ⟨u ∉ circline-set x-axis⟩
    using * ⟨φ = Arg u'⟩ ⟨ϑ = Arg y'⟩ ⟨φ * ϑ < 0⟩
    by (subst poincare-between-rev, auto simp add: mult.commute)
  next
  show poincare-between 0h ?a (of-complex v)
  proof-
    have -?ReByz / ?Ayz ≤ -?ReByu / ?Ayu
      using ⟨?ReByz / ?Ayz ≥ ?ReByu / ?Ayu⟩
      by linarith
    have outward ?yz ?yu
      using * ⟨y' = cor ry * cis ϑ⟩ ⟨z' = cor rz * cis φ⟩ ⟨u' = cor ru * cis φ⟩
      apply simp
      apply (transfer fixing: ry rz ru ϑ φ)
      apply (transfer fixing: ry rz ru ϑ φ)
      apply (subst yz yu)+
      unfolding outward-cmat-def
      apply (simp only: Let-def prod.case)
      apply (subst yz yu)+
      using ⟨-?ReByz / ?Ayz ≤ -?ReByu / ?Ayu⟩
      by simp
    hence Re a' ≤ Re v
      using ⟨?v = of-complex v⟩
      using ⟨?a = of-complex a'⟩
      using ⟨intersects-x-axis-positive ?yz⟩ ⟨intersects-x-axis-positive ?yu⟩
      using outward[OF is-poincare-line-poincare-line[OF ⟨y ≠ z⟩] is-poincare-line-poincare-line[OF ⟨y ≠ u⟩]]

```



```

    by simp
  thus ?thesis
    using ⟨?v = of-complex v⟩
    using poincare-between-x-axis-0uv[of Re a' Re v] a' v
    by simp
qed
next
  show ?a ∈ unit-disc
    by fact
qed
qed
next
  show x ∈ unit-disc v ∈ unit-disc x ≠ v
    using assms
    by auto
next
  fix M x v
  let ?Mx = moebius-pt M x and ?Mv = moebius-pt M v
  assume 1: unit-disc-fix M x ∈ unit-disc v ∈ unit-disc x ≠ v
  assume 2: ?P ?Mx ?Mv
  show ?P x v
  proof safe
    fix y z u
    let ?My = moebius-pt M y and ?Mz = moebius-pt M z and ?Mu = moebius-pt M u
    assume distinct [x, y, z, u, v] ¬ poincare-collinear {x, y, z} y ∈ unit-disc z ∈ unit-disc u ∈ unit-disc
      poincare-between x u z poincare-between y v z
    hence ∃ Ma. Ma ∈ unit-disc ∧ poincare-between ?Mu Ma ?My ∧ poincare-between ?Mx Ma ?Mv
      using 1 2[rule-format, of ?My ?Mz ?Mu]
      by simp
    then obtain Ma where Ma: Ma ∈ unit-disc poincare-between ?Mu Ma ?My ∧ poincare-between ?Mx Ma ?Mv
      by blast
    let ?a = moebius-pt (-M) Ma
    let ?Ma = moebius-pt M ?a
    have ?Ma = Ma
      by (metis moebius-pt-invert uminus-moebius-def)
    hence ?Ma ∈ unit-disc poincare-between ?Mu ?Ma ?My ∧ poincare-between ?Mx ?Ma ?Mv
      using Ma
      by auto
    thus ∃ a. a ∈ unit-disc ∧ poincare-between u a y ∧ poincare-between x a v
      using unit-disc-fix-moebius-inv[OF ⟨unit-disc-fix M⟩ ⟨unit-disc-fix M⟩ ⟨Ma ∈ unit-disc⟩]
      using ⟨u ∈ unit-disc⟩ ⟨v ∈ unit-disc⟩ ⟨x ∈ unit-disc⟩ ⟨y ∈ unit-disc⟩
      by (rule-tac x=?a in exI, simp del: moebius-pt-comp-inv-right)
  qed
qed
thus ?thesis
  using assms
  by auto
qed

```

Pasch axiom, only degenerative cases.

**lemma** *Pasch-deg*:

```

assumes x ∈ unit-disc and y ∈ unit-disc and z ∈ unit-disc and u ∈ unit-disc and v ∈ unit-disc
assumes ¬ distinct [x, y, z, u, v] ∨ poincare-collinear {x, y, z}
assumes poincare-between x u z and poincare-between y v z
shows ∃ a. a ∈ unit-disc ∧ poincare-between u a y ∧ poincare-between x a v
proof (cases poincare-collinear {x, y, z})
  case True
  hence poincare-between x y z ∨ poincare-between y x z ∨ poincare-between y z x
    using assms(1, 2, 3) poincare-collinear3-between poincare-between-rev by blast
  show ?thesis
  proof (cases poincare-between x y z)
    case True
    have poincare-between x y v
      using True assms poincare-between-transitivity
      by (meson poincare-between-rev)
    thus ?thesis
  
```

```

    using assms(2)
    by (rule-tac  $x=y$  in exI, simp)
next
case False
hence poincare-between  $y\ x\ z \vee \textit{poincare-between}\ y\ z\ x$ 
  using  $\langle \textit{poincare-between}\ x\ y\ z \vee \textit{poincare-between}\ y\ x\ z \vee \textit{poincare-between}\ y\ z\ x \rangle$ 
  by simp
show ?thesis
proof(cases poincare-between  $y\ x\ z$ )
  case True
  hence poincare-between  $u\ x\ y$ 
    using assms
    by (meson poincare-between-rev poincare-between-transitivity)
  thus ?thesis
    using assms
    by (rule-tac  $x=x$  in exI, simp)
next
case False
hence poincare-between  $y\ z\ x$ 
  using  $\langle \textit{poincare-between}\ y\ x\ z \vee \textit{poincare-between}\ y\ z\ x \rangle$ 
  by auto
hence poincare-between  $x\ z\ v$ 
  using assms
  by (meson poincare-between-rev poincare-between-transitivity)
hence poincare-between  $x\ u\ v$ 
  using assms poincare-between-transitivity poincare-between-rev
  by (smt poincare-between-sum-distances)
  thus ?thesis
    using assms
    by (rule-tac  $x=u$  in exI, simp)
qed
qed
next
case False
hence  $\neg \textit{distinct}\ [x, y, z, u, v]$ 
  using assms(6) by auto
show ?thesis
proof(cases  $u=z$ )
  case True
  thus ?thesis
    using assms
    apply(rule-tac  $x=v$  in exI)
    by(simp add:poincare-between-rev)
next
case False
hence  $x \neq z$ 
  using assms poincare-between-sandwich by blast
show ?thesis
proof(cases  $v=z$ )
  case True
  thus ?thesis
    using assms
    by (rule-tac  $x=u$  in exI, simp)
next
case False
hence  $y \neq z$ 
  using assms poincare-between-sandwich by blast
show ?thesis
proof(cases  $u = x$ )
  case True
  thus ?thesis
    using assms
    by (rule-tac  $x=x$  in exI, simp)
next
case False
have  $x \neq y$ 

```

```

    using assms <¬ poicare-collinear {x, y, z}>
  by fastforce
have x ≠ v
  using assms <¬ poicare-collinear {x, y, z}>
  by (metis insert-commute poicare-between-poicare-collinear)
have u ≠ y
  using assms <¬ poicare-collinear {x, y, z}>
  using poicare-between-poicare-collinear by blast
have u ≠ v
proof(rule ccontr)
  assume ¬ u ≠ v
  hence poicare-between x v z
    using assms by auto
  hence x ∈ circline-set (poicare-line z v)
    using poicare-between-rev[of x v z]
    using poicare-between-poicare-line-uvw[of z v x]
    using assms <v ≠ z>
    by auto
  have y ∈ circline-set (poicare-line z v)
    using assms <¬ u ≠ v>
    using poicare-between-rev[of y v z]
    using poicare-between-poicare-line-uvw[of z v y]
    using assms <v ≠ z>
    by auto
  have z ∈ circline-set (poicare-line z v)
    using ex-poicare-line-two-points[of z v] <v ≠ z>
    by auto
  have is-poicare-line (poicare-line z v)
    using <v ≠ z>
    by auto
  hence poicare-collinear {x, y, z}
    using <x ∈ circline-set (poicare-line z v)>
    using <y ∈ circline-set (poicare-line z v)>
    using <z ∈ circline-set (poicare-line z v)>
    unfolding poicare-collinear-def
    by (rule-tac x=poicare-line z v in exI, simp)
  thus False
    using <¬ poicare-collinear {x, y, z}> by simp
qed
have v = y
  using <u ≠ v> <u ≠ y> <x ≠ v> <x ≠ y> <u ≠ x> <y ≠ z> <v ≠ z> <x ≠ z> <u ≠ z>
  using <¬ distinct [x, y, z, u, v]>
  by auto
thus ?thesis
  using assms
  by (rule-tac x=y in exI, simp)
qed
qed
qed
qed

```

Axiom of Pasch

lemma Pasch:

```

  assumes x ∈ unit-disc and y ∈ unit-disc and z ∈ unit-disc and u ∈ unit-disc and v ∈ unit-disc
  assumes poicare-between x u z and poicare-between y v z
  shows ∃ a. a ∈ unit-disc ∧ poicare-between u a y ∧ poicare-between x a v
proof(cases distinct [x, y, z, u, v] ∧ ¬ poicare-collinear {x, y, z})
  case True
  thus ?thesis
    using assms Pasch-nondeg by auto
next
  case False
  thus ?thesis
    using assms Pasch-deg by auto
qed

```

## 11.2 Segment construction axiom

lemma *segment-construction*:

assumes  $x \in \text{unit-disc}$  and  $y \in \text{unit-disc}$

assumes  $a \in \text{unit-disc}$  and  $b \in \text{unit-disc}$

shows  $\exists z. z \in \text{unit-disc} \wedge \text{poincare-between } x \ y \ z \wedge \text{poincare-distance } y \ z = \text{poincare-distance } a \ b$

proof –

obtain  $d$  where  $d: d = \text{poincare-distance } a \ b$

by *auto*

have  $d \geq 0$

using *assms*

by (*simp add: poincare-distance-ge0*)

have  $\exists z. z \in \text{unit-disc} \wedge \text{poincare-between } x \ y \ z \wedge \text{poincare-distance } y \ z = d$  (*is ?P x y*)

proof (*cases x = y*)

case *True*

have  $\exists z. z \in \text{unit-disc} \wedge \text{poincare-distance } x \ z = d$

proof (*rule wlog-zero*)

show  $\exists z. z \in \text{unit-disc} \wedge \text{poincare-distance } 0_h \ z = d$

using *ex-x-axis-poincare-distance-negative[of d] <d ≥ 0>*

by *blast*

next

show  $x \in \text{unit-disc}$

by *fact*

next

fix  $a \ u$

assume  $u \in \text{unit-disc} \ \text{cmod } a < 1$

assume  $\exists z. z \in \text{unit-disc} \wedge \text{poincare-distance } (\text{moebius-pt } (\text{blaschke } a) \ u) \ z = d$

then obtain  $z$  where  $z: z \in \text{unit-disc} \ \text{poincare-distance } (\text{moebius-pt } (\text{blaschke } a) \ u) \ z = d$

by *auto*

obtain  $z'$  where  $z': z = \text{moebius-pt } (\text{blaschke } a) \ z' \ z' \in \text{unit-disc}$

using  $\langle z \in \text{unit-disc} \rangle$

using *unit-disc-fix-iff[of blaschke a] <cmod a < 1>*

using *blaschke-unit-disc-fix[of a]*

by *blast*

show  $\exists z. z \in \text{unit-disc} \wedge \text{poincare-distance } u \ z = d$

using  $\ast \ z' \ \langle u : \text{unit-disc} \rangle$

using *blaschke-unit-disc-fix[of a] <cmod a < 1>*

by (*rule-tac x=z' in exI, simp*)

qed

thus *?thesis*

using  $\langle x = y \rangle$

unfolding *poincare-between-def*

by *auto*

next

case *False*

show *?thesis*

proof (*rule wlog-positive-x-axis[where P= $\lambda y \ x. \ ?P \ x \ y$ ]*)

fix  $x$

assume *is-real*  $x \ 0 < \text{Re } x \ \text{Re } x < 1$

then obtain  $z$  where  $z: \text{is-real } z \ \text{Re } z \leq 0 - 1 < \text{Re } z \ \text{of-complex } z \in \text{unit-disc}$

*of-complex*  $z \in \text{unit-disc} \ \text{of-complex } z \in \text{circline-set } x\text{-axis} \ \text{poincare-distance } 0_h \ (\text{of-complex } z) = d$

using *ex-x-axis-poincare-distance-negative[of d] <d ≥ 0>*

by *auto*

have *poincare-between* (*of-complex*  $x$ )  $0_h$  (*of-complex*  $z$ )

proof (*cases z = 0*)

case *True*

thus *?thesis*

unfolding *poincare-between-def*

by *auto*

next

case *False*

have  $x \neq 0$

using  $\langle \text{is-real } x \rangle \ \langle \text{Re } x > 0 \rangle$

```

    by auto
  thus ?thesis
    using poincare-between-x-axis-u0v[of x z]
    using z <is-real x> <x ≠ 0> <Re x > 0> False
    using complex-eq-if-Re-eq mult-pos-neg
    by fastforce
qed
thus ?P (of-complex x) 0h
  using <poincare-distance 0h (of-complex z) = d> <of-complex z ∈ unit-disc>
  by blast
next
  show x ∈ unit-disc y ∈ unit-disc
    by fact+
next
  show y ≠ x using <x ≠ y> by simp
next
  fix M u v
  assume unit-disc-fix M u ∈ unit-disc v ∈ unit-disc u ≠ v
  assume ?P (moebius-pt M v) (moebius-pt M u)
  then obtain z where *: z ∈ unit-disc poincare-between (moebius-pt M v) (moebius-pt M u) z poincare-distance
    (moebius-pt M u) z = d
    by auto
  obtain z' where z': z = moebius-pt M z' z' ∈ unit-disc
    using <z ∈ unit-disc>
    using unit-disc-fix-iff[of M] <unit-disc-fix M>
    by blast
  thus ?P v u
    using * <u ∈ unit-disc> <v ∈ unit-disc> <unit-disc-fix M>
    by auto
qed
qed
thus ?thesis
  using assms d
  by auto
qed

```

### 11.3 Five segment axiom

lemma five-segment-axiom:

```

assumes
  in-disc: x ∈ unit-disc y ∈ unit-disc z ∈ unit-disc u ∈ unit-disc and
  in-disc': x' ∈ unit-disc y' ∈ unit-disc z' ∈ unit-disc u' ∈ unit-disc and
  x ≠ y and
  betw: poincare-between x y z poincare-between x' y' z' and
  xy: poincare-distance x y = poincare-distance x' y' and
  xu: poincare-distance x u = poincare-distance x' u' and
  yu: poincare-distance y u = poincare-distance y' u' and
  yz: poincare-distance y z = poincare-distance y' z'
shows
  poincare-distance z u = poincare-distance z' u'
proof-
  from assms obtain M where
  M: unit-disc-fix-f M M x = x' M u = u' M y = y'
    using unit-disc-fix-f-congruent-triangles[of x y u]
    by blast
  have M z = z'
  proof (rule unique-poincare-distance-on-ray[where u=x' and v=y' and y=M z and z=z' and d=poincare-distance
    x z])
    show 0 ≤ poincare-distance x z
      using poincare-distance-ge0 in-disc
      by simp
  next
    show x' ≠ y'
      using M <x ≠ y>
      using in-disc in-disc' poincare-distance-eq-0-iff xy
      by auto
  qed

```

```

next
  show poincare-distance  $x' (M z) = \textit{poincare-distance } x z$ 
    using M in-disc
    unfolding unit-disc-fix-f-def
    by auto
next
  show  $M z \in \textit{unit-disc}$ 
    using M in-disc
    unfolding unit-disc-fix-f-def
    by auto
next
  show poincare-distance  $x' z' = \textit{poincare-distance } x z$ 
    using xy yz betw
    using poincare-between-sum-distances[of x y z]
    using poincare-between-sum-distances[of x' y' z']
    using in-disc in-disc'
    by auto
next
  show poincare-between  $x' y' (M z)$ 
    using M
    using in-disc betw
    unfolding unit-disc-fix-f-def
    by auto
qed fact+
thus ?thesis
  using  $\langle \textit{unit-disc-fix-f } M \rangle$ 
  using in-disc in-disc'
   $\langle M u = u' \rangle$ 
  unfolding unit-disc-fix-f-def
  by auto
qed

```

## 11.4 Upper dimension axiom

**lemma** *upper-dimension-axiom*:

```

assumes in-disc:  $x \in \textit{unit-disc } y \in \textit{unit-disc } z \in \textit{unit-disc } u \in \textit{unit-disc } v \in \textit{unit-disc}$ 
assumes poincare-distance  $x u = \textit{poincare-distance } x v$ 
  poincare-distance  $y u = \textit{poincare-distance } y v$ 
  poincare-distance  $z u = \textit{poincare-distance } z v$ 
   $u \neq v$ 
shows poincare-between  $x y z \vee \textit{poincare-between } y z x \vee \textit{poincare-between } z x y$ 
proof (cases  $x = y \vee y = z \vee x = z$ )
  case True
  thus ?thesis
    using in-disc
    by auto
next
  case False
  hence  $x \neq y \wedge x \neq z \wedge y \neq z$ 
    by auto
  let  $?cong = \lambda a b a' b'. \textit{poincare-distance } a b = \textit{poincare-distance } a' b'$ 
  have  $\forall z u v. z \in \textit{unit-disc } \wedge u \in \textit{unit-disc } \wedge v \in \textit{unit-disc } \wedge$ 
     $?cong x u x v \wedge ?cong y u y v \wedge ?cong z u z v \wedge u \neq v \longrightarrow$ 
    poincare-collinear  $\{x, y, z\}$  (is  $?P x y$ )
  proof (rule wlog-positive-x-axis[where  $P=?P$ ])
  fix  $x$ 
  assume  $x: \textit{is-real } x \ 0 < \textit{Re } x \ \textit{Re } x < 1$ 
  hence  $x \neq 0$ 
    by auto
  have  $0_h \in \textit{circline-set } x\text{-axis}$ 
    by simp
  show  $?P 0_h$  (of-complex  $x$ )
  proof safe
  fix  $z u v$ 
  assume in-disc:  $z \in \textit{unit-disc } u \in \textit{unit-disc } v \in \textit{unit-disc}$ 
  then obtain  $z' u' v'$  where  $z = \textit{of-complex } z' \ u = \textit{of-complex } u' \ v = \textit{of-complex } v'$ 

```

```

using inf-or-of-complex[of z] inf-or-of-complex[of u] inf-or-of-complex[of v]
by auto

assume cong: ?cong 0h u 0h v ?cong (of-complex x) u (of-complex x) v ?cong z u z v u ≠ v

let ?r0 = poincare-distance 0h u and
    ?rx = poincare-distance (of-complex x) u

have ?r0 > 0 ?rx > 0
  using in-disc cong
  using poincare-distance-eq-0-iff[of 0h u] poincare-distance-ge0[of 0h u]
  using poincare-distance-eq-0-iff[of 0h v] poincare-distance-ge0[of 0h v]
  using poincare-distance-eq-0-iff[of of-complex x u] poincare-distance-ge0[of of-complex x u]
  using poincare-distance-eq-0-iff[of of-complex x v] poincare-distance-ge0[of of-complex x v]
  using x
  by (auto simp add: cmod-eq-Re)

let ?pc0 = poincare-circle 0h ?r0 and
    ?pcx = poincare-circle (of-complex x) ?rx
have u ∈ ?pc0 ∩ ?pcx v ∈ ?pc0 ∩ ?pcx
  using in-disc cong
  by (auto simp add: poincare-circle-def)
hence u = conjugate v
  using intersect-poincare-circles-x-axis[of 0 x ?r0 ?rx u v]
  using x ⟨x ≠ 0⟩ ⟨u ≠ v⟩ ⟨?r0 > 0⟩ ⟨?rx > 0⟩
  by simp

let ?ru = poincare-distance u z
have ?ru > 0
  using poincare-distance-ge0[of u z] in-disc
  using cong
  using poincare-distance-eq-0-iff[of z u] poincare-distance-eq-0-iff[of z v]
  using poincare-distance-eq-0-iff
  by force

have z ∈ poincare-circle u ?ru ∩ poincare-circle v ?ru
  using cong in-disc
  unfolding poincare-circle-def
  by (simp add: poincare-distance-sym)

hence is-real z'
  using intersect-poincare-circles-conjugate-centers[of u v ?ru z] ⟨u = conjugate v⟩ zuv
  using in-disc ⟨u ≠ v⟩ ⟨?ru > 0⟩
  by simp

thus poincare-collinear {0h, of-complex x, z}
  using poincare-line-0-real-is-x-axis[of of-complex x] x ⟨x ≠ 0⟩ zuv ⟨0h ∈ circline-set x-axis⟩
  unfolding poincare-collinear-def
  by (rule-tac x=x-axis in exI, auto simp add: circline-set-x-axis)
qed
next
fix M x y
assume 1: unit-disc-fix M x ∈ unit-disc y ∈ unit-disc x ≠ y
assume 2: ?P (moebius-pt M x) (moebius-pt M y)
show ?P x y
proof safe
  fix z u v
  assume z ∈ unit-disc u ∈ unit-disc v ∈ unit-disc
    ?cong x u x v ?cong y u y v ?cong z u z v u ≠ v
  hence poincare-collinear {moebius-pt M x, moebius-pt M y, moebius-pt M z}
    using 1 2[rule-format, of moebius-pt M z moebius-pt M u moebius-pt M v]
    by simp
  then obtain p where is-poincare-line p {moebius-pt M x, moebius-pt M y, moebius-pt M z} ⊆ circline-set p
    unfolding poincare-collinear-def
    by auto
  thus poincare-collinear {x, y, z}

```

```

    using ⟨unit-disc-fix M⟩
    unfolding poincare-collinear-def
    by (rule-tac x=moebius-circline (-M) p in exI, auto)
  qed
qed fact+

```

```

thus ?thesis
  using assms
  using poincare-collinear3-between[of x y z]
  using poincare-between-rev
  by auto
qed

```

## 11.5 Lower dimension axiom

lemma lower-dimension-axiom:

```

shows  $\exists a \in \text{unit-disc. } \exists b \in \text{unit-disc. } \exists c \in \text{unit-disc.}$ 
       $\neg \text{poincare-between } a \ b \ c \wedge \neg \text{poincare-between } b \ c \ a \wedge \neg \text{poincare-between } c \ a \ b$ 

```

proof-

```

let ?u = of-complex (1/2) and ?v = of-complex (i/2)
have 1:  $0_h \in \text{unit-disc}$  and 2:  $?u \in \text{unit-disc}$  and 3:  $?v \in \text{unit-disc}$ 
  by simp-all

```

```

have *:  $\neg \text{poincare-collinear } \{0_h, ?u, ?v\}$ 

```

proof (rule ccontr)

```

  assume  $\neg ?thesis$ 

```

```

  then obtain p where is-poincare-line p  $\{0_h, ?u, ?v\} \subseteq \text{circline-set } p$ 

```

```

    unfolding poincare-collinear-def

```

```

    by auto

```

moreover

```

have of-complex (1 / 2)  $\neq$  of-complex (i / 2)

```

```

  using of-complex-inj

```

```

  by fastforce

```

ultimately

```

have  $0_h \in \text{circline-set } (\text{poincare-line } ?u \ ?v)$ 

```

```

  using unique-poincare-line[of ?u ?v p]

```

```

  by auto

```

thus False

```

  unfolding circline-set-def

```

```

  by simp (transfer, transfer, simp add: vec-cnj-def)

```

qed

show ?thesis

```

  apply (rule-tac x=0_h in bexI, rule-tac x=?u in bexI, rule-tac x=?v in bexI)

```

```

  apply (rule ccontr, auto)

```

```

  using *

```

```

  using poincare-between-poincare-collinear[OF 1 2 3]

```

```

  using poincare-between-poincare-collinear[OF 2 3 1]

```

```

  using poincare-between-poincare-collinear[OF 3 1 2]

```

```

  by (metis insert-commute)+

```

qed

## 11.6 Negated Euclidean axiom

lemma negated-euclidean-axiom-aux:

```

assumes on-circline H (of-complex (1/2 + i/2)) and is-poincare-line H

```

```

assumes intersects-x-axis-positive H

```

```

shows  $\neg \text{intersects-y-axis-positive } H$ 

```

```

using assms

```

proof (transfer, transfer)

```

  fix H

```

```

  assume hh: hermitean H  $\wedge$  H  $\neq$  mat-zero is-poincare-line-cmat H

```

```

  obtain A B C D where H = (A, B, C, D)

```

```

  by (cases H, auto)

```

```

  hence *: is-real A H = (A, B, cnj B, A) (cmod B)2 > (cmod A)2

```

```

  using hermitean-elems[of A B C D] hh

```

```

  by auto

```



```

assume intersects-x-axis-positive-cmat H
hence  $\text{Re } A \neq 0 \text{ Re } B / \text{Re } A < -1$ 
  using *
  by auto

assume on-circline-cmat-cvec H (of-complex-cvec (1 / 2 + i / 2))
hence  $6*A + 4*Re B + 4*Im B = 0$ 
  using *
  unfolding of-real-mult
  apply (subst Re-express-cnj[of B])
  apply (subst Im-express-cnj[of B])
  apply (simp add: vec-cnj-def)
  apply (simp add: field-simps)
  done
hence  $\text{Re } (6*A + 4*Re B + 4*Im B) = 0$ 
  by simp
hence  $3*Re A + 2*Re B + 2*Im B = 0$ 
  using  $\langle \text{is-real } A \rangle$ 
  by simp

hence  $3/2 + \text{Re } B/\text{Re } A + \text{Im } B/\text{Re } A = 0$ 
  using  $\langle \text{Re } A \neq 0 \rangle$ 
  by (simp add: field-simps)

hence  $-\text{Im } B/\text{Re } A - 3/2 < -1$ 
  using  $\langle \text{Re } B / \text{Re } A < -1 \rangle$ 
  by simp
hence  $\text{Im } B/\text{Re } A > -1/2$ 
  by (simp add: field-simps)
thus  $\neg$  intersects-y-axis-positive-cmat H
  using *
  by simp
qed

```

**lemma** *negated-euclidean-axiom:*

**shows**  $\exists a b c d t.$

$a \in \text{unit-disc} \wedge b \in \text{unit-disc} \wedge c \in \text{unit-disc} \wedge d \in \text{unit-disc} \wedge t \in \text{unit-disc} \wedge$   
 $\text{poincare-between } a d t \wedge \text{poincare-between } b d c \wedge a \neq d \wedge$

$(\forall x y. x \in \text{unit-disc} \wedge y \in \text{unit-disc} \wedge$

$\text{poincare-between } a b x \wedge \text{poincare-between } x t y \longrightarrow \neg \text{poincare-between } a c y)$

**proof**–

**let**  $?a = 0_h$

**let**  $?b = \text{of-complex } (1/2)$

**let**  $?c = \text{of-complex } (i/2)$

**let**  $?dl = (5 - \text{sqrt } 17) / 4$

**let**  $?d = \text{of-complex } (?dl + i*?dl)$

**let**  $?t = \text{of-complex } (1/2 + i/2)$

**have**  $?dl \neq 0$

**proof**–

**have**  $(\text{sqrt } 17)^2 \neq 5^2$

**by** *simp*

**hence**  $\text{sqrt } 17 \neq 5$

**by** *force*

**thus**  $?thesis$

**by** *simp*

**qed**

**have**  $?d \neq ?a$

**proof** (*rule ccontr*)

**assume**  $\neg ?thesis$

**hence**  $?dl + i*?dl = 0$

**by** *simp*

**hence**  $\text{Re } (?dl + i*?dl) = 0$

**by** *simp*

**thus** *False*

```

    using ⟨?dl ≠ 0⟩
    by simp
qed

have ?dl > 0
proof-
  have (sqrt 17)2 < 52
    by (simp add: power2-eq-square)
  hence sqrt 17 < 5
    by (rule power2-less-imp-less, simp)
  thus ?thesis
    by simp
qed

have ?a ≠ ?b
  by (metis divide-eq-0-iff of-complex-zero-iff zero-neq-numeral zero-neq-one)

have ?a ≠ ?c
  by (metis complex-i-not-zero divide-eq-0-iff of-complex-zero-iff zero-neq-numeral)

show ?thesis
proof (rule-tac x=?a in exI, rule-tac x=?b in exI, rule-tac x=?c in exI, rule-tac x=?d in exI, rule-tac x=?t in exI,
safe)

  show ?a ∈ unit-disc ?b ∈ unit-disc ?c ∈ unit-disc ?t ∈ unit-disc
    by (auto simp add: cmod-def power2-eq-square)

  have cmod-d: cmod (?dl + i*?dl) = ?dl * sqrt 2
    using ⟨?dl > 0⟩
    unfolding cmod-def
    by (simp add: real-sqrt-mult)

  show ?d ∈ unit-disc
  proof-
    have ?dl < 1 / sqrt 2
    proof-
      have 172 < (5 * sqrt 17)2
        by (simp add: field-simps)
      hence 17 < 5 * sqrt 17
        by (rule power2-less-imp-less, simp)
      hence ?dl2 < (1 / sqrt 2)2
        by (simp add: power2-eq-square field-simps)
      thus ?dl < 1 / sqrt 2
        by (rule power2-less-imp-less, simp)
    qed
    thus ?thesis
      using cmod-d
      by (simp add: field-simps)
  qed

  have cmod-d: 1 - (cmod (to-complex ?d))2 = (-17 + 5*sqrt 17) / 4 (is - = ?cmod-d)
    apply (simp only: to-complex-of-complex)
    apply (subst cmod-d)
    apply (simp add: power-mult-distrib)
    apply (simp add: power2-eq-square field-simps)
    done

  have cmod-d-c: (cmod (to-complex ?d - to-complex ?c))2 = (17 - 4*sqrt 17) / 4 (is - = ?cmod-dc)
    unfolding cmod-square
    by (simp add: field-simps)

  have cmod-c: 1 - (cmod (to-complex ?c))2 = 3/4 (is - = ?cmod-c)
    by (simp add: power2-eq-square)

  have xx: ∧ x::real. x + x = 2*x

```

```

by simp

have cmod ((to-complex ?b) - (to-complex ?d)) = cmod ((to-complex ?d) - (to-complex ?c))
  by (simp add: cmod-def power2-eq-square field-simps)
moreover
have cmod (to-complex ?b) = cmod (to-complex ?c)
  by simp
ultimately
have *: poincare-distance-formula' (to-complex ?b) (to-complex ?d) =
  poincare-distance-formula' (to-complex ?d) (to-complex ?c)
  unfolding poincare-distance-formula'-def
  by simp

have **: poincare-distance-formula' (to-complex ?d) (to-complex ?c) = (sqrt 17) / 3
  unfolding poincare-distance-formula'-def
proof (subst cmod-d, subst cmod-c, subst cmod-d-c)
  have (sqrt 17 * 15)2 ≠ 512
    by simp
  hence sqrt 17 * 15 ≠ 51
    by force
  hence sqrt 17 * 15 - 51 ≠ 0
    by simp

  have (5 * sqrt 17)2 ≠ 172
    by simp
  hence 5 * sqrt 17 ≠ 17
    by force
  hence ?cmod-d * ?cmod-c ≠ 0
    by simp
  hence 1 + 2 * (?cmod-dc / (?cmod-d * ?cmod-c)) = (?cmod-d * ?cmod-c + 2 * ?cmod-dc) / (?cmod-d * ?cmod-c)
    using add-frac-num[of ?cmod-d * ?cmod-c 2 * ?cmod-dc 1]
    by (simp add: field-simps)
  also have ... = (64 * (85 - sqrt 17 * 17)) / (64 * (sqrt 17 * 15 - 51))
    by (simp add: field-simps)
  also have ... = (85 - sqrt 17 * 17) / (sqrt 17 * 15 - 51)
    by (rule mult-divide-mult-cancel-left, simp)
  also have ... = sqrt 17 / 3
    by (subst frac-eq-eq, fact, simp, simp add: field-simps)
  finally
  show 1 + 2 * (?cmod-dc / (?cmod-d * ?cmod-c)) = sqrt 17 / 3
  .
qed

have sqrt 17 ≥ 3
proof-
  have (sqrt 17)2 ≥ 32
    by simp
  thus ?thesis
    by (rule power2-le-imp-le, simp)
qed
thus poincare-between ?b ?d ?c
  unfolding poincare-between-sum-distances[OF ‹?b ∈ unit-disc› ‹?d ∈ unit-disc› ‹?c ∈ unit-disc›]
  unfolding poincare-distance-formula[OF ‹?b ∈ unit-disc› ‹?d ∈ unit-disc›]
  unfolding poincare-distance-formula[OF ‹?d ∈ unit-disc› ‹?c ∈ unit-disc›]
  unfolding poincare-distance-formula[OF ‹?b ∈ unit-disc› ‹?c ∈ unit-disc›]
  unfolding poincare-distance-formula-def
  apply (subst *, subst xx, subst **, subst arcosh-double)
  apply (simp-all add: cmod-def power2-eq-square)
  done

show poincare-between ?a ?d ?t
proof (subst poincare-between-0uv[OF ‹?d ∈ unit-disc› ‹?t ∈ unit-disc› ‹?d ≠ ?a›])
  show ?t ≠ 0h
  proof (rule ccontr)
    assume ¬ ?thesis
    hence 1/2 + i/2 = 0

```

```

    by simp
  hence Re (1/2 + i/2) = 0
    by simp
  thus False
    by simp
qed
next
have 192 ≤ (5 * sqrt 17)2
  by simp
hence 19 ≤ 5 * sqrt 17
  by (rule power2-le-imp-le, simp)
hence cmod (to-complex ?d) ≤ cmod (to-complex ?t)
  by (simp add: Let-def cmod-def power2-eq-square field-simps)
moreover
have Arg (to-complex ?d) = Arg (to-complex ?t)
proof-
  have 1: to-complex ?d = ((5 - sqrt 17) / 4) * (1 + i)
    by (simp add: field-simps)

  have 2: to-complex ?t = (cor (1/2)) * (1 + i)
    by (simp add: field-simps)

  have (sqrt 17)2 < 52
    by simp
  hence sqrt 17 < 5
    by (rule power2-less-imp-less, simp)
  hence 3: (5 - sqrt 17) / 4 > 0
    by simp

  have 4: (1::real) / 2 > 0
    by simp

  show ?thesis
    apply (subst 1, subst 2)
    apply (subst arg-mult-real-positive[OF 3])
    apply (subst arg-mult-real-positive[OF 4])
    by simp
qed
ultimately
show let d' = to-complex ?d; t' = to-complex ?t in Arg d' = Arg t' ∧ cmod d' ≤ cmod t'
  by simp
qed

show ?a = ?d ⇒ False
  using ⟨?d ≠ ?a⟩
  by simp

fix x y
assume x ∈ unit-disc y ∈ unit-disc

assume abx: poincare-between ?a ?b x
hence x ∈ circline-set x-axis
  using poincare-between-poincare-line-wvz[of ?a ?b x] ⟨x ∈ unit-disc⟩ ⟨?a ≠ ?b⟩
  using poincare-line-0-real-is-x-axis[of ?b]
  by (auto simp add: circline-set-x-axis)

have x ≠ 0h
  using abx poincare-between-sandwich[of ?a ?b] ⟨?a ≠ ?b⟩
  by auto

have x ∈ positive-x-axis
  using ⟨x ∈ circline-set x-axis⟩ ⟨x ≠ 0h⟩ ⟨x ∈ unit-disc⟩
  using abx poincare-between-x-axis-0uv[of 1/2 Re (to-complex x)]
  unfolding circline-set-x-axis positive-x-axis-def
  by (auto simp add: cmod-eq-Re abs-less-iff complex-eq-if-Re-eq)

```

```

assume acy: poincare-between ?a ?c y
hence  $y \in \text{circline-set } y\text{-axis}$ 
  using poincare-between-poincare-line-wvz[of ?a ?c y]  $\langle y \in \text{unit-disc} \rangle \langle ?a \neq ?c \rangle$ 
  using poincare-line-0-imag-is-y-axis[of ?c]
  by (auto simp add: circline-set-y-axis)

have  $y \neq 0_h$ 
  using acy poincare-between-sandwich[of ?a ?c]  $\langle ?a \neq ?c \rangle$ 
  by auto

have  $y \in \text{positive-y-axis}$ 
proof–
  have  $\bigwedge x. \llbracket x \neq 0; \text{poincare-between } 0_h \text{ (of-complex } (i / 2)) \text{ (of-complex } x); \text{is-imag } x; -1 < \text{Im } x \rrbracket \implies 0 < \text{Im } x$ 
  by (smt add.left-neutral complex.expand divide-complex-def complex-eq divide-less-0-1-iff divide-less-eq-1-pos imaginary-unit.simps(1) mult.left-neutral of-real-1 of-real-add of-real-divide of-real-eq-0-iff one-add-one poincare-between-y-axis-0uv zero-complex.simps(1) zero-complex.simps(2) zero-less-divide-1-iff)
  thus ?thesis
    using  $\langle y \in \text{circline-set } y\text{-axis} \rangle \langle y \neq 0_h \rangle \langle y \in \text{unit-disc} \rangle$ 
    using acy
    unfolding circline-set-y-axis positive-y-axis-def
    by (auto simp add: cmod-eq-Im abs-less-iff)
qed

have  $x \neq y$ 
  using  $\langle x \in \text{positive-x-axis} \rangle \langle y \in \text{positive-y-axis} \rangle$ 
  unfolding positive-x-axis-def positive-y-axis-def circline-set-x-axis circline-set-y-axis
  by auto

assume xy: poincare-between x ?t y

let ?xy = poincare-line x y

have ?t  $\in \text{circline-set } ?xy$ 
  using xy poincare-between-poincare-line-uzv[OF  $\langle x \neq y \rangle \langle x \in \text{unit-disc} \rangle \langle y \in \text{unit-disc} \rangle \langle ?t \in \text{unit-disc} \rangle$ ]
  by simp

moreover

have ?xy  $\neq x\text{-axis}$ 
  using poincare-line-circline-set[OF  $\langle x \neq y \rangle$ ]  $\langle y \in \text{positive-y-axis} \rangle$ 
  by (auto simp add: circline-set-x-axis positive-y-axis-def)
hence intersects-x-axis-positive ?xy
  using intersects-x-axis-positive-iff[of ?xy]  $\langle x \neq y \rangle \langle x \in \text{unit-disc} \rangle \langle x \in \text{positive-x-axis} \rangle$ 
  by auto

moreover

have ?xy  $\neq y\text{-axis}$ 
  using poincare-line-circline-set[OF  $\langle x \neq y \rangle$ ]  $\langle x \in \text{positive-x-axis} \rangle$ 
  by (auto simp add: circline-set-y-axis positive-x-axis-def)
hence intersects-y-axis-positive ?xy
  using intersects-y-axis-positive-iff[of ?xy]  $\langle x \neq y \rangle \langle y \in \text{unit-disc} \rangle \langle y \in \text{positive-y-axis} \rangle$ 
  by auto

ultimately

show False
  using negated-euclidean-axiom-aux[of ?xy]  $\langle x \neq y \rangle$ 
  unfolding circline-set-def
  by auto
qed
qed

```

Alternate form of the Euclidean axiom – this one is much easier to prove

**lemma** *negated-euclidean-axiom'*:  
**shows**  $\exists a b c.$

$a \in \text{unit-disc} \wedge b \in \text{unit-disc} \wedge c \in \text{unit-disc} \wedge \neg(\text{poincare-collinear } \{a, b, c\}) \wedge$   
 $\neg(\exists x. x \in \text{unit-disc} \wedge$   
 $\text{poincare-distance } a \ x = \text{poincare-distance } b \ x \wedge$   
 $\text{poincare-distance } a \ x = \text{poincare-distance } c \ x)$

**proof-**

**let**  $?a = \text{of-complex } (i/2)$   
**let**  $?b = \text{of-complex } (-i/2)$   
**let**  $?c = \text{of-complex } (1/5)$

**have**  $(i/2) \neq (-i/2)$   
**by** *simp*  
**hence**  $?a \neq ?b$   
**by** *(metis to-complex-of-complex)*  
**have**  $(i/2) \neq (1/5)$   
**by** *simp*  
**hence**  $?a \neq ?c$   
**by** *(metis to-complex-of-complex)*  
**have**  $(-i/2) \neq (1/5)$   
**by** *(simp add: minus-equation-iff)*  
**hence**  $?b \neq ?c$   
**by** *(metis to-complex-of-complex)*

**have**  $?a \in \text{unit-disc } ?b \in \text{unit-disc } ?c \in \text{unit-disc}$   
**by** *auto*

**moreover**

**have**  $\neg(\text{poincare-collinear } \{?a, ?b, ?c\})$   
**unfolding** *poincare-collinear-def*

**proof**(*rule ccontr*)

**assume**  $\neg(\exists p. \text{is-poincare-line } p \wedge \{?a, ?b, ?c\} \subseteq \text{circline-set } p)$   
**then obtain**  $p$  **where**  $\text{is-poincare-line } p \wedge \{?a, ?b, ?c\} \subseteq \text{circline-set } p$   
**by** *auto*

**let**  $?ab = \text{poincare-line } ?a \ ?b$

**have**  $p = ?ab$

**using**  $\langle \text{is-poincare-line } p \wedge \{?a, ?b, ?c\} \subseteq \text{circline-set } p \rangle$   
**using**  $\text{unique-poincare-line}[of \ ?a \ ?b] \langle ?a \neq ?b \rangle \langle ?a \in \text{unit-disc} \rangle \langle ?b \in \text{unit-disc} \rangle$   
**by** *auto*

**have**  $?c \notin \text{circline-set } ?ab$

**proof**(*rule ccontr*)

**assume**  $\neg ?c \notin \text{circline-set } ?ab$

**have**  $\text{poincare-between } ?a \ 0_h \ ?b$

**unfolding** *poincare-between-def*

**using** *cross-ratio-0inf* **by** *auto*

**hence**  $0_h \in \text{circline-set } ?ab$

**using**  $\langle ?a \neq ?b \rangle \langle ?a \in \text{unit-disc} \rangle \langle ?b \in \text{unit-disc} \rangle$

**using** *poincare-between-poincare-line-uzv zero-in-unit-disc*

**by** *blast*

**hence**  $?ab = \text{poincare-line } 0_h \ ?a$

**using**  $\text{unique-poincare-line}[of \ ?a \ ?b] \langle ?a \neq ?b \rangle \langle ?a \in \text{unit-disc} \rangle \langle ?b \in \text{unit-disc} \rangle$

**using**  $\langle \text{is-poincare-line } p \wedge \{?a, ?b, ?c\} \subseteq \text{circline-set } p \rangle$

**using**  $\langle p = ?ab \rangle \text{poincare-line-circline-set}(1) \text{unique-poincare-line}$

**by** *(metis add.inverse-neutral divide-minus-left of-complex-zero-iff zero-in-unit-disc)*

**hence**  $(i/2) * \text{cnj}(1/5) = \text{cnj}(i/2) * (1/5)$

**using**  $\text{poincare-collinear-zero-iff}[of \ (i/2) \ (1/5)]$

**using**  $\langle ?a \neq ?c \rangle \langle \neg ?c \notin \text{circline-set } ?ab \rangle \langle ?a \in \text{unit-disc} \rangle \langle ?c \in \text{unit-disc} \rangle \langle p = ?ab \rangle$

**using**  $\langle 0_h \in \text{circline-set } ?ab \rangle \langle \text{is-poincare-line } p \wedge \{?a, ?b, ?c\} \subseteq \text{circline-set } p \rangle$

**using** *poincare-collinear-def* **by** *auto*

**thus** *False*

**by** *simp*

**qed**

**thus** *False*

**using**  $\langle p = ?ab \rangle \langle \text{is-poincare-line } p \wedge \{?a, ?b, ?c\} \subseteq \text{circline-set } p \rangle$

**by** *auto*

**qed**

**moreover**

```

have  $\neg(\exists x. x \in \text{unit-disc} \wedge$ 
   $\text{poincare-distance } ?a x = \text{poincare-distance } ?b x \wedge$ 
   $\text{poincare-distance } ?a x = \text{poincare-distance } ?c x)$ 
proof(rule ccontr)
  assume  $\neg ?thesis$ 
  then obtain  $x$  where  $x \in \text{unit-disc}$   $\text{poincare-distance } ?a x = \text{poincare-distance } ?b x$ 
     $\text{poincare-distance } ?a x = \text{poincare-distance } ?c x$ 
    by blast
  let  $?x = \text{to-complex } x$ 
  have  $\text{poincare-distance-formula}' (i/2) ?x = \text{poincare-distance-formula}' (-i/2) ?x$ 
    using  $\langle \text{poincare-distance } ?a x = \text{poincare-distance } ?b x \rangle$ 
    using  $\langle x \in \text{unit-disc} \rangle \langle ?a \in \text{unit-disc} \rangle \langle ?b \in \text{unit-disc} \rangle$ 
    by (metis cosh-dist to-complex-of-complex)
  hence  $(\text{cmod } (i / 2 - ?x))^2 = (\text{cmod } (- i / 2 - ?x))^2$ 
    unfolding  $\text{poincare-distance-formula}'\text{-def}$ 
    apply (simp add:field-simps)
    using  $\langle x \in \text{unit-disc} \rangle \text{unit-disc-cmod-square-lt-1}$  by fastforce
  hence  $\text{Im } ?x = 0$ 
    unfolding  $\text{cmod-def}$ 
    by (simp add: power2-eq-iff)

  have  $1 - (\text{Re } ?x)^2 \neq 0$ 
    using  $\langle x \in \text{unit-disc} \rangle \text{unit-disc-cmod-square-lt-1}$ 
    using  $\text{cmod-power2}$  by force
  hence  $24 - 24 * (\text{Re } ?x)^2 \neq 0$ 
    by simp
  have  $\text{poincare-distance-formula}' (i/2) ?x = \text{poincare-distance-formula}' (1/5) ?x$ 
    using  $\langle \text{poincare-distance } ?a x = \text{poincare-distance } ?c x \rangle$ 
    using  $\langle x \in \text{unit-disc} \rangle \langle ?a \in \text{unit-disc} \rangle \langle ?c \in \text{unit-disc} \rangle$ 
    by (metis cosh-dist to-complex-of-complex)
  hence  $(2 + 8 * (\text{Re } ?x)^2) / (3 - 3 * (\text{Re } ?x)^2) = 2 * (1 - \text{Re } ?x * 5)^2 / (24 - 24 * (\text{Re } ?x)^2)$  (is ?lhs = ?rhs)
    unfolding  $\text{poincare-distance-formula}'\text{-def}$ 
    apply (simp add:field-simps)
    unfolding  $\text{cmod-def}$ 
    using  $\langle \text{Im } ?x = 0 \rangle$ 
    by (simp add:field-simps)
  hence  $*: ?lhs * (24 - 24 * (\text{Re } ?x)^2) = ?rhs * (24 - 24 * (\text{Re } ?x)^2)$ 
    using  $\langle (24 - 24 * (\text{Re } ?x)^2) \neq 0 \rangle$ 
    by simp
  have  $?lhs * (24 - 24 * (\text{Re } ?x)^2) = (2 + 8 * (\text{Re } ?x)^2) * 8$ 
    using  $\langle (24 - 24 * (\text{Re } ?x)^2) \neq 0 \rangle \langle 1 - (\text{Re } ?x)^2 \neq 0 \rangle$ 
    by (simp add:field-simps)
  have  $?rhs * (24 - 24 * (\text{Re } ?x)^2) = 2 * (1 - \text{Re } ?x * 5)^2$ 
    using  $\langle (24 - 24 * (\text{Re } ?x)^2) \neq 0 \rangle \langle 1 - (\text{Re } ?x)^2 \neq 0 \rangle$ 
    by (simp add:field-simps)
  hence  $(2 + 8 * (\text{Re } ?x)^2) * 8 = 2 * (1 - \text{Re } ?x * 5)^2$ 
    using  $* \langle ?lhs * (24 - 24 * (\text{Re } ?x)^2) = (2 + 8 * (\text{Re } ?x)^2) * 8 \rangle$ 
    by simp
  hence  $7 * (\text{Re } ?x)^2 + 10 * (\text{Re } ?x) + 7 = 0$ 
    by (simp add:field-simps comm-ring-1-class.power2-diff)
  thus False
    using  $\text{discriminant-iff}[of 7 \text{Re } (\text{to-complex } x) 10 7]$   $\text{discrim-def}[of 7 10 7]$ 
    by auto

```

qed

```

ultimately show ?thesis
  apply (rule-tac  $x=?a$  in exI)
  apply (rule-tac  $x=?b$  in exI)
  apply (rule-tac  $x=?c$  in exI)
  by auto

```

qed

## 11.7 Continuity axiom

The set  $\phi$  is on the left of the set  $\psi$

**abbreviation** *set-order* **where**

*set-order*  $A \varphi \psi \equiv \forall x \in \text{unit-disc. } \forall y \in \text{unit-disc. } \varphi x \wedge \psi y \longrightarrow \text{poincare-between } A x y$

The point  $B$  is between the sets  $\phi$  and  $\psi$

**abbreviation** *point-between-sets* **where**

*point-between-sets*  $\varphi B \psi \equiv \forall x \in \text{unit-disc. } \forall y \in \text{unit-disc. } \varphi x \wedge \psi y \longrightarrow \text{poincare-between } x B y$

**lemma** *continuity*:

**assumes**  $\exists A \in \text{unit-disc. } \text{set-order } A \varphi \psi$

**shows**  $\exists B \in \text{unit-disc. } \text{point-between-sets } \varphi B \psi$

**proof** (*cases*  $(\exists x0 \in \text{unit-disc. } \varphi x0) \wedge (\exists y0 \in \text{unit-disc. } \psi y0)$ )

**case** *False*

**thus** *?thesis*

**using** *assms* **by** *blast*

**next**

**case** *True*

**then obtain**  $Y0$  **where**  $\psi Y0 Y0 \in \text{unit-disc}$

**by** *auto*

**obtain**  $A$  **where**  $*: A \in \text{unit-disc } \text{set-order } A \varphi \psi$

**using** *assms*

**by** *auto*

**show** *?thesis*

**proof**(*cases*  $\forall x \in \text{unit-disc. } \varphi x \longrightarrow x = A$ )

**case** *True*

**thus** *?thesis*

**using**  $\langle A \in \text{unit-disc} \rangle$

**using** *poincare-between-nonstrict(1)* **by** *blast*

**next**

**case** *False*

**then obtain**  $X0$  **where**  $\varphi X0 X0 \neq A X0 \in \text{unit-disc}$

**by** *auto*

**have**  $Y0 \neq A$

**proof**(*rule ccontr*)

**assume**  $\neg Y0 \neq A$

**hence**  $\forall x \in \text{unit-disc. } \varphi x \longrightarrow \text{poincare-between } A x A$

**using**  $* \langle \psi Y0 \rangle$

**by** (*cases*  $A$ ) *force*

**hence**  $\forall x \in \text{unit-disc. } \varphi x \longrightarrow x = A$

**using**  $* \text{poincare-between-sandwich}$  **by** *blast*

**thus** *False*

**using** *False* **by** *auto*

**qed**

**show** *?thesis*

**proof** (*cases*  $\exists B \in \text{unit-disc. } \varphi B \wedge \psi B$ )

**case** *True*

**then obtain**  $B$  **where**  $B \in \text{unit-disc } \varphi B \psi B$

**by** *auto*

**hence**  $\forall x \in \text{unit-disc. } \varphi x \longrightarrow \text{poincare-between } A x B$

**using**  $* \text{by auto}$

**have**  $\forall y \in \text{unit-disc. } \psi y \longrightarrow \text{poincare-between } A B y$

**using**  $* \langle B \in \text{unit-disc} \rangle \langle \varphi B \rangle$

**by** *auto*

**show** *?thesis*

**proof**(*rule+*)

**show**  $B \in \text{unit-disc}$

**by** *fact*

**next**

**fix**  $x y$

**assume**  $x \in \text{unit-disc } y \in \text{unit-disc } \varphi x \wedge \psi y$

**hence** *poincare-between*  $A x B$  *poincare-between*  $A B y$

**using**  $\langle \forall x \in \text{unit-disc. } \varphi x \longrightarrow \text{poincare-between } A x B \rangle$

**using**  $\langle \forall y \in \text{unit-disc. } \psi y \longrightarrow \text{poincare-between } A B y \rangle$

**by** *simp+*

**thus** *poincare-between*  $x B y$



```

    using ⟨x ∈ unit-disc⟩ ⟨y ∈ unit-disc⟩ ⟨B ∈ unit-disc⟩ ⟨A ∈ unit-disc⟩
    using poincare-between-transitivity[of A x B y]
    by simp
qed
next
case False
have poincare-between A X0 Y0
  using ⟨φ X0⟩ ⟨ψ Y0⟩ * ⟨Y0 ∈ unit-disc⟩ ⟨X0 ∈ unit-disc⟩
  by auto
have ∀ φ. ∀ ψ. set-order A φ ψ ∧ ¬ (∃ B ∈ unit-disc. φ B ∧ ψ B) ∧ φ X0 ∧
  (∃ y ∈ unit-disc. ψ y) ∧ (∃ x ∈ unit-disc. φ x)
  → (∃ B ∈ unit-disc. point-between-sets φ B ψ)
  (is ?P A X0)
proof (rule wlog-positive-x-axis[where P=?P])
  show A ∈ unit-disc
  by fact
next
show X0 ∈ unit-disc
  by fact
next
show A ≠ X0
  using ⟨X0 ≠ A⟩ by simp
next
fix M u v
let ?M = λ x. moebius-pt M x
let ?Mu = ?M u and ?Mv = ?M v
assume hip: unit-disc-fix M u ∈ unit-disc v ∈ unit-disc u ≠ v
  ?P ?Mu ?Mv
show ?P u v
proof safe
  fix φ ψ x y
  assume set-order u φ ψ ∧ ¬ (∃ B ∈ unit-disc. φ B ∧ ψ B) φ v
  y ∈ unit-disc ψ y x ∈ unit-disc φ x

  let ?Mφ = λ X'. ∃ X. φ X ∧ ?M X = X'
  let ?Mψ = λ X'. ∃ X. ψ X ∧ ?M X = X'

  obtain Mφ where Mφ = ?Mφ by simp
  obtain Mψ where Mψ = ?Mψ by simp

  have Mφ ?Mv
    using ⟨φ v⟩ using ⟨Mφ = ?Mφ⟩
    by blast
  moreover
  have ¬ (∃ B ∈ unit-disc. Mφ B ∧ Mψ B)
    using ⟨¬ (∃ B ∈ unit-disc. φ B ∧ ψ B)⟩
    using ⟨Mφ = ?Mφ⟩ ⟨Mψ = ?Mψ⟩
    by (metis hip(1) moebius-pt-invert unit-disc-fix-discI unit-disc-fix-moebius-inv)
  moreover
  have ∃ y ∈ unit-disc. Mψ y
    using ⟨y ∈ unit-disc⟩ ⟨ψ y⟩ ⟨Mψ = ?Mψ⟩ ⟨unit-disc-fix M⟩
    by auto
  moreover
  have set-order ?Mu ?Mφ ?Mψ
  proof ((rule ballI)+, rule impI)
    fix Mx My
    assume Mx ∈ unit-disc My ∈ unit-disc ?Mφ Mx ∧ ?Mψ My
    then obtain x y where φ x ∧ ?M x = Mx ψ y ∧ ?M y = My
    by blast

  hence x ∈ unit-disc y ∈ unit-disc
    using ⟨Mx ∈ unit-disc⟩ ⟨My ∈ unit-disc⟩ ⟨unit-disc-fix M⟩
    by (metis moebius-pt-comp-inv-left unit-disc-fix-discI unit-disc-fix-moebius-inv)+

  hence poincare-between u x y
    using ⟨set-order u φ ψ⟩

```

```

    using ⟨Mx ∈ unit-disc⟩ ⟨My ∈ unit-disc⟩ ⟨φ x ∧ ?M x = Mx⟩ ⟨ψ y ∧ ?M y = My⟩
    by blast
  then show poincare-between ?Mu Mx My
    using ⟨φ x ∧ ?M x = Mx⟩ ⟨ψ y ∧ ?M y = My⟩
    using ⟨x ∈ unit-disc⟩ ⟨y ∈ unit-disc⟩ ⟨u ∈ unit-disc⟩ ⟨unit-disc-fix M⟩
    using unit-disc-fix-moebius-preserve-poincare-between by blast
qed

hence set-order ?Mu Mφ Mψ
  using ⟨Mφ = ?Mφ⟩ ⟨Mψ = ?Mψ⟩
  by simp
ultimately
have ∃ Mb ∈ unit-disc. point-between-sets Mφ Mb Mψ
  using hip(5)
  by blast
then obtain Mb where bbb:
  Mb ∈ unit-disc point-between-sets ?Mφ Mb ?Mψ
  using ⟨Mφ = ?Mφ⟩ ⟨Mψ = ?Mψ⟩
  by auto

let ?b = moebius-pt (moebius-inv M) Mb
show ∃ b ∈ unit-disc. point-between-sets φ b ψ
proof (rule-tac x=?b in bexI, (rule ballI)+, rule impI)
  fix x y
  assume x ∈ unit-disc y ∈ unit-disc φ x ∧ ψ y
  hence poincare-between u x y
    using ⟨set-order u φ ψ⟩
    by blast

  let ?Mx = ?M x and ?My = ?M y

  have ?Mφ ?Mx ?Mψ ?My
    using ⟨φ x ∧ ψ y⟩
    by blast+
  have ?Mx ∈ unit-disc ?My ∈ unit-disc
    using ⟨x ∈ unit-disc⟩ ⟨unit-disc-fix M⟩ ⟨y ∈ unit-disc⟩
    by auto

  hence poincare-between ?Mx Mb ?My
    using ⟨?Mφ ?Mx⟩ ⟨?Mψ ?My⟩ ⟨?Mx ∈ unit-disc⟩ ⟨?My ∈ unit-disc⟩ bbb
    by auto

  then show poincare-between x ?b y
    using ⟨unit-disc-fix M⟩
    using ⟨x ∈ unit-disc⟩ ⟨y ∈ unit-disc⟩ ⟨Mb ∈ unit-disc⟩ ⟨?Mx ∈ unit-disc⟩ ⟨?My ∈ unit-disc⟩
    using unit-disc-fix-moebius-preserve-poincare-between[of M x ?b y]
    by auto
next
show ?b ∈ unit-disc
  using bbb ⟨unit-disc-fix M⟩
  by auto
qed
qed
next
fix X
assume xx: is-real X 0 < Re X Re X < 1
let ?X = of-complex X
show ?P 0h ?X
proof ((rule allI)+, rule impI, (erule conjE)+)
  fix φ ψ
  assume set-order 0h φ ψ ¬ (∃ B ∈ unit-disc. φ B ∧ ψ B) φ ?X
    ∃ y ∈ unit-disc. ψ y ∃ x ∈ unit-disc. φ x
  have ?X ∈ unit-disc
    using xx
    by (simp add: cmod-eq-Re)

```

**have**  $\psi_{\text{pos}}: \forall y \in \text{unit-disc}. \psi y \longrightarrow (\text{is-real } (\text{to-complex } y) \wedge \text{Re } (\text{to-complex } y) > 0)$   
**proof**(rule ballI, rule impI)  
**fix**  $y$   
**let**  $?y = \text{to-complex } y$   
**assume**  $y \in \text{unit-disc } \psi y$

**hence** *poincare-between*  $0_h ?X y$   
**using**  $\langle \text{set-order } 0_h \varphi \psi \rangle$   
**using**  $\langle ?X \in \text{unit-disc} \rangle \langle \varphi ?X \rangle$   
**by** *auto*

**thus** *is-real*  $?y \wedge 0 < \text{Re } ?y$   
**using**  $xx \langle ?X \in \text{unit-disc} \rangle \langle y \in \text{unit-disc} \rangle$   
**by** (*metis* (*mono-tags*, *opaque-lifting*) *arg-0-iff of-complex-zero-iff poincare-between-0uv poincare-between-sandwich to-complex-of-complex unit-disc-to-complex-inj zero-in-unit-disc*)  
**qed**

**have**  $\varphi_{\text{noneg}}: \forall x \in \text{unit-disc}. \varphi x \longrightarrow (\text{is-real } (\text{to-complex } x) \wedge \text{Re } (\text{to-complex } x) \geq 0)$   
**proof**(rule ballI, rule impI)  
**fix**  $x$   
**assume**  $x \in \text{unit-disc } \varphi x$

**obtain**  $y$  **where**  $y \in \text{unit-disc } \psi y$   
**using**  $\langle \exists y \in \text{unit-disc}. \psi y \rangle$  **by** *blast*

**let**  $?x = \text{to-complex } x$  **and**  $?y = \text{to-complex } y$

**have** *is-real*  $?y \text{Re } ?y > 0$   
**using**  $\psi_{\text{pos}} \langle \psi y \rangle \langle y \in \text{unit-disc} \rangle$   
**by** *auto*

**have** *poincare-between*  $0_h x y$   
**using**  $\langle \text{set-order } 0_h \varphi \psi \rangle$   
**using**  $\langle x \in \text{unit-disc} \rangle \langle \varphi x \rangle \langle y \in \text{unit-disc} \rangle \langle \psi y \rangle$   
**by** *auto*

**thus** *is-real*  $?x \wedge 0 \leq \text{Re } ?x$   
**using**  $\langle x \in \text{unit-disc} \rangle \langle y \in \text{unit-disc} \rangle \langle \text{is-real } (\text{to-complex } y) \rangle \langle \psi y \rangle$   
**using**  $\langle \text{set-order } 0_h \varphi \psi \rangle$   
**using**  $\langle \varphi ?X \rangle \langle ?X \in \text{unit-disc} \rangle \langle \text{Re } ?y > 0 \rangle$   
**by** (*metis* *arg-0-iff le-less of-complex-zero poincare-between-0uv to-complex-of-complex zero-complex.simps(1) zero-complex.simps(2)*)  
**qed**

**have**  $\varphi_{\text{less}\psi}: \forall x \in \text{unit-disc}. \forall y \in \text{unit-disc}. \varphi x \wedge \psi y \longrightarrow \text{Re } (\text{to-complex } x) < \text{Re } (\text{to-complex } y)$   
**proof**((rule ballI)+, rule impI)  
**fix**  $x y$   
**let**  $?x = \text{to-complex } x$  **and**  $?y = \text{to-complex } y$   
**assume**  $x \in \text{unit-disc } y \in \text{unit-disc } \varphi x \wedge \psi y$

**hence** *poincare-between*  $0_h x y$   
**using**  $\langle \text{set-order } 0_h \varphi \psi \rangle$   
**by** *auto*

**moreover**  
**have** *is-real*  $?x \text{Re } ?x \geq 0$   
**using**  $\varphi_{\text{noneg}}$   
**using**  $\langle x \in \text{unit-disc} \rangle \langle \varphi x \wedge \psi y \rangle$  **by** *auto*

**moreover**  
**have** *is-real*  $?y \text{Re } ?y > 0$   
**using**  $\psi_{\text{pos}}$   
**using**  $\langle y \in \text{unit-disc} \rangle \langle \varphi x \wedge \psi y \rangle$  **by** *auto*

**ultimately**  
**have**  $\text{Re } ?x \leq \text{Re } ?y$   
**using**  $\langle x \in \text{unit-disc} \rangle \langle y \in \text{unit-disc} \rangle$   
**by** (*metis* *Re-complex-of-real arg-0-iff le-less of-complex-zero poincare-between-0uv rcis-cmod-Arg rcis-zero-arg to-complex-of-complex*)

```

have Re ?x ≠ Re ?y
  using ⟨φ x ∧ ψ y⟩ ⟨is-real ?x⟩ ⟨is-real ?y⟩
  using ⟨¬ (∃ B ∈ unit-disc. φ B ∧ ψ B)⟩ ⟨x ∈ unit-disc⟩ ⟨y ∈ unit-disc⟩
  by (metis complex.expand unit-disc-to-complex-inj)

thus Re ?x < Re ?y
  using ⟨Re ?x ≤ Re ?y⟩ by auto
qed

have ∃ b ∈ unit-disc. ∀ x ∈ unit-disc. ∀ y ∈ unit-disc.
  is-real (to-complex b) ∧
  (φ x ∧ ψ y → (Re (to-complex x) ≤ Re (to-complex b) ∧ Re (to-complex b) ≤ Re (to-complex y)))
proof -
let ?Phi = {x. (of-complex (cor x)) ∈ unit-disc ∧ φ (of-complex (cor x))}

have ∀ x ∈ unit-disc. φ x → Re (to-complex x) ≤ Sup ?Phi
proof (safe)
fix x
let ?x = to-complex x
assume x ∈ unit-disc φ x
hence is-real ?x Re ?x ≥ 0
  using φnoneg
  by auto
hence cor (Re ?x) = ?x
  using complex-of-real-Re by blast
hence of-complex (cor (Re ?x)) ∈ unit-disc
  using ⟨x ∈ unit-disc⟩
  by (metis inf-notin-unit-disc of-complex-to-complex)
moreover
have φ (of-complex (cor (Re ?x)))
  using ⟨cor (Re ?x) = ?x⟩ ⟨φ x⟩ ⟨x ∈ unit-disc⟩
  by (metis inf-notin-unit-disc of-complex-to-complex)
ultimately
have Re ?x ∈ ?Phi
  by auto

have ∃ M. ∀ x ∈ ?Phi. x ≤ M
  using φlessψ
  using ⟨∃ y ∈ unit-disc. ψ y⟩
  by (metis (mono-tags, lifting) Re-complex-of-real le-less mem-Collect-eq to-complex-of-complex)

thus Re ?x ≤ Sup ?Phi
  using cSup-upper[of Re ?x ?Phi]
  unfolding bdd-above-def
  using ⟨Re ?x ∈ ?Phi⟩
  by auto
qed

have ∀ y ∈ unit-disc. ψ y → Sup ?Phi ≤ Re (to-complex y)
proof (safe)
fix y
let ?y = to-complex y
assume ψ y y ∈ unit-disc
show Sup ?Phi ≤ Re ?y
proof (rule ccontr)
assume ¬ ?thesis
hence Re ?y < Sup ?Phi
  by auto
qed

have ∃ x. φ (of-complex (cor x)) ∧ (of-complex (cor x)) ∈ unit-disc
proof -
obtain x' where x' ∈ unit-disc φ x'
  using ⟨∃ x ∈ unit-disc. φ x⟩ by blast
let ?x' = to-complex x'
have is-real ?x'

```

**using**  $\langle x' \in \text{unit-disc} \rangle \langle \varphi x' \rangle$   
**using**  $\varphi \text{noneg}$   
**by** *auto*  
**hence**  $\text{cor } (Re ?x') = ?x'$   
**using** *complex-of-real-Re* **by** *blast*  
**hence**  $x' = \text{of-complex } (\text{cor } (Re ?x'))$   
**using**  $\langle x' \in \text{unit-disc} \rangle$   
**by** (*metis inf-notin-unit-disc of-complex-to-complex*)  
**show** *?thesis*  
**apply** (*rule-tac x=Re ?x' in exI*)  
**using**  $\langle x' \in \text{unit-disc} \rangle$   
**apply** (*subst (asm)  $\langle x' = \text{of-complex } (\text{cor } (Re ?x')) \rangle$ , simp*)  
**using**  $\langle \varphi x' \rangle$   
**by** (*subst (asm) (2)  $\langle x' = \text{of-complex } (\text{cor } (Re ?x')) \rangle$ , simp*)  
**qed**

**hence**  $?Phi \neq \{\}$   
**by** *auto*

**then obtain**  $x$  **where**  $\varphi (\text{of-complex } (\text{cor } x)) Re ?y < x$   
 $(\text{of-complex } (\text{cor } x)) \in \text{unit-disc}$

**using**  $\langle Re ?y < \text{Sup } ?Phi \rangle$   
**using** *less-cSupE[ $\text{of } Re ?y ?Phi$ ]*  
**by** *auto*

**moreover**  
**have**  $Re ?y < Re (\text{to-complex } (\text{of-complex } (\text{cor } x)))$   
**using**  $\langle Re ?y < x \rangle$   
**by** *simp*

**ultimately**  
**show** *False*  
**using**  $\varphi \text{less}\psi$   
**using**  $\langle \psi y \rangle \langle y \in \text{unit-disc} \rangle$   
**by** (*metis less-not-sym*)

**qed**  
**qed**

**thus** *?thesis*

**using**  $\langle \forall x \in \text{unit-disc}. \varphi x \longrightarrow Re (\text{to-complex } x) \leq \text{Sup } ?Phi \rangle$   
**apply** (*rule-tac x=(of-complex (cor (Sup ?Phi))) in bexI, simp*)  
**using**  $\langle \exists y \in \text{unit-disc}. \psi y \rangle \langle \varphi ?X \rangle \langle ?X \in \text{unit-disc} \rangle$   
**using**  $\langle \forall y \in \text{unit-disc}. \psi y \longrightarrow \text{is-real } (\text{to-complex } y) \wedge 0 < Re (\text{to-complex } y) \rangle$

**by** (*smt complex-of-real-Re inf-notin-unit-disc norm-of-real of-complex-to-complex to-complex-of-complex unit-disc-iff-cmod-lt-1 xx(2)*)  
**qed**

**then obtain**  $B$  **where**  $B \in \text{unit-disc}$  *is-real* (*to-complex*  $B$ )

$\forall x \in \text{unit-disc}. \forall y \in \text{unit-disc}. \varphi x \wedge \psi y \longrightarrow Re (\text{to-complex } x) \leq Re (\text{to-complex } B) \wedge$   
 $Re (\text{to-complex } B) \leq Re (\text{to-complex } y)$   
**by** *blast*

**show**  $\exists b \in \text{unit-disc}. \text{point-between-sets } \varphi b \psi$

**proof** (*rule-tac x=B in bexI*)

**show**  $B \in \text{unit-disc}$   
**by** *fact*

**next**

**show** *point-between-sets*  $\varphi B \psi$

**proof** (*(rule ballI)+, rule impI*)

**fix**  $x y$

**let**  $?x = \text{to-complex } x$  **and**  $?y = \text{to-complex } y$  **and**  $?B = \text{to-complex } B$

**assume**  $x \in \text{unit-disc } y \in \text{unit-disc } \varphi x \wedge \psi y$

**hence**  $Re ?x \leq Re ?B \wedge Re ?B \leq Re ?y$

**using**  $\langle \forall x \in \text{unit-disc}. \forall y \in \text{unit-disc}. \varphi x \wedge \psi y \longrightarrow Re (\text{to-complex } x) \leq Re ?B \wedge$   
 $Re (\text{to-complex } B) \leq Re (\text{to-complex } y) \rangle$

**by** *auto*

**moreover**

```

have is-real ?x Re ?x ≥ 0
  using φnoneg
  using ⟨x ∈ unit-disc⟩ ⟨φ x ∧ ψ y⟩
  by auto
moreover
have is-real ?y Re ?y > 0
  using ψpos
  using ⟨y ∈ unit-disc⟩ ⟨φ x ∧ ψ y⟩
  by auto
moreover
have cor (Re ?x) = ?x
  using complex-of-real-Re ⟨is-real ?x⟩ by blast
hence x = of-complex (cor (Re ?x))
  using ⟨x ∈ unit-disc⟩
  by (metis inf-notin-unit-disc of-complex-to-complex)
moreover
have cor (Re ?y) = ?y
  using complex-of-real-Re ⟨is-real ?y⟩ by blast
hence y = of-complex (cor (Re ?y))
  using ⟨y ∈ unit-disc⟩
  by (metis inf-notin-unit-disc of-complex-to-complex)
moreover
have cor (Re ?B) = ?B
  using complex-of-real-Re ⟨is-real (to-complex B)⟩ by blast
hence B = of-complex (cor (Re ?B))
  using ⟨B ∈ unit-disc⟩
  by (metis inf-notin-unit-disc of-complex-to-complex)
ultimately
show poincare-between x B y
  using ⟨is-real (to-complex B)⟩ ⟨x ∈ unit-disc⟩ ⟨y ∈ unit-disc⟩ ⟨B ∈ unit-disc⟩
  using poincare-between-x-axis-uvw[of Re (to-complex x) Re (to-complex B) Re (to-complex y)]
by (smt Re-complex-of-real arg-0-iff poincare-between-nonstrict(1) rcis-cmod-Arg rcis-zero-arg unit-disc-iff-cmod-lt-1)
qed
qed
qed
thus ?thesis
  using False ⟨φ X0⟩ ⟨ψ Y0⟩ * ⟨Y0 ∈ unit-disc⟩ ⟨X0 ∈ unit-disc⟩
  by auto
qed
qed
qed

```

## 11.8 Limiting parallels axiom

Auxiliary definitions

**definition** *poincare-on-line* **where**

*poincare-on-line*  $p a b \longleftrightarrow \text{poincare-collinear } \{p, a, b\}$

**definition** *poincare-on-ray* **where**

*poincare-on-ray*  $p a b \longleftrightarrow \text{poincare-between } a p b \vee \text{poincare-between } a b p$

**definition** *poincare-in-angle* **where**

*poincare-in-angle*  $p a b c \longleftrightarrow$

$b \neq a \wedge b \neq c \wedge p \neq b \wedge (\exists x \in \text{unit-disc. } \text{poincare-between } a x c \wedge x \neq a \wedge x \neq c \wedge \text{poincare-on-ray } p b x)$

**definition** *poincare-ray-meets-line* **where**

*poincare-ray-meets-line*  $a b c d \longleftrightarrow (\exists x \in \text{unit-disc. } \text{poincare-on-ray } x a b \wedge \text{poincare-on-line } x c d)$

All points on ray are collinear

**lemma** *poincare-on-ray-poincare-collinear*:

**assumes**  $p \in \text{unit-disc}$  **and**  $a \in \text{unit-disc}$  **and**  $b \in \text{unit-disc}$  **and** *poincare-on-ray*  $p a b$

**shows** *poincare-collinear*  $\{p, a, b\}$

**using** *assms poincare-between-poincare-collinear*

**unfolding** *poincare-on-ray-def*

by (*metis insert-commute*)

H-isometries preserve all defined auxiliary relations

**lemma** *unit-disc-fix-preserves-poincare-on-line* [*simp*]:

**assumes** *unit-disc-fix*  $M$  **and**  $p \in \text{unit-disc}$   $a \in \text{unit-disc}$   $b \in \text{unit-disc}$   
**shows** *poincare-on-line* (*moebius-pt*  $M$   $p$ ) (*moebius-pt*  $M$   $a$ ) (*moebius-pt*  $M$   $b$ )  $\longleftrightarrow$  *poincare-on-line*  $p$   $a$   $b$   
**using** *assms*  
**unfolding** *poincare-on-line-def*  
**by** *auto*

**lemma** *unit-disc-fix-preserves-poincare-on-ray* [*simp*]:

**assumes** *unit-disc-fix*  $M$   $p \in \text{unit-disc}$   $a \in \text{unit-disc}$   $b \in \text{unit-disc}$   
**shows** *poincare-on-ray* (*moebius-pt*  $M$   $p$ ) (*moebius-pt*  $M$   $a$ ) (*moebius-pt*  $M$   $b$ )  $\longleftrightarrow$  *poincare-on-ray*  $p$   $a$   $b$   
**using** *assms*  
**unfolding** *poincare-on-ray-def*  
**by** *auto*

**lemma** *unit-disc-fix-preserves-poincare-in-angle* [*simp*]:

**assumes** *unit-disc-fix*  $M$   $p \in \text{unit-disc}$   $a \in \text{unit-disc}$   $b \in \text{unit-disc}$   $c \in \text{unit-disc}$   
**shows** *poincare-in-angle* (*moebius-pt*  $M$   $p$ ) (*moebius-pt*  $M$   $a$ ) (*moebius-pt*  $M$   $b$ ) (*moebius-pt*  $M$   $c$ )  $\longleftrightarrow$  *poincare-in-angle*  $p$   $a$   $b$   $c$  (**is** *?lhs*  $\longleftrightarrow$  *?rhs*)

**proof**

**assume** *?lhs*  
**then obtain**  $Mx$  **where**  $*$ :  $Mx \in \text{unit-disc}$   
    *poincare-between* (*moebius-pt*  $M$   $a$ )  $Mx$  (*moebius-pt*  $M$   $c$ )  
     $Mx \neq \text{moebius-pt } M \ a$   $Mx \neq \text{moebius-pt } M \ c$  *poincare-on-ray* (*moebius-pt*  $M$   $p$ ) (*moebius-pt*  $M$   $b$ )  $Mx$   
    *moebius-pt*  $M$   $b \neq \text{moebius-pt } M \ a$  *moebius-pt*  $M$   $b \neq \text{moebius-pt } M \ c$  *moebius-pt*  $M$   $p \neq \text{moebius-pt } M \ b$   
**unfolding** *poincare-in-angle-def*  
**by** *auto*

**obtain**  $x$  **where**  $Mx = \text{moebius-pt } M \ x$   $x \in \text{unit-disc}$

**by** (*metis*  $*(1)$  *assms(1)* *image-iff unit-disc-fix-iff*)

**thus** *?rhs*

**using**  $*$  *assms*

**unfolding** *poincare-in-angle-def*

**by** *auto*

**next**

**assume** *?rhs*

**then obtain**  $x$  **where**  $*$ :  $x \in \text{unit-disc}$

*poincare-between*  $a$   $x$   $c$

$x \neq a$   $x \neq c$  *poincare-on-ray*  $p$   $b$   $x$

$b \neq a$   $b \neq c$   $p \neq b$

**unfolding** *poincare-in-angle-def*

**by** *auto*

**thus** *?lhs*

**using** *assms*

**unfolding** *poincare-in-angle-def*

**by** *auto* (*rule-tac*  $x = \text{moebius-pt } M \ x$  **in** *bestI*, *auto*)

**qed**

**lemma** *unit-disc-fix-preserves-poincare-ray-meets-line* [*simp*]:

**assumes** *unit-disc-fix*  $M$   $a \in \text{unit-disc}$   $b \in \text{unit-disc}$   $c \in \text{unit-disc}$   $d \in \text{unit-disc}$

**shows** *poincare-ray-meets-line* (*moebius-pt*  $M$   $a$ ) (*moebius-pt*  $M$   $b$ ) (*moebius-pt*  $M$   $c$ ) (*moebius-pt*  $M$   $d$ )  $\longleftrightarrow$  *poincare-ray-meets-line*  $a$   $b$   $c$   $d$  (**is** *?lhs*  $\longleftrightarrow$  *?rhs*)

**proof**

**assume** *?lhs*

**then obtain**  $Mx$  **where**  $*$ :  $Mx \in \text{unit-disc}$  *poincare-on-ray*  $Mx$  (*moebius-pt*  $M$   $a$ ) (*moebius-pt*  $M$   $b$ )

*poincare-on-line*  $Mx$  (*moebius-pt*  $M$   $c$ ) (*moebius-pt*  $M$   $d$ )

**unfolding** *poincare-ray-meets-line-def*

**by** *auto*

**obtain**  $x$  **where**  $Mx = \text{moebius-pt } M \ x$   $x \in \text{unit-disc}$

**by** (*metis*  $*(1)$  *assms(1)* *image-iff unit-disc-fix-iff*)

**thus** *?rhs*

**using** *assms*  $*$

**unfolding** *poincare-ray-meets-line-def* *poincare-on-line-def*

**by** *auto*

**next**

```

assume ?rhs
then obtain x where *: x ∈ unit-disc poincare-on-ray x a b
  poincare-on-line x c d
  unfolding poincare-ray-meets-line-def
  by auto
thus ?lhs
  using assms *
  unfolding poincare-ray-meets-line-def poincare-on-line-def
  by auto (rule-tac x=moebius-pt M x in be $\mathbb{R}$ I, auto)
qed

```

H-lines that intersect on the absolute do not meet (they do not share a common h-point)

**lemma** *tangent-not-meet*:

```

assumes x1 ∈ unit-disc and x2 ∈ unit-disc and x1 ≠ x2 and ¬ poincare-collinear {0h, x1, x2}
assumes i ∈ ideal-points (poincare-line x1 x2) a ∈ unit-disc a ≠ 0h poincare-collinear {0h, a, i}
shows ¬ poincare-ray-meets-line 0h a x1 x2

```

**proof** (*rule* *ccontr*)

**assume** ¬ ?thesis

**then obtain** x **where** x ∈ unit-disc *poincare-on-ray* x 0<sub>h</sub> a *poincare-collinear* {x, x1, x2}

**unfolding** *poincare-ray-meets-line-def* *poincare-on-line-def*

**by** auto

**have** *poincare-collinear* {0<sub>h</sub>, a, x}

**using** ⟨*poincare-on-ray* x 0<sub>h</sub> a⟩ ⟨x ∈ unit-disc⟩ ⟨a ∈ unit-disc⟩

**by** (*meson* *poincare-between-poincare-collinear* *poincare-between-rev* *poincare-on-ray-def* *poincare-on-ray-poincare-collinear* *zero-in-unit-disc*)

**have** x ≠ 0<sub>h</sub>

**using** ⟨¬ *poincare-collinear* {0<sub>h</sub>, x1, x2}⟩ ⟨*poincare-collinear* {x, x1, x2}⟩

**unfolding** *poincare-collinear-def*

**by** (*auto* *simp* *add*: *assms*(2) *assms*(3) *poincare-between-rev*)

**let** ?l1 = *poincare-line* 0<sub>h</sub> a

**let** ?l2 = *poincare-line* x1 x2

**have** i ∈ *circline-set* *unit-circle*

**using** ⟨i ∈ *ideal-points* (*poincare-line* x1 x2)⟩

**using** *assms*(3) *ideal-points-on-unit-circle* *is-poincare-line-poincare-line* **by** *blast*

**have** i ∈ *circline-set* ?l1

**using** ⟨*poincare-collinear* {0<sub>h</sub>, a, i}⟩

**unfolding** *poincare-collinear-def*

**using** ⟨a ∈ unit-disc⟩ ⟨a ≠ 0<sub>h</sub>⟩

**by** (*metis* *insert-subset* *unique-poincare-line* *zero-in-unit-disc*)

**moreover**

**have** x ∈ *circline-set* ?l1

**using** ⟨a ∈ unit-disc⟩ ⟨a ≠ 0<sub>h</sub>⟩ ⟨*poincare-collinear* {0<sub>h</sub>, a, x}⟩ ⟨x ∈ unit-disc⟩

**by** (*metis* *poincare-collinear3-between* *poincare-between-poincare-line-uvw* *poincare-between-poincare-line-uvw* *poincare-line-sym* *zero-in-unit-disc*)

**moreover**

**have** *inversion* x ∈ *circline-set* ?l1

**using** ⟨*poincare-collinear* {0<sub>h</sub>, a, x}⟩

**using** *poincare-line-inversion-full*[of 0<sub>h</sub> a x] ⟨a ∈ unit-disc⟩ ⟨a ≠ 0<sub>h</sub>⟩ ⟨x ∈ unit-disc⟩

**by** (*metis* *poincare-collinear3-between* *is-poincare-line-inverse-point* *is-poincare-line-poincare-line* *poincare-between-poincare-line-uvw* *poincare-between-poincare-line-uvw* *poincare-line-sym* *zero-in-unit-disc*)

**moreover**

**have** x ∈ *circline-set* ?l2

**using** ⟨*poincare-collinear* {x, x1, x2}⟩ ⟨x1 ≠ x2⟩ ⟨x1 ∈ unit-disc⟩ ⟨x2 ∈ unit-disc⟩ ⟨x ∈ unit-disc⟩

**by** (*metis* *insert-commute* *inversion-noteq-unit-disc* *poincare-between-poincare-line-uvw* *poincare-between-poincare-line-uvw* *poincare-collinear3-iff* *poincare-line-sym-general*)



moreover

hence  $\text{inversion } x \in \text{circline-set } ?l2$   
using  $\langle x1 \neq x2 \rangle \langle x1 \in \text{unit-disc} \rangle \langle x2 \in \text{unit-disc} \rangle \langle x \in \text{unit-disc} \rangle$   
using  $\text{poincare-line-inversion-full}[of\ x1\ x2\ x]$   
unfolding  $\text{circline-set-def}$   
by  $\text{auto}$

moreover

have  $i \in \text{circline-set } ?l2$   
using  $\langle x1 \neq x2 \rangle \langle x1 \in \text{unit-disc} \rangle \langle x2 \in \text{unit-disc} \rangle$   
using  $\langle i \in \text{ideal-points } ?l2 \rangle$   
by  $(\text{simp add: ideal-points-on-circline})$

moreover

have  $x \neq \text{inversion } x$   
using  $\langle x \in \text{unit-disc} \rangle$   
using  $\text{inversion-noteq-unit-disc}$  by  $\text{fastforce}$

moreover

have  $x \neq i$   
using  $\langle x \in \text{unit-disc} \rangle$   
using  $\langle i \in \text{circline-set unit-circle} \rangle \text{circline-set-def inversion-noteq-unit-disc}$   
by  $\text{fastforce+}$

moreover

have  $\text{inversion } x \neq i$   
using  $\langle i \in \text{circline-set unit-circle} \rangle \langle x \neq i \rangle \text{circline-set-def inversion-unit-circle}$   
by  $\text{fastforce}$

ultimately

have  $?l1 = ?l2$   
using  $\text{unique-circline-set}[of\ x\ \text{inversion } x\ i]$   
by  $\text{blast}$

hence  $0_h \in \text{circline-set } ?l2$   
by  $(\text{metis } \langle a \neq 0_h \rangle \text{poincare-line-circline-set}(1))$

thus  $\text{False}$

using  $\langle \neg \text{poincare-collinear } \{0_h, x1, x2\} \rangle$   
unfolding  $\text{poincare-collinear-def}$   
using  $\langle \text{poincare-collinear } \{x, x1, x2\} \rangle \langle x1 \neq x2 \rangle \langle x1 \in \text{unit-disc} \rangle \langle x2 \in \text{unit-disc} \rangle \text{poincare-collinear-def unique-poincare-line}$   
by  $\text{auto}$

qed

lemma  $\text{limiting-parallels}$ :

assumes  $a \in \text{unit-disc}$  and  $x1 \in \text{unit-disc}$  and  $x2 \in \text{unit-disc}$  and  $\neg \text{poincare-on-line } a\ x1\ x2$

shows  $\exists a1 \in \text{unit-disc}. \exists a2 \in \text{unit-disc}.$

$\neg \text{poincare-on-line } a\ a1\ a2 \wedge$

$\neg \text{poincare-ray-meets-line } a\ a1\ x1\ x2 \wedge \neg \text{poincare-ray-meets-line } a\ a2\ x1\ x2 \wedge$

$(\forall a' \in \text{unit-disc}. \text{poincare-in-angle } a'\ a1\ a\ a2 \longrightarrow \text{poincare-ray-meets-line } a'\ x1\ x2)$  (is ?P a x1 x2)

proof –

have  $\neg \text{poincare-collinear } \{a, x1, x2\}$   
using  $\langle \neg \text{poincare-on-line } a\ x1\ x2 \rangle$   
unfolding  $\text{poincare-on-line-def}$   
by  $\text{simp}$

have  $\forall x1\ x2. x1 \in \text{unit-disc} \wedge x2 \in \text{unit-disc} \wedge \neg \text{poincare-collinear } \{a, x1, x2\} \longrightarrow ?P\ a\ x1\ x2$  (is ?Q a)

proof (rule  $\text{wlog-zero}[OF\ \langle a \in \text{unit-disc} \rangle]$ )

fix a u

```

assume *:  $u \in \text{unit-disc cmod } a < 1$ 
hence  $uf$ :  $\text{unit-disc-fix (blaschke } a)$ 
  by simp
assume **:  $?Q (\text{moebius-pt (blaschke } a) u)$ 
show  $?Q u$ 
proof safe
  fix  $x1\ x2$ 
  let  $?M = \text{moebius-pt (blaschke } a)$ 
  assume  $xx$ :  $x1 \in \text{unit-disc } x2 \in \text{unit-disc} \neg \text{poincare-collinear } \{u, x1, x2\}$ 
  hence  $MM$ :  $?M\ x1 \in \text{unit-disc} \wedge ?M\ x2 \in \text{unit-disc} \wedge \neg \text{poincare-collinear } \{?M\ u, ?M\ x1, ?M\ x2\}$ 
    using *
    by auto
  show  $?P\ u\ x1\ x2$  (is  $\exists a1 \in \text{unit-disc}. \exists a2 \in \text{unit-disc}. ?P'\ a1\ a2\ u\ x1\ x2$ )
  proof–
    obtain  $Ma1\ Ma2$  where  $MM$ :  $Ma1 \in \text{unit-disc } Ma2 \in \text{unit-disc } ?P'\ Ma1\ Ma2\ (?M\ u)\ (?M\ x1)\ (?M\ x2)$ 
      using  $**[\text{rule-format, OF } MM]$ 
      by blast
    hence  $MM'$ :  $\forall a' \in \text{unit-disc}. \text{poincare-in-angle } a'\ Ma1\ (?M\ u)\ Ma2 \longrightarrow \text{poincare-ray-meets-line } (?M\ u)\ a'\ (?M\ x1)\ (?M\ x2)$ 
      by auto
    obtain  $a1\ a2$  where  $a$ :  $a1 \in \text{unit-disc } a2 \in \text{unit-disc } ?M\ a1 = Ma1\ ?M\ a2 = Ma2$ 
      using  $uf$ 
      by ( $\text{metis } \langle Ma1 \in \text{unit-disc} \rangle \langle Ma2 \in \text{unit-disc} \rangle \text{image-iff unit-disc-fix-iff}$ )

  have  $\forall a' \in \text{unit-disc}. \text{poincare-in-angle } a'\ a1\ u\ a2 \longrightarrow \text{poincare-ray-meets-line } u\ a'\ x1\ x2$ 
  proof safe
    fix  $a'$ 
    assume  $a' \in \text{unit-disc } \text{poincare-in-angle } a'\ a1\ u\ a2$ 
    thus  $\text{poincare-ray-meets-line } u\ a'\ x1\ x2$ 
      using  $MM(1-2)\ MM'[\text{rule-format, of } ?M\ a'] * uf\ a\ xx$ 
    by ( $\text{meson unit-disc-fix-discI unit-disc-fix-preserves-poincare-in-angle unit-disc-fix-preserves-poincare-ray-meets-line}$ )
  qed

  hence  $?P'\ a1\ a2\ u\ x1\ x2$ 
    using  $MM * uf\ xx\ a$ 
    by auto

  thus  $?thesis$ 
    using  $\langle a1 \in \text{unit-disc} \rangle \langle a2 \in \text{unit-disc} \rangle$ 
    by blast
qed
qed
next
show  $?Q\ 0_h$ 
proof safe
  fix  $x1\ x2$ 
  assume  $x1 \in \text{unit-disc } x2 \in \text{unit-disc}$ 
  assume  $\neg \text{poincare-collinear } \{0_h, x1, x2\}$ 
  show  $?P\ 0_h\ x1\ x2$ 
  proof–
    let  $?lx = \text{poincare-line } x1\ x2$ 

    have  $x1 \neq x2$ 
      using  $\langle x1 \in \text{unit-disc} \rangle \langle x2 \in \text{unit-disc} \rangle \langle \neg \text{poincare-collinear } \{0_h, x1, x2\} \rangle$ 
      using  $\text{poincare-collinear3-between}$ 
      by auto

    have  $lx$ :  $\text{is-poincare-line } ?lx$ 
      using  $\text{is-poincare-line-poincare-line}[OF\ \langle x1 \neq x2 \rangle]$ 
      by simp

    obtain  $i1\ i2$  where  $\text{ideal-points } ?lx = \{i1, i2\}$ 
      by ( $\text{meson } \langle x1 \neq x2 \rangle \text{is-poincare-line-poincare-line obtain-ideal-points}$ )

    let  $?li = \text{poincare-line } i1\ i2$ 
    let  $?i1 = \text{to-complex } i1$ 

```

```

let ?i2 = to-complex i2

have i1 ∈ unit-circle-set i2 ∈ unit-circle-set
  using ⟨ideal-points ?lx = {i1, i2}⟩
  unfolding unit-circle-set-def
  by (metis ideal-points-on-unit-circle insertI1, metis ideal-points-on-unit-circle insertI1 insertI2)

have i1 ≠ i2
  using ⟨ideal-points ?lx = {i1, i2}⟩ ⟨x1 ∈ unit-disc⟩ ⟨x1 ≠ x2⟩ ⟨x2 ∈ unit-disc⟩ ideal-points-different(1)
  by blast

let ?a1 = of-complex (?i1 / 2)
let ?a2 = of-complex (?i2 / 2)
let ?la = poincare-line ?a1 ?a2

have ?a1 ∈ unit-disc ?a2 ∈ unit-disc
  using ⟨i1 ∈ unit-circle-set⟩ ⟨i2 ∈ unit-circle-set⟩
  unfolding unit-circle-set-def unit-disc-def disc-def circline-set-def
  by auto (transfer, transfer, case-tac i1, case-tac i2, simp add: vec-cnj-def)+

have ?a1 ≠ 0h ?a2 ≠ 0h
  using ⟨i1 ∈ unit-circle-set⟩ ⟨i2 ∈ unit-circle-set⟩
  unfolding unit-circle-set-def
  by auto

have ?a1 ≠ ?a2
  using ⟨i1 ≠ i2⟩
  by (metis ⟨i1 ∈ unit-circle-set⟩ ⟨i2 ∈ unit-circle-set⟩ circline-set-def divide-cancel-right inversion-infty inversion-unit-circle mem-Collect-eq-of-complex-to-complex-of-complex-zero to-complex-of-complex unit-circle-set-def zero-neq-numeral)

have poincare-collinear {0h, ?a1, i1}
  unfolding poincare-collinear-def
  using ⟨?a1 ≠ 0h⟩[symmetric] is-poincare-line-poincare-line[of 0h ?a1]
  unfolding circline-set-def
  apply (rule-tac x=poincare-line 0h ?a1 in exI, auto)
  apply (transfer, transfer, auto simp add: vec-cnj-def)
  done

have poincare-collinear {0h, ?a2, i2}
  unfolding poincare-collinear-def
  using ⟨?a2 ≠ 0h⟩[symmetric] is-poincare-line-poincare-line[of 0h ?a2]
  unfolding circline-set-def
  apply (rule-tac x=poincare-line 0h ?a2 in exI, auto)
  apply (transfer, transfer, auto simp add: vec-cnj-def)
  done

have ¬ poincare-ray-meets-line 0h ?a1 x1 x2
  using tangent-not-meet[of x1 x2 i1 ?a1]
  using ⟨x1 ∈ unit-disc⟩ ⟨x2 ∈ unit-disc⟩ ⟨?a1 ∈ unit-disc⟩ ⟨x1 ≠ x2⟩ ⟨¬ poincare-collinear {0h, x1, x2}⟩
  using ⟨ideal-points ?lx = {i1, i2}⟩ ⟨?a1 ≠ 0h⟩ ⟨poincare-collinear {0h, ?a1, i1}⟩
  by simp

moreover

have ¬ poincare-ray-meets-line 0h ?a2 x1 x2
  using tangent-not-meet[of x1 x2 i2 ?a2]
  using ⟨x1 ∈ unit-disc⟩ ⟨x2 ∈ unit-disc⟩ ⟨?a2 ∈ unit-disc⟩ ⟨x1 ≠ x2⟩ ⟨¬ poincare-collinear {0h, x1, x2}⟩
  using ⟨ideal-points ?lx = {i1, i2}⟩ ⟨?a2 ≠ 0h⟩ ⟨poincare-collinear {0h, ?a2, i2}⟩
  by simp

moreover

have ∀ a' ∈ unit-disc. poincare-in-angle a' ?a1 0h ?a2 → poincare-ray-meets-line 0h a' x1 x2
  unfolding poincare-in-angle-def
proof safe
  fix a' a

```

```

assume *:  $a' \in \text{unit-disc } a \in \text{unit-disc } \text{poincare-on-ray } a' \ 0_h \ a \ a' \neq 0_h$ 
            $\text{poincare-between } ?a1 \ a \ ?a2 \ a \neq ?a1 \ a \neq ?a2$ 
show  $\text{poincare-ray-meets-line } 0_h \ a' \ x1 \ x2$ 
proof–
  have  $\forall a' \ a1 \ a2 \ x1 \ x2 \ i1 \ i2.$ 
     $a' \in \text{unit-disc} \wedge x1 \in \text{unit-disc} \wedge x2 \in \text{unit-disc} \wedge x1 \neq x2 \wedge$ 
     $\neg \text{poincare-collinear } \{0_h, x1, x2\} \wedge \text{ideal-points } (\text{poincare-line } x1 \ x2) = \{i1, i2\} \wedge$ 
     $a1 = \text{of-complex } (\text{to-complex } i1 / 2) \wedge a2 = \text{of-complex } (\text{to-complex } i2 / 2) \wedge$ 
     $i1 \neq i2 \wedge a1 \neq a2 \wedge \text{poincare-collinear } \{0_h, a1, i1\} \wedge \text{poincare-collinear } \{0_h, a2, i2\} \wedge$ 
     $a1 \in \text{unit-disc} \wedge a2 \in \text{unit-disc} \wedge i1 \in \text{unit-circle-set} \wedge i2 \in \text{unit-circle-set} \wedge$ 
     $\text{poincare-on-ray } a' \ 0_h \ a \wedge a' \neq 0_h \wedge \text{poincare-between } a1 \ a \ a2 \wedge a \neq a1 \wedge a \neq a2 \longrightarrow$ 
     $\text{poincare-ray-meets-line } 0_h \ a' \ x1 \ x2$  (is  $\forall a' \ a1 \ a2 \ x1 \ x2 \ i1 \ i2. ?R \ 0_h \ a' \ a1 \ a2 \ x1 \ x2 \ i1 \ i2 \ a$ )
  proof (rule wlog-rotation-to-positive-x-axis[OF  $\langle a \in \text{unit-disc} \rangle$ ])
  let  $?R' = \lambda a \ \text{zero}. \forall a' \ a1 \ a2 \ x1 \ x2 \ i1 \ i2. ?R \ \text{zero } a' \ a1 \ a2 \ x1 \ x2 \ i1 \ i2 \ a$ 
  fix  $xa$ 
  assume  $xa: \text{is-real } xa \ 0 < \text{Re } xa \ \text{Re } xa < 1$ 
  let  $?a = \text{of-complex } xa$ 
  show  $?R' \ ?a \ 0_h$ 
  proof safe
    fix  $a' \ a1 \ a2 \ x1 \ x2 \ i1 \ i2$ 
    let  $?i1 = \text{to-complex } i1$  and  $?i2 = \text{to-complex } i2$ 
    let  $?a1 = \text{of-complex } (?i1 / 2)$  and  $?a2 = \text{of-complex } (?i2 / 2)$ 
    let  $?la = \text{poincare-line } ?a1 \ ?a2$  and  $?lx = \text{poincare-line } x1 \ x2$  and  $?li = \text{poincare-line } i1 \ i2$ 
    assume  $a' \in \text{unit-disc} \ x1 \in \text{unit-disc} \ x2 \in \text{unit-disc} \ x1 \neq x2$ 
    assume  $\neg \text{poincare-collinear } \{0_h, x1, x2\} \ \text{ideal-points } ?lx = \{i1, i2\}$ 
    assume  $\text{poincare-on-ray } a' \ 0_h \ ?a \ a' \neq 0_h$ 
    assume  $\text{poincare-between } ?a1 \ ?a \ ?a2 \ ?a \neq ?a1 \ ?a \neq ?a2$ 
    assume  $i1 \neq i2 \ ?a1 \neq ?a2 \ \text{poincare-collinear } \{0_h, ?a1, i1\} \ \text{poincare-collinear } \{0_h, ?a2, i2\}$ 
    assume  $?a1 \in \text{unit-disc} \ ?a2 \in \text{unit-disc}$ 
    assume  $i1 \in \text{unit-circle-set} \ i2 \in \text{unit-circle-set}$ 
    show  $\text{poincare-ray-meets-line } 0_h \ a' \ x1 \ x2$ 
    proof–
      have  $?lx = ?li$ 
        using  $\langle \text{ideal-points } ?lx = \{i1, i2\} \rangle \langle x1 \neq x2 \rangle \ \text{ideal-points-line-unique}$ 
        by auto

      have  $lx: \text{is-poincare-line } ?lx$ 
        using  $\text{is-poincare-line-poincare-line}$ [OF  $\langle x1 \neq x2 \rangle$ ]
        by simp

      have  $x1 \in \text{circline-set } ?lx \ x2 \in \text{circline-set } ?lx$ 
        using  $lx \ \langle x1 \neq x2 \rangle$ 
        by auto

      have  $?lx \neq x\text{-axis}$ 
        using  $\langle \neg \text{poincare-collinear } \{0_h, x1, x2\} \rangle \langle x1 \in \text{circline-set } ?lx \rangle \langle x2 \in \text{circline-set } ?lx \rangle \ lx$ 
        unfolding  $\text{poincare-collinear-def}$ 
        by auto

      have  $0_h \notin \text{circline-set } ?lx$ 
        using  $\langle \neg \text{poincare-collinear } \{0_h, x1, x2\} \rangle \ lx \ \langle x1 \in \text{circline-set } ?lx \rangle \langle x2 \in \text{circline-set } ?lx \rangle$ 
        unfolding  $\text{poincare-collinear-def}$ 
        by auto

      have  $xa \neq 0 \ ?a \neq 0_h$ 
        using  $xa$ 
        by auto
      hence  $0_h \neq ?a$ 
        by metis

      have  $?a \in \text{positive-x-axis}$ 
        using  $xa$ 
        unfolding  $\text{positive-x-axis-def}$ 
        by simp

      have  $?a \in \text{unit-disc}$ 

```

```

using xa
by (auto simp add: cmod-eq-Re)

have ?a ∈ circline-set ?la
  using ⟨poincare-between ?a1 ?a ?a2⟩
using ⟨?a1 ≠ ?a2⟩ ⟨?a ∈ unit-disc⟩ ⟨?a1 ∈ unit-disc⟩ ⟨?a2 ∈ unit-disc⟩ poincare-between-poincare-line-uzv

  by blast

have ?a1 ∈ circline-set ?la ?a2 ∈ circline-set ?la
  by (auto simp add: ⟨?a1 ≠ ?a2⟩)

have la: is-poincare-line ?la
  using is-poincare-line-poincare-line[OF ⟨?a1 ≠ ?a2⟩]
  by simp

have inv: inversion i1 = i1 inversion i2 = i2
  using ⟨i1 ∈ unit-circle-set⟩ ⟨i2 ∈ unit-circle-set⟩
  by (auto simp add: circline-set-def unit-circle-set-def)

have i1 ≠ ∞h i2 ≠ ∞h
  using inv
  by auto

have ?a1 ∉ circline-set x-axis ∧ ?a2 ∉ circline-set x-axis
proof (rule ccontr)
  assume ¬ ?thesis
  hence ?a1 ∈ circline-set x-axis ∨ ?a2 ∈ circline-set x-axis
    by auto
  hence ?la = x-axis
  proof
    assume ?a1 ∈ circline-set x-axis
    hence {?a, ?a1} ⊆ circline-set ?la ∩ circline-set x-axis
      using ⟨?a ∈ circline-set ?la⟩ ⟨?a1 ∈ circline-set ?la⟩ ⟨?a ∈ positive-x-axis⟩
      using circline-set-x-axis-I xa(1)
      by blast
    thus ?la = x-axis
      using unique-is-poincare-line[of ?a ?a1 ?la x-axis]
      using ⟨?a1 ∈ unit-disc⟩ ⟨?a ∈ unit-disc⟩ la ⟨?a ≠ ?a1⟩
      by auto
  next
    assume ?a2 ∈ circline-set x-axis
    hence {?a, ?a2} ⊆ circline-set ?la ∩ circline-set x-axis
      using ⟨?a ∈ circline-set ?la⟩ ⟨?a2 ∈ circline-set ?la⟩ ⟨?a ∈ positive-x-axis⟩
      using circline-set-x-axis-I xa(1)
      by blast
    thus ?la = x-axis
      using unique-is-poincare-line[of ?a ?a2 ?la x-axis]
      using ⟨?a2 ∈ unit-disc⟩ ⟨?a ∈ unit-disc⟩ la ⟨?a ≠ ?a2⟩
      by auto
  qed

  hence i1 ∈ circline-set x-axis ∧ i2 ∈ circline-set x-axis
    using ⟨?a1 ∈ circline-set ?la⟩ ⟨?a2 ∈ circline-set ?la⟩
    by (metis ⟨i1 ≠ ∞h⟩ ⟨i2 ≠ ∞h⟩ ⟨of-complex (to-complex i1 / 2) ∈ unit-disc⟩ ⟨of-complex (to-complex i2 / 2) ∈ unit-disc⟩ ⟨poincare-collinear {0h, of-complex (to-complex i1 / 2), i1}⟩ ⟨poincare-collinear {0h, of-complex (to-complex i2 / 2), i2}⟩ divide-eq-0-iff inf-not-of-complex inv(1) inv(2) inversion-not-eq-unit-disc of-complex-to-complex of-complex-zero-iff poincare-collinear3-poincare-lines-equal-general poincare-line-0-real-is-x-axis poincare-line-circline-set(2) zero-in-unit-disc zero-neq-numeral)

  thus False
    using ⟨?lx ≠ x-axis⟩ unique-is-poincare-line-general[of i1 i2 ?li x-axis] ⟨i1 ≠ i2⟩ inv ⟨?lx = ?li⟩
    by auto
qed

hence ?la ≠ x-axis

```

using  $\langle ?a1 \neq ?a2 \rangle$  *poincare-line-circline-set(1)*  
 by *fastforce*

have *intersects-x-axis-positive ?la*  
 using *intersects-x-axis-positive-iff*[of ?la]  $\langle ?la \neq x\text{-axis} \rangle \langle ?a \in \text{circline-set } ?la \rangle$  *la*  
 using  $\langle ?a \in \text{unit-disc} \rangle \langle ?a \in \text{positive-x-axis} \rangle$   
 by *auto*

have *intersects-x-axis ?lx*

proof–

have  $\text{Arg } (\text{to-complex } ?a1) * \text{Arg } (\text{to-complex } ?a2) < 0$   
 using  $\langle \text{poincare-between } ?a1 ?a ?a2 \rangle \langle ?a1 \in \text{unit-disc} \rangle \langle ?a2 \in \text{unit-disc} \rangle$   
 using *poincare-between-x-axis-intersection*[of ?a1 ?a2 of-complex *xa*]  
 using  $\langle ?a1 \neq ?a2 \rangle \langle ?a \in \text{unit-disc} \rangle \langle ?a1 \notin \text{circline-set } x\text{-axis} \wedge ?a2 \notin \text{circline-set } x\text{-axis} \rangle \langle ?a \in$

*positive-x-axis*

using  $\langle ?a \in \text{circline-set } ?la \rangle$

unfolding *positive-x-axis-def*

by *simp*

moreover

have  $\bigwedge x y x' y' :: \text{real. } [\text{sgn } x' = \text{sgn } x; \text{sgn } y' = \text{sgn } y] \implies x*y < 0 \longleftrightarrow x'*y' < 0$   
 by (*metis sgn-less sgn-mult*)

ultimately

have  $\text{Im } (\text{to-complex } ?a1) * \text{Im } (\text{to-complex } ?a2) < 0$   
 using *arg-Im-sgn*[of *to-complex ?a1*] *arg-Im-sgn*[of *to-complex ?a2*]  
 using  $\langle ?a1 \in \text{unit-disc} \rangle \langle ?a2 \in \text{unit-disc} \rangle \langle ?a1 \notin \text{circline-set } x\text{-axis} \wedge ?a2 \notin \text{circline-set } x\text{-axis} \rangle$   
 using *inf-or-of-complex*[of ?a1] *inf-or-of-complex*[of ?a2] *circline-set-x-axis*  
 by (*metis circline-set-x-axis-I to-complex-of-complex*)

thus *?thesis*

using *ideal-points-intersects-x-axis*[of ?lx *i1 i2*]

using  $\langle \text{ideal-points } ?lx = \{i1, i2\} \rangle$  *lx*  $\langle ?lx \neq x\text{-axis} \rangle$

by *simp*

qed

have *intersects-x-axis-positive ?lx*

proof–

have  $\text{cmod } ?i1 = 1 \text{ cmod } ?i2 = 1$   
 using  $\langle i1 \in \text{unit-circle-set} \rangle \langle i2 \in \text{unit-circle-set} \rangle$   
 unfolding *unit-circle-set-def*  
 by *auto*

let  $?a1' = ?i1 / 2$  and  $?a2' = ?i2 / 2$

let  $?Aa1 = i * (?a1' * \text{cnj } ?a2' - ?a2' * \text{cnj } ?a1')$  and  
 $?Ba1 = i * (?a2' * \text{cor } ((\text{cmod } ?a1')^2 + 1) - ?a1' * \text{cor } ((\text{cmod } ?a2')^2 + 1))$

have  $?Aa1 \neq 0 \vee ?Ba1 \neq 0$

using  $\langle \text{cmod } (\text{to-complex } i1) = 1 \rangle \langle \text{cmod } (\text{to-complex } i2) = 1 \rangle \langle ?a1 \neq ?a2 \rangle$   
 by (*auto simp add: power-divide complex-mult-cnj-cmod*)

have *is-real ?Aa1*

by *simp*

have  $?a1 \neq \text{inversion } ?a2$

using  $\langle ?a1 \in \text{unit-disc} \rangle \langle ?a2 \in \text{unit-disc} \rangle$  *inversion-noteq-unit-disc* by *fastforce*

hence  $\text{Re } ?Ba1 / \text{Re } ?Aa1 < -1$

using  $\langle \text{intersects-x-axis-positive } ?la \rangle \langle ?a1 \neq ?a2 \rangle$

using *intersects-x-axis-positive-mk-circline*[of ?Aa1 ?Ba1]  $\langle ?Aa1 \neq 0 \vee ?Ba1 \neq 0 \rangle \langle \text{is-real } ?Aa1 \rangle$

using *poincare-line-non-homogenous*[of ?a1 ?a2]

by (*simp add: Let-def*)

moreover

```

let ?i1' = to-complex i1 and ?i2' = to-complex i2
let ?Ai1 = i * (?i1' * cnj ?i2' - ?i2' * cnj ?i1') and
    ?Bi1 = i * (?i2' * cor ((cmod ?i1')2 + 1) - ?i1' * cor ((cmod ?i2')2 + 1))

have ?Ai1 ≠ 0 ∨ ?Bi1 ≠ 0
  using ⟨cmod (to-complex i1) = 1⟩ ⟨cmod (to-complex i2) = 1⟩ ⟨?a1 ≠ ?a2⟩
  by (auto simp add: power-divide complex-mult-cnj-cmod)

have is-real ?Ai1
  by simp

have sgn (Re ?Bi1 / Re ?Ai1) = sgn (Re ?Ba1 / Re ?Aa1)
proof-
  have Re ?Bi1 / Re ?Ai1 = (Im ?i1 * 2 - Im ?i2 * 2) /
    (Im ?i2 * (Re ?i1 * 2) - Im ?i1 * (Re ?i2 * 2))
    using ⟨cmod ?i1 = 1⟩ ⟨cmod ?i2 = 1⟩
    by (auto simp add: complex-mult-cnj-cmod field-simps)
  also have ... = (Im ?i1 - Im ?i2) /
    (Im ?i2 * (Re ?i1) - Im ?i1 * (Re ?i2)) (is ... = ?expr)
    apply (subst left-diff-distrib[symmetric])
    apply (subst semiring-normalization-rules(18))+
    apply (subst left-diff-distrib[symmetric])
    by (metis mult.commute mult-divide-mult-cancel-left-if zero-neq-numeral)
  finally have 1: Re ?Bi1 / Re ?Ai1 = (Im ?i1 - Im ?i2) / (Im ?i2 * (Re ?i1) - Im ?i1 * (Re ?i2))
    .

  have Re ?Ba1 / Re ?Aa1 = (Im ?i1 * 20 - Im ?i2 * 20) /
    (Im ?i2 * (Re ?i1 * 16) - Im ?i1 * (Re ?i2 * 16))
    using ⟨cmod (to-complex i1) = 1⟩ ⟨cmod (to-complex i2) = 1⟩
    by (auto simp add: complex-mult-cnj-cmod field-simps)
  also have ... = (20 / 16) * ((Im ?i1 - Im ?i2) /
    (Im ?i2 * (Re ?i1) - Im ?i1 * (Re ?i2)))
    apply (subst left-diff-distrib[symmetric])+
    apply (subst semiring-normalization-rules(18))+
    apply (subst left-diff-distrib[symmetric])+
  by (metis (no-types, opaque-lifting) field-class.field-divide-inverse mult.commute times-divide-times-eq)
  finally have 2: Re ?Ba1 / Re ?Aa1 = (5 / 4) * ((Im ?i1 - Im ?i2) / (Im ?i2 * (Re ?i1) - Im ?i1
    * (Re ?i2)))
    by simp

  have ?expr ≠ 0
    using ⟨Re ?Ba1 / Re ?Aa1 < -1⟩
    apply (subst (asm) 2)
    by linarith
  thus ?thesis
    apply (subst 1, subst 2)
    apply (simp only: sgn-mult)
    by simp
qed

```

moreover

```

have i1 ≠ inversion i2
  by (simp add: ⟨i1 ≠ i2⟩ inv(2))

have (Re ?Bi1 / Re ?Ai1)2 > 1
proof-
  have ?Ai1 = 0 ∨ (Re ?Bi1)2 > (Re ?Ai1)2
    using ⟨intersects-x-axis ?lx⟩
    using ⟨i1 ≠ i2⟩ ⟨i1 ≠ ∞h⟩ ⟨i2 ≠ ∞h⟩ ⟨i1 ≠ inversion i2⟩
    using intersects-x-axis-mk-circline[of ?Ai1 ?Bi1] ⟨?Ai1 ≠ 0 ∨ ?Bi1 ≠ 0⟩ is-real ?Ai1
    using poincare-line-non-homogenous[of i1 i2] ⟨?lx = ?li⟩
    by metis

```

```

moreover
have  $?Ai1 \neq 0$ 
proof (rule ccontr)
  assume  $\neg ?thesis$ 
  hence  $0_h \in \text{circline-set } ?li$ 
  unfolding circline-set-def
  apply simp
  apply (transfer, transfer, case-tac i1, case-tac i2)
  by (auto simp add: vec-cnj-def field-simps)
thus False
  using  $\langle 0_h \notin \text{circline-set } ?lx \rangle \langle ?lx = ?li \rangle$ 
  by simp
qed

ultimately

have  $(\text{Re } ?Bi1)^2 > (\text{Re } ?Ai1)^2$ 
by auto

moreover

have  $\text{Re } ?Ai1 \neq 0$ 
  using  $\langle \text{is-real } ?Ai1 \rangle \langle ?Ai1 \neq 0 \rangle$ 
  by (simp add: complex-eq-iff)

ultimately

show ?thesis
  by (simp add: power-divide)
qed

moreover

{
  fix  $x1\ x2 :: \text{real}$ 
  assume  $\text{sgn } x1 = \text{sgn } x2\ x1 < -1\ x2^2 > 1$ 
  hence  $x2 < -1$ 
  by (smt one-power2 real-sqrt-abs real-sqrt-less-iff sgn-neg sgn-pos)
}

ultimately

have  $\text{Re } ?Bi1 / \text{Re } ?Ai1 < -1$ 
by metis

thus ?thesis
  using  $\langle i1 \neq i2 \rangle \langle i1 \neq \infty_h \rangle \langle i2 \neq \infty_h \rangle \langle i1 \neq \text{inversion } i2 \rangle$ 
  using intersects-x-axis-positive-mk-circline[of  $?Ai1\ ?Bi1$ ]  $\langle ?Ai1 \neq 0 \vee ?Bi1 \neq 0 \rangle \langle \text{is-real } ?Ai1 \rangle$ 
  using poincare-line-non-homogenous[of  $i1\ i2$ ]  $\langle ?lx = ?li \rangle$ 
  by (simp add: Let-def)
qed

then obtain  $x$  where  $x \in \text{unit-disc } x \in \text{circline-set } ?lx \cap \text{positive-x-axis}$ 
  using intersects-x-axis-positive-iff[OF  $lx\ \langle ?lx \neq \text{x-axis} \rangle$ ]
  by auto

have poincare-on-ray  $x\ 0_h\ a' \wedge \text{poincare-collinear } \{x1, x2, x\}$ 
proof
  show poincare-collinear  $\{x1, x2, x\}$ 
  using  $x\ lx\ \langle x1 \in \text{circline-set } ?lx \rangle \langle x2 \in \text{circline-set } ?lx \rangle$ 
  unfolding poincare-collinear-def
  by auto
next
  show poincare-on-ray  $x\ 0_h\ a'$ 
  unfolding poincare-on-ray-def

```



**proof**–

**have**  $a' \in \text{circline-set } x\text{-axis}$   
**using**  $\langle \text{poincare-on-ray } a' 0_h \text{ ?}a \rangle \langle xa \langle 0_h \neq ?a \rangle \langle xa \neq 0 \rangle \langle a' \in \text{unit-disc} \rangle$   
**unfolding**  $\text{poincare-on-ray-def}$   
**using**  $\text{poincare-line-0-real-is-x-axis}$ [of of-complex  $xa$ ]  
**using**  $\text{poincare-between-poincare-line-uvw}$ [of  $0_h$  of-complex  $xa a'$ ]  
**using**  $\text{poincare-between-poincare-line-uzv}$ [of  $0_h$  of-complex  $xa a'$ ]  
**by** (*auto simp add: cmod-eq-Re*)

**then obtain**  $xa'$  **where**  $xa': a' = \text{of-complex } xa' \text{ is-real } xa'$   
**using**  $\langle a' \in \text{unit-disc} \rangle$   
**using**  $\text{circline-set-def on-circline-x-axis}$   
**by** *auto*

**hence**  $-1 < \text{Re } xa' \text{ Re } xa' < 1 \text{ } xa' \neq 0$   
**using**  $\langle a' \in \text{unit-disc} \rangle \langle a' \neq 0_h \rangle$   
**by** (*auto simp add: cmod-eq-Re*)

**hence**  $\text{Re } xa' > 0 \text{ Re } xa' < 1 \text{ is-real } xa'$   
**using**  $\langle \text{poincare-on-ray } a' 0_h \text{ (of-complex } xa) \rangle$   
**using**  $\text{poincare-between-x-axis-0uv}$ [of  $\text{Re } xa' \text{ Re } xa$ ]  
**using**  $\text{poincare-between-x-axis-0uv}$ [of  $\text{Re } xa \text{ Re } xa'$ ]  
**using**  $\text{circline-set-positive-x-axis-I}$ [of  $\text{Re } xa'$ ]  
**using**  $xa \text{ } xa' \text{ complex-of-real-Re}$   
**unfolding**  $\text{poincare-on-ray-def}$   
**by** (*smt of-real-0, linarith, blast*)

**moreover**

**obtain**  $xx$  **where**  $\text{is-real } xx \text{ Re } xx > 0 \text{ Re } xx < 1 \text{ } x = \text{of-complex } xx$   
**using**  $x$   
**unfolding**  $\text{positive-x-axis-def}$   
**using**  $\text{circline-set-def cmod-eq-Re on-circline-x-axis}$   
**by** *auto*

**ultimately**

**show**  $\text{poincare-between } 0_h \text{ } x \text{ } a' \vee \text{poincare-between } 0_h \text{ } a' \text{ } x$   
**using**  $\langle a' = \text{of-complex } xa' \rangle$   
**by** (*smt  $\langle a' \in \text{unit-disc} \rangle \text{ arg-0-iff poincare-between-0uv poincare-between-def to-complex-of-complex}$* )

$x(1)$

**qed**

**qed**

**thus**  $?thesis$   
**using**  $\langle x \in \text{unit-disc} \rangle$   
**unfolding**  $\text{poincare-ray-meets-line-def poincare-on-line-def}$   
**by** (*metis insert-commute*)

**qed**

**qed**

**next**

**show**  $a \neq 0_h$

**proof** (*rule ccontr*)

**assume**  $\neg ?thesis$

**then obtain**  $k$  **where**  $k < 0 \text{ to-complex } ?a1 = \text{cor } k * \text{to-complex } ?a2$

**using**  $\text{poincare-between-u0v}$ [OF  $\langle ?a1 \in \text{unit-disc} \rangle \langle ?a2 \in \text{unit-disc} \rangle \langle ?a1 \neq 0_h \rangle \langle ?a2 \neq 0_h \rangle$ ]

**using**  $\langle \text{poincare-between } ?a1 \text{ } a \text{ } ?a2 \rangle$

**by** *auto*

**hence**  $\text{to-complex } i1 = \text{cor } k * \text{to-complex } i2 \text{ } k < 0$

**by** *auto*

**hence**  $0_h \in \text{circline-set (poincare-line } x1 \text{ } x2)$

**using**  $\text{ideal-points-proportional}$ [of  $\text{poincare-line } x1 \text{ } x2 \text{ } i1 \text{ } i2 \text{ } k$ ]  $\langle \text{ideal-points (poincare-line } x1 \text{ } x2) = \{i1,$

$i2\} \rangle$

**using**  $\text{is-poincare-line-poincare-line}$ [OF  $\langle x1 \neq x2 \rangle$ ]

**by** *simp*

```

thus False
  using  $\langle \neg \text{poincare-collinear } \{0_h, x1, x2\} \rangle$ 
  using  $\text{is-poincare-line-poincare-line}[OF \langle x1 \neq x2 \rangle]$ 
  unfolding poincare-collinear-def
  by (meson  $\langle x1 \neq x2 \rangle$  empty-subsetI insert-subset poincare-line-circline-set(1) poincare-line-circline-set(2))
qed
next
fix  $\varphi u$ 
let  $?R' = \lambda a \text{ zero. } \forall a' a1 a2 x1 x2 i1 i2. ?R \text{ zero } a' a1 a2 x1 x2 i1 i2 a$ 
let  $?M = \text{moebius-pt } (\text{moebius-rotation } \varphi)$ 
assume  $*$ :  $u \in \text{unit-disc } u \neq 0_h$  and  $**$ :  $?R' (?M u) 0_h$ 
have  $uf$ :  $\text{unit-disc-fix } (\text{moebius-rotation } \varphi)$ 
  by simp
have  $?M 0_h = 0_h$ 
  by auto
hence  $**$ :  $?R' (?M u) (?M 0_h)$ 
  using  $**$ 
  by simp
show  $?R' u 0_h$ 
proof (rule allI)+
  fix  $a' a1 a2 x1 x2 i1 i2$ 
  have  $i1$ :  $i1 \in \text{unit-circle-set} \longrightarrow \text{moebius-pt } (\text{moebius-rotation } \varphi) (\text{of-complex } (\text{to-complex } i1 / 2)) =$ 
of-complex (to-complex (moebius-pt (moebius-rotation  $\varphi$ )  $i1$ ) / 2)
  using unit-circle-set-def by force

  have  $i2$ :  $i2 \in \text{unit-circle-set} \longrightarrow \text{moebius-pt } (\text{moebius-rotation } \varphi) (\text{of-complex } (\text{to-complex } i2 / 2)) =$ 
of-complex (to-complex (moebius-pt (moebius-rotation  $\varphi$ )  $i2$ ) / 2)
  using unit-circle-set-def by force

  show  $?R 0_h a' a1 a2 x1 x2 i1 i2 u$ 
  using  $**[\text{rule-format, of } ?M a' ?M x1 ?M x2 ?M i1 ?M i2 ?M a1 ?M a2]$   $uf *$ 
  apply (auto simp del: moebius-pt-moebius-rotation-zero moebius-pt-moebius-rotation)
  using  $i1 i2$ 
  by simp
qed
qed
thus ?thesis
  using  $\langle a' \in \text{unit-disc} \rangle \langle x1 \in \text{unit-disc} \rangle \langle x2 \in \text{unit-disc} \rangle \langle x1 \neq x2 \rangle$ 
  using  $\langle \neg \text{poincare-collinear } \{0_h, x1, x2\} \rangle \langle \text{ideal-points } ?lx = \{i1, i2\} \rangle \langle i1 \neq i2 \rangle$ 
  using  $\langle ?a1 \neq ?a2 \rangle \langle \text{poincare-collinear } \{0_h, ?a1, i1\} \rangle \langle \text{poincare-collinear } \{0_h, ?a2, i2\} \rangle$ 
  using  $\langle ?a1 \in \text{unit-disc} \rangle \langle ?a2 \in \text{unit-disc} \rangle \langle i1 \in \text{unit-circle-set} \rangle \langle i2 \in \text{unit-circle-set} \rangle$ 
  using  $\langle \text{poincare-on-ray } a' 0_h a \rangle \langle a' \neq 0_h \rangle \langle \text{poincare-between } ?a1 a ?a2 \rangle \langle a \neq ?a1 \rangle \langle a \neq ?a2 \rangle$ 
  by blast
qed
qed

moreover

have  $\neg \text{poincare-on-line } 0_h ?a1 ?a2$ 
proof
assume  $*$ :  $\text{poincare-on-line } 0_h ?a1 ?a2$ 
hence  $\text{poincare-collinear } \{0_h, ?a1, ?a2\}$ 
  unfolding poincare-on-line-def
  by simp
hence  $\text{poincare-line } 0_h ?a1 = \text{poincare-line } 0_h ?a2$ 
  using poincare-collinear3-poincare-lines-equal-general[of  $0_h ?a1 ?a2$ ]
  using  $\langle ?a1 \in \text{unit-disc} \rangle \langle ?a1 \neq 0_h \rangle \langle ?a2 \in \text{unit-disc} \rangle \langle ?a2 \neq 0_h \rangle$ 
  by (metis inversion-noteq-unit-disc zero-in-unit-disc)

have  $i1 \in \text{circline-set } (\text{poincare-line } 0_h ?a1)$ 
  using  $\langle \text{poincare-collinear } \{0_h, ?a1, i1\} \rangle$ 
  using poincare-collinear3-poincare-line-general[of  $i1 0_h ?a1$ ]
  using  $\langle ?a1 \in \text{unit-disc} \rangle \langle ?a1 \neq 0_h \rangle$ 
  by (metis insert-commute inversion-noteq-unit-disc zero-in-unit-disc)
moreover
have  $i2 \in \text{circline-set } (\text{poincare-line } 0_h ?a1)$ 

```

```

using ⟨poincare-collinear {0h, ?a2, i2}⟩
using poincare-collinear3-poincare-line-general[of i2 0h ?a2]
using ⟨?a2 ∈ unit-disc⟩ ⟨?a2 ≠ 0h⟩ ⟨poincare-line 0h ?a1 = poincare-line 0h ?a2⟩
by (metis insert-commute inversion-noteq-unit-disc zero-in-unit-disc)

```

**ultimately**

```

have poincare-collinear {0h, i1, i2}
  using ⟨?a1 ∈ unit-disc⟩ ⟨?a1 ≠ 0h⟩ ⟨poincare-collinear {0h, ?a1, i1}⟩
  by (smt insert-subset poincare-collinear-def unique-poincare-line zero-in-unit-disc)
hence 0h ∈ circline-set (poincare-line i1 i2)
  using poincare-collinear3-poincare-line-general[of 0h i1 i2]
  using ⟨i1 ≠ i2⟩ ⟨i2 ∈ unit-circle-set⟩ unit-circle-set-def
  by force

```

**moreover**

```

have ?lx = ?li
  using ⟨ideal-points ?lx = {i1, i2}⟩ ⟨x1 ≠ x2⟩ ideal-points-line-unique
  by auto

```

**ultimately**

```

show False
  using ⟨¬ poincare-collinear {0h, x1, x2}⟩
  using ⟨x1 ≠ x2⟩ poincare-line-poincare-collinear3-general
  by auto

```

**qed**

**ultimately**

```

show ?thesis
  using ⟨?a1 ∈ unit-disc⟩ ⟨?a2 ∈ unit-disc⟩
  by blast

```

**qed**

**qed**

**qed**

**thus** ?thesis

```

using ⟨x1 ∈ unit-disc⟩ ⟨x2 ∈ unit-disc⟩ ⟨¬ poincare-collinear {a, x1, x2}⟩
by blast

```

**qed**

## 11.9 Interpretation of locales

**global-interpretation** *PoincareTarskiAbsolute*: *TarskiAbsolute* **where** *cong* = *p-congruent* **and** *betw* = *p-between*

**defines** *p-on-line* = *PoincareTarskiAbsolute.on-line* **and**

*p-on-ray* = *PoincareTarskiAbsolute.on-ray* **and**

*p-in-angle* = *PoincareTarskiAbsolute.in-angle* **and**

*p-ray-meets-line* = *PoincareTarskiAbsolute.ray-meets-line*

**proof**–

**show** *TarskiAbsolute* *p-congruent* *p-between*

**proof**

1. Reflexivity of congruence

**fix** *x y*

**show** *p-congruent* *x y y x*

**unfolding** *p-congruent-def*

**by** *transfer* (*simp* *add*: *poincare-distance-sym*)

**next**

2. Transitivity of congruence

**fix** *x y z u v w*

**show** *p-congruent* *x y z u* ∧ *p-congruent* *x y v w*  $\longrightarrow$  *p-congruent* *z u v w*

**by** (*transfer*, *simp*)

**next**

3. Identity of congruence

**fix**  $x y z$   
**show**  $p\text{-congruent } x y z z \longrightarrow x = y$   
   **unfolding**  $p\text{-congruent-def}$   
   **by**  $\text{transfer (simp add: poincare-distance-eq-0-iff)}$   
**next**

4. Segment construction

**fix**  $x y a b$   
**show**  $\exists z. p\text{-between } x y z \wedge p\text{-congruent } y z a b$   
   **using**  $\text{segment-construction}$   
   **unfolding**  $p\text{-congruent-def}$   
   **by**  $\text{transfer (simp, blast)}$   
**next**

5. Five segment

**fix**  $x y z x' y' z' u u'$   
**show**  $x \neq y \wedge p\text{-between } x y z \wedge p\text{-between } x' y' z' \wedge$   
 $p\text{-congruent } x y x' y' \wedge p\text{-congruent } y z y' z' \wedge$   
 $p\text{-congruent } x u x' u' \wedge p\text{-congruent } y u y' u' \longrightarrow$   
 $p\text{-congruent } z u z' u'$   
   **unfolding**  $p\text{-congruent-def}$   
   **apply**  $\text{transfer}$   
   **using**  $\text{five-segment-axiom}$   
   **by**  $\text{meson}$   
**next**

6. Identity of betweenness

**fix**  $x y$   
**show**  $p\text{-between } x y x \longrightarrow x = y$   
   **by**  $\text{transfer (simp add: poincare-between-sum-distances poincare-distance-eq-0-iff poincare-distance-sym)}$   
**next**

7. Pasch

**fix**  $x y z u v$   
**show**  $p\text{-between } x u z \wedge p\text{-between } y v z \longrightarrow (\exists a. p\text{-between } u a y \wedge p\text{-between } x a v)$   
   **apply**  $\text{transfer}$   
   **using**  $\text{Pasch}$   
   **by**  $\text{blast}$   
**next**

8. Lower dimension

**show**  $\exists a. \exists b. \exists c. \neg p\text{-between } a b c \wedge \neg p\text{-between } b c a \wedge \neg p\text{-between } c a b$   
   **apply**  $(\text{transfer})$   
   **using**  $\text{lower-dimension-axiom}$   
   **by**  $\text{simp}$   
**next**

9. Upper dimension

**fix**  $x y z u v$   
**show**  $p\text{-congruent } x u x v \wedge p\text{-congruent } y u y v \wedge p\text{-congruent } z u z v \wedge u \neq v \longrightarrow$   
 $p\text{-between } x y z \vee p\text{-between } y z x \vee p\text{-between } z x y$   
   **unfolding**  $p\text{-congruent-def}$   
   **by**  $(\text{transfer, simp add: upper-dimension-axiom})$   
**qed**  
**qed**

**interpretation**  $\text{PoincareTarskiHyperbolic: TarskiHyperbolic}$

**where**  $\text{cong} = p\text{-congruent}$  **and**  $\text{betw} = p\text{-between}$

**proof**

10. Euclid negation

**show**  $\exists a b c d t. p\text{-between } a d t \wedge p\text{-between } b d c \wedge a \neq d \wedge$

( $\forall x y. p\text{-between } a b x \wedge p\text{-between } a c y \longrightarrow \neg p\text{-between } x t y$ )

**using** *negated-euclidean-axiom*  
**by** *transfer (auto, blast)*

**next**  
**fix**  $a x1 x2$   
**assume**  $\neg \text{TarskiAbsolute.on-line } p\text{-between } a x1 x2$   
**hence**  $\neg p\text{-on-line } a x1 x2$   
**using** *TarskiAbsolute.on-line-def[OF PoincareTarskiAbsolute.TarskiAbsolute-axioms]*  
**using** *PoincareTarskiAbsolute.on-line-def*  
**by** *simp*

11. Limiting parallels

**thus**  $\exists a1 a2.$   
 $\neg \text{TarskiAbsolute.on-line } p\text{-between } a a1 a2 \wedge$   
 $\neg \text{TarskiAbsolute.ray-meets-line } p\text{-between } a a1 x1 x2 \wedge$   
 $\neg \text{TarskiAbsolute.ray-meets-line } p\text{-between } a a2 x1 x2 \wedge$   
 $(\forall a'. \text{TarskiAbsolute.in-angle } p\text{-between } a' a1 a a2 \longrightarrow \text{TarskiAbsolute.ray-meets-line } p\text{-between } a a' x1 x2)$

**unfolding** *TarskiAbsolute.in-angle-def[OF PoincareTarskiAbsolute.TarskiAbsolute-axioms]*  
**unfolding** *TarskiAbsolute.on-ray-def[OF PoincareTarskiAbsolute.TarskiAbsolute-axioms]*  
**unfolding** *TarskiAbsolute.ray-meets-line-def[OF PoincareTarskiAbsolute.TarskiAbsolute-axioms]*  
**unfolding** *TarskiAbsolute.on-ray-def[OF PoincareTarskiAbsolute.TarskiAbsolute-axioms]*  
**unfolding** *TarskiAbsolute.on-line-def[OF PoincareTarskiAbsolute.TarskiAbsolute-axioms]*  
**unfolding** *PoincareTarskiAbsolute.on-line-def*  
**apply** *transfer*

**proof**–  
**fix**  $a x1 x2$   
**assume**  $*$ :  $a \in \text{unit-disc } x1 \in \text{unit-disc } x2 \in \text{unit-disc}$   
 $\neg (p\text{oincare-between } a x1 x2 \vee p\text{oincare-between } x1 a x2 \vee p\text{oincare-between } x1 x2 a)$   
**hence**  $\neg p\text{oincare-on-line } a x1 x2$   
**using** *poincare-collinear3-iff[of a x1 x2]*  
**using** *poincare-between-rev poincare-on-line-def by blast*  
**hence**  $\exists a1 \in \text{unit-disc}.$   
 $\exists a2 \in \text{unit-disc}.$   
 $\neg p\text{oincare-on-line } a a1 a2 \wedge$   
 $\neg p\text{oincare-ray-meets-line } a a1 x1 x2 \wedge$   
 $\neg p\text{oincare-ray-meets-line } a a2 x1 x2 \wedge$   
 $(\forall a' \in \text{unit-disc}.$   
 $p\text{oincare-in-angle } a' a1 a a2 \longrightarrow$   
 $p\text{oincare-ray-meets-line } a a' x1 x2)$   
**using** *limiting-parallels[of a x1 x2] \**  
**by** *blast*

**then obtain**  $a1 a2$  **where**  $**$ :  $a1 \in \text{unit-disc } a2 \in \text{unit-disc } \neg p\text{oincare-on-line } a a1 a2$   
 $\neg p\text{oincare-ray-meets-line } a a2 x1 x2$   
 $\neg p\text{oincare-ray-meets-line } a a1 x1 x2$   
 $\forall a' \in \text{unit-disc}.$   
 $p\text{oincare-in-angle } a' a1 a a2 \longrightarrow$   
 $p\text{oincare-ray-meets-line } a a' x1 x2$

**by** *blast*

**have**  $\neg (\exists x \in \{z. z \in \text{unit-disc}\}.$   
 $(p\text{oincare-between } a x a1 \vee$   
 $p\text{oincare-between } a a1 x) \wedge$   
 $(p\text{oincare-between } x x1 x2 \vee$   
 $p\text{oincare-between } x1 x x2 \vee$   
 $p\text{oincare-between } x1 x2 x))$   
**using**  $\langle \neg p\text{oincare-ray-meets-line } a a1 x1 x2 \rangle$   
**unfolding** *poincare-on-line-def poincare-ray-meets-line-def poincare-on-ray-def*  
**using** *poincare-collinear3-iff[of - x1 x2] poincare-between-rev \*(2, 3)*  
**by** *auto*

**moreover**  
**have**  $\neg (\exists x \in \{z. z \in \text{unit-disc}\}.$   
 $(p\text{oincare-between } a x a2 \vee$   
 $p\text{oincare-between } a a2 x) \wedge$   
 $(p\text{oincare-between } x x1 x2 \vee$   
 $p\text{oincare-between } x1 x x2 \vee$   
 $p\text{oincare-between } x1 x2 x))$   
**using**  $\langle \neg p\text{oincare-ray-meets-line } a a2 x1 x2 \rangle$

```

unfolding poincare-on-line-def poincare-ray-meets-line-def poincare-on-ray-def
using poincare-collinear3-iff[of - x1 x2] poincare-between-rev *(2, 3)
by auto
moreover
have  $\neg$  (poincare-between a a1 a2  $\vee$  poincare-between a1 a a2  $\vee$  poincare-between a1 a2 a)
using  $\langle \neg$  poincare-on-line a a1 a2  $\rangle$  poincare-collinear3-iff[of a a1 a2]
using *(1) *(1-2)
unfolding poincare-on-line-def
by simp
moreover
have  $(\forall a' \in \{z. z \in \text{unit-disc}\}.$ 
  a  $\neq$  a1  $\wedge$ 
  a  $\neq$  a2  $\wedge$ 
  a'  $\neq$  a  $\wedge$ 
   $(\exists x \in \{z. z \in \text{unit-disc}\}.$ 
    poincare-between a1 x a2  $\wedge$ 
    x  $\neq$  a1  $\wedge$ 
    x  $\neq$  a2  $\wedge$ 
    (poincare-between a a' x  $\vee$ 
     poincare-between a x a'))  $\longrightarrow$ 
   $(\exists x \in \{z. z \in \text{unit-disc}\}.$ 
    (poincare-between a x a'  $\vee$ 
     poincare-between a a' x)  $\wedge$ 
    (poincare-between x x1 x2  $\vee$ 
     poincare-between x1 x x2  $\vee$ 
     poincare-between x1 x2 x)))
```

**using** \*(6)

```

unfolding poincare-on-line-def poincare-in-angle-def poincare-ray-meets-line-def poincare-on-ray-def
using poincare-collinear3-iff[of - x1 x2] poincare-between-rev *(2, 3)
by auto
ultimately
show  $\exists a1 \in \{z. z \in \text{unit-disc}\}.$ 
   $\exists a2 \in \{z. z \in \text{unit-disc}\}.$ 
   $\neg$  (poincare-between a a1 a2  $\vee$  poincare-between a1 a a2  $\vee$  poincare-between a1 a2 a)  $\wedge$ 
   $\neg$  ( $\exists x \in \{z. z \in \text{unit-disc}\}.$ 
    (poincare-between a x a1  $\vee$ 
     poincare-between a a1 x)  $\wedge$ 
    (poincare-between x x1 x2  $\vee$ 
     poincare-between x1 x x2  $\vee$ 
     poincare-between x1 x2 x))  $\wedge$ 
   $\neg$  ( $\exists x \in \{z. z \in \text{unit-disc}\}.$ 
    (poincare-between a x a2  $\vee$ 
     poincare-between a a2 x)  $\wedge$ 
    (poincare-between x x1 x2  $\vee$ 
     poincare-between x1 x x2  $\vee$ 
     poincare-between x1 x2 x))  $\wedge$ 
   $(\forall a' \in \{z. z \in \text{unit-disc}\}.$ 
    a  $\neq$  a1  $\wedge$ 
    a  $\neq$  a2  $\wedge$ 
    a'  $\neq$  a  $\wedge$ 
     $(\exists x \in \{z. z \in \text{unit-disc}\}.$ 
      poincare-between a1 x a2  $\wedge$ 
      x  $\neq$  a1  $\wedge$ 
      x  $\neq$  a2  $\wedge$ 
      (poincare-between a a' x  $\vee$ 
       poincare-between a x a'))  $\longrightarrow$ 
     $(\exists x \in \{z. z \in \text{unit-disc}\}.$ 
      (poincare-between a x a'  $\vee$ 
       poincare-between a a' x)  $\wedge$ 
      (poincare-between x x1 x2  $\vee$ 
       poincare-between x1 x x2  $\vee$ 
       poincare-between x1 x2 x)))
```

**using** \*(1, 2)

**by** *auto*

**qed**

**qed**

**interpretation** *PoincareElementaryTarskiHyperbolic: ElementaryTarskiHyperbolic p-congruent p-between*  
**proof**

12. Continuity

**fix**  $\varphi \psi$   
**assume**  $\exists a. \forall x. \forall y. \varphi x \wedge \psi y \longrightarrow p\text{-between } a \ x \ y$   
**thus**  $\exists b. \forall x. \forall y. \varphi x \wedge \psi y \longrightarrow p\text{-between } x \ b \ y$   
**apply** *transfer*  
**using** *continuity*  
**by** *auto*  
**qed**  
**end**

## References

- [1] R. Coghetto. Klein-Beltrami Model. Part I. *Formalized Mathematics*, 26(1):21–32, 2018.
- [2] R. Coghetto. Klein-Beltrami Model. Part II. *Formalized Mathematics*, 26(1):33–48, 2018.
- [3] N. Lobatschewsky. *Geometrische Untersuchungen zur Theorie der Parallellinien*, pages 159–223. Springer Vienna, Vienna, 1985.
- [4] T. J. M. Makarios. A Mechanical Verification of the Independence of Tarski’s Euclidean Axiom. Master’s thesis, Victoria University of Wellington, 2012. Master Thesis.
- [5] F. Marić and D. Simić. Formalizing Complex Plane Geometry. *Annals of Mathematics and Artificial Intelligence*, 74(3-4):271–308, 2015.
- [6] F. Mari and D. Simi. Complex geometry. *Archive of Formal Proofs*, Dec. 2019. [http://isa-afp.org/entries/Complex\\_Geometry.html](http://isa-afp.org/entries/Complex_Geometry.html), Formal proof development.
- [7] W. Schwabhäuser, W. Szmielew, and A. Tarski. *Metamathematische Methoden in der Geometrie*. Springer-Verlag, Berlin, 1983.
- [8] H. Schwerdtfeger. *Geometry of Complex Numbers: Circle Geometry, Moebius Transformation, Non-euclidean Geometry*. Courier Corporation, 1979.