The independence of Tarski’s Euclidean axiom

T. J. M. Makarios

August 16, 2018

Abstract

Tarski’s axioms of plane geometry are formalized and, using the standard real Cartesian model, shown to be consistent. A substantial theory of the projective plane is developed. Building on this theory, the Klein–Beltrami model of the hyperbolic plane is defined and shown to satisfy all of Tarski’s axioms except his Euclidean axiom; thus Tarski’s Euclidean axiom is shown to be independent of his other axioms of plane geometry.

An earlier version of this work was the subject of the author’s MSc thesis [2], which contains natural-language explanations of some of the more interesting proofs.

Contents

1 Metric and semimetric spaces 2

2 Miscellaneous results 4

3 Tarski’s geometry 11
   3.1 The axioms .............................................. 12
   3.2 Semimetric spaces satisfy the first three axioms .......... 12
   3.3 Some consequences of the first three axioms ............. 13
   3.4 Some consequences of the first five axioms ............... 17
   3.5 Simple theorems about betweenness ..................... 18
   3.6 Simple theorems about congruence and betweenness ...... 20

4 Real Euclidean space and Tarski’s axioms 20
   4.1 Real Euclidean space satisfies the first five axioms .. 20
   4.2 Real Euclidean space also satisfies axioms 6, 7, and 11 . 25
   4.3 Real Euclidean space satisfies the Euclidean axiom .... 30
   4.4 The real Euclidean plane .................................. 31
   4.5 Special cases of theorems of Tarski’s geometry .......... 35

5 Linear algebra 37
   5.1 Matrices .................................................. 40
# 6 Right group actions

# 7 Projective geometry

7.1 Proportionality on non-zero vectors

7.2 Points of the real projective plane

7.3 Lines of the real projective plane

7.4 Collineations of the real projective plane

7.4.1 As a group

7.4.2 As a group action

7.4.3 Parts of some Statements from [1]

7.5 Cross ratios

7.6 Cartesian subspace of the real projective plane

# 8 The hyperbolic plane and Tarski’s axioms

8.1 Characterizing a specific conic in the projective plane

8.2 Some specific points and lines of the projective plane

8.3 Definition of the Klein–Beltrami model of the hyperbolic plane

8.4 $K$-isometries map the interior of the conic to itself

8.5 The $K$-isometries form a group action

8.6 The Klein–Beltrami model satisfies Tarski’s first three axioms

8.7 Some lemmas about betweenness

8.8 The Klein–Beltrami model satisfies axiom 4

8.9 More betweenness theorems

8.10 Perpendicularity

8.11 Functions of distance

8.11.1 A formula for a cross ratio involving a perpendicular foot

8.12 The Klein–Beltrami model satisfies axiom 5

8.13 The Klein–Beltrami model satisfies axioms 6, 7, and 11

8.14 The Klein–Beltrami model satisfies the dimension-specific axioms

8.15 The Klein–Beltrami model violates the Euclidean axiom

# 1 Metric and semimetric spaces
lemma refl [simp]: dist x x = 0
  by simp
end

locale metric =
  fixes dist :: 'p ⇒ 'p ⇒ real
  assumes [simp]: dist x y = 0 ⟷ x = y
  and triangle: dist x z ≤ dist y x + dist y z

sublocale metric < semimetric
proof
  { fix w
    have dist w w = 0 by simp }
  note [simp] = this
  fix x y
  show 0 ≤ dist x y
  proof
    from triangle [of y y x] show 0 ≤ dist x y by simp
    qed
  show dist x y = 0 ⟷ x = y by simp
  show dist x y = dist y x
  proof
    { fix w z
      have dist w z ≤ dist z w
      proof
        from triangle [of w z z] show dist w z ≤ dist z w by simp
        qed
      } hence dist x y ≤ dist y x and dist y x ≤ dist x y by simp+
      thus dist x y = dist y x by simp
    qed
    qed

definition norm-dist :: ('a::real-normed-vector) ⇒ 'a ⇒ real where
  [simp]: norm-dist x y ≜ norm (x - y)

interpretation norm-metric: metric norm-dist
proof
  fix x y
  show norm-dist x y = 0 ⟷ x = y by simp
  fix z
  from norm-triangle-ineq [of x - y y - z] have
    norm (x - z) ≤ norm (x - y) + norm (y - z) by simp
  with norm-minus-commute [of x y] show
    norm-dist x z ≤ norm-dist y x + norm-dist y z by simp
  qed
end
2 Miscellaneous results

theory Miscellany
imports Metric
begin

lemma unordered-pair-element-equality:
  assumes \( \{p, q\} = \{r, s\} \) and \( p = r \)
  shows \( q = s \)
  using assms by (auto simp: doubleton-eq-iff)

lemma unordered-pair-equality: \( \{p, q\} = \{q, p\} \)
  by auto

lemma cosine-rule:
  fixes \( a \) :: 'a::real  
    shows \( (\text{norm-dist} a c)^2 = (\text{norm-dist} a b)^2 + (\text{norm-dist} b c)^2 + 2 \cdot (a \cdot (b - c)) \)
proof
  have \( (a - b) + (b - c) = a - c \) by simp
  with \( \text{dot-norm [of a - b b - c]} \)
  have \( (a - b) \cdot (b - c) = ((\text{norm} (a - c))^2 - (\text{norm} (a - b))^2 - (\text{norm} (b - c))^2) / 2 \)
    by simp
  thus \( \text{thesis} \) by simp
qed

lemma scalar-equiv: \( r \cdot s \cdot x = r \cdot R \cdot x \)
  by vector

lemma norm-dist-dot: \( (\text{norm-dist} x y)^2 = (x - y) \cdot (x - y) \)
  by (simp add: power2-norm-eq-inner)

definition dep2 :: 'a::real-vector \( \Rightarrow \) 'a \( \Rightarrow \) bool where
  \( \text{dep2 } u \ v \triangleq \exists w \ r \ s. \ u = r \cdot R \cdot w \wedge v = s \cdot R \cdot w \)

lemma real2-eq:
  fixes \( u \ v \:: real^2 \)
  assumes \( u\$1 = v\$1 \) and \( u\$2 = v\$2 \)
  shows \( u = v \)
  by (simp add: vec-eq-iff [of u v] forall-2 assms)

definition rotate2 :: real^2 \( \Rightarrow \) real^2 where
  \( \text{rotate2 } x \triangleq \text{vector} [-x\$2, x\$1] \)

declare vector-2 [simp]

lemma rotate2 [simp];
  \( (\text{rotate2 } x)\$1 = -x\$2 \)
\[(\text{rotate2 }x) \odot 2 = x \odot 1\]

by \(\text{simp add: rotate2-def}\)

**Lemma 1:** \text{rotate2-rotate2} [simp]: \(\text{rotate2 } (\text{rotate2 } x) = -x\)

**Proof** -

- have \((\text{rotate2 } (\text{rotate2 } x)) \odot 1 = -x \odot 1\) and \((\text{rotate2 } (\text{rotate2 } x)) \odot 2 = -x \odot 2\)
  by \(\text{simp}\)
- with \(\text{real2-eq}\) show \(\text{rotate2 } (\text{rotate2 } x) = -x\) by \(\text{simp}\)

qed

**Lemma 2:** \text{rotate2-dot} [simp]: \((\text{rotate2 } u) \cdot (\text{rotate2 } v) = u \cdot v\)

unfolding inner-vec-def

- by \(\text{simp add: sum-2}\)

**Lemma 3:** \text{rotate2-scaleR} [simp]: \(\text{rotate2 } (k \times R \cdot x) = k \times R \cdot (\text{rotate2 } x)\)

**Proof** -

- have \((\text{rotate2 } (k \times R \cdot x)) \odot 1 = (k \times R \cdot (\text{rotate2 } x)) \odot 1\) and \((\text{rotate2 } (k \times R \cdot x)) \odot 2 = (k \times R \cdot (\text{rotate2 } x)) \odot 2\) by \(\text{simp}\+)
- with \(\text{real2-eq}\) show \(\text{thesis}\) by \(\text{simp}\)

qed

**Lemma 4:** \text{rotate2-uminus} [simp]: \(\text{rotate2 } (-x) = -(\text{rotate2 } x)\)

**Proof** -

- from \(\text{scaleR-minus-left [of 1]}\) have \(-1 \times R \cdot x = -x\) and \(-1 \times R \cdot (\text{rotate2 } x) = -(\text{rotate2 } x)\) by \(\text{auto}\)
- with \(\text{rotate2-scaleR [of -1 x]}\) show \(\text{thesis}\) by \(\text{simp}\)

qed

**Lemma 5:** \text{rotate2-eq} [iff]: \(\text{rotate2 } x = \text{rotate2 } y \iff x = y\)

**Proof**

- assume \(x = y\)
- thus \(\text{rotate2 } x = \text{rotate2 } y\) by \(\text{simp}\)

next

- assume \(\text{rotate2 } x = \text{rotate2 } y\)
- hence \(\text{rotate2 } (\text{rotate2 } x) = \text{rotate2 } (\text{rotate2 } y)\) by \(\text{simp}\)
- hence \(-(-x) = -(y)\) by \(\text{simp}\)
- thus \(x = y\) by \(\text{simp}\)

qed

**Lemma 6:** \text{dot2-rearrange-1}: \(\text{fixes } u, x :: \text{real-2}\)

- assumes \(u \cdot x = 0\) and \(x \cdot 1 \neq 0\)
- shows \(u = (u \odot 2 / x \odot 1) \times R \cdot (\text{rotate2 } x)\) (is \(u = ?u')\)

**Proof** -

- from \(u \cdot x = 0\) have \(u \odot 1 = -(u \odot 2) \times (x \odot 2)\)
  unfolding inner-vec-def
  by \(\text{simp add: sum-2}\)
- hence \(u \odot 1 = -(u \odot 2) / x \odot 1 = -(u \odot 2) \times x \odot 2\) by \(\text{simp}\)
  with \((x \odot 1 \neq 0)\) have \(u \odot 1 = ?u' \odot 1\) by \(\text{simp}\)
from \((x \neq 0)\) have \(u \neq 0\) by simp

with \((u \neq 0)\) and real2-eq show \(u = u'\) by simp

qed

lemma dot2-rearrange-2:
  fixes \(u, x\) :: \(\mathbb{R}^2\)
  assumes \(u \cdot x = 0\) and \(x \neq 0\)
  shows \(\exists k. u = k \cdot (\text{rotate}_2 x)\)

proof cases
  assume \(x_1 = 0\)
  with real2-eq [of \(x\) 0] and \(\langle x \neq 0 \rangle\) have \(x_2 \neq 0\) by auto
  with dot2-rearrange-1 and \(\langle u \cdot x = 0 \rangle\) show \(?\)thesis by blast

next
  assume \(x_1 \neq 0\)
  with dot2-rearrange-1 and \(\langle u \cdot x = 0 \rangle\) show \(?\)thesis by blast

qed

lemma dot2-rearrange:
  fixes \(u, x\) :: \(\mathbb{R}^2\)
  assumes \(u \cdot x = 0\) and \(x \neq 0\)
  shows \(\exists k. u = k \cdot (\text{rotate}_2 x)\)

proof cases
  assume \(x_1 = 0\)
  with real2-eq [of \(x\) 0] and \(\langle x \neq 0 \rangle\) have \(x_2 \neq 0\) by auto
  with dot2-rearrange-1 and \(\langle u \cdot x = 0 \rangle\) show \(?\)thesis by blast

next
  assume \(x_1 \neq 0\)
  with dot2-rearrange-1 and \(\langle u \cdot x = 0 \rangle\) show \(?\)thesis by blast

qed

lemma real2-orthogonal-dep2:
  fixes \(u, v, x\) :: \(\mathbb{R}^2\)
  assumes \(x \neq 0\) and \(u \cdot x = 0\) and \(v \cdot x = 0\)
  shows \(\text{dep}_2 u v\)

proof
  let \(\hat{w} = \text{rotate}_2 x\)
  from dot2-rearrange and \(\langle u \cdot x = 0 \rangle\) show \(\exists r s. u = r \cdot \hat{w} \land v = s \cdot \hat{w}\) by simp
  with dep2-def show \(?\)thesis by auto

qed

lemma dot-left-diff-distrib:
  fixes \(u, v\) :: \(\mathbb{R}^2\)
  shows \((u - v) \cdot x = (u \cdot x) - (v \cdot x)\)

proof
  have \((u \cdot x) - (v \cdot x) = (\sum_{i \in \text{UNIV.}} u_i \cdot x_i) - (\sum_{i \in \text{UNIV.}} v_i \cdot x_i)\)
    unfolding inner-vec-def
    by simp
  also from sum-subtractf [of \(i\) \(u_i \cdot x_i\) \(v_i \cdot x_i\)] have \(... = (\sum_{i \in \text{UNIV.}} u_i \cdot x_i - v_i \cdot x_i)\) by simp
  also from left-diff-distrib [where \('a\) = real] have \(... = (\sum_{i \in \text{UNIV.}} (u_i - v_i) \cdot x_i)\) by simp

6
also have
\[ \ldots = (u - v) \cdot x \]
unfolding inner-vec-def
by simp
finally show \( ?thesis \).
qued

lemma dot-right-diff-distrib:
fixes \( u, v, x \) :: real
shows \( x \cdot (u - v) = (x \cdot u) - (x \cdot v) \)
proof
from inner-commute have \( x \cdot (u - v) = (u - v) \cdot x \) by auto
also from dot-left-diff-distrib [of \( u, v, x \)] have
\( \ldots = u \cdot x - v \cdot x \).
also from inner-commute [of \( x \)] have
\( \ldots = x \cdot u - x \cdot v \) by simp
finally show \( ?thesis \).
qued

lemma am-gm2:
fixes \( a, b \) :: real
assumes \( a \geq 0 \) and \( b \geq 0 \)
shows \( \sqrt{a \cdot b} \leq (a + b) / 2 \)
and \( \sqrt{a \cdot b} = (a + b) / 2 \leftrightarrow a = b \)
proof
have \( 0 \leq (a - b) \cdot (a - b) \) and \( 0 = (a - b) \cdot (a - b) \leftrightarrow a = b \) by simp+
with right-diff-distrib [of \( a - b, a, b \)] and left-diff-distrib [of \( a, b \)] have
\( 0 \leq a \cdot a - 2 \cdot a \cdot b + b \cdot b \)
and \( 0 = a \cdot a - 2 \cdot a \cdot b + b \cdot b \leftrightarrow a = b \) by auto
hence \( \sqrt{a \cdot a - 2 \cdot a \cdot b + b \cdot b} \leq a \cdot a - 2 \cdot a \cdot b + b \cdot b \)
and \( \sqrt{a \cdot a - 2 \cdot a \cdot b + b \cdot b} = a \cdot a - 2 \cdot a \cdot b + b \cdot b \leftrightarrow a = b \) by auto
with distrib-right [of \( a + b, a, b \)] and distrib-left [of \( a, b \)] have
\( \sqrt{a \cdot a + b} \leq (a + b) \cdot (a + b) \)
and \( \sqrt{a \cdot a + b} = (a + b) \cdot (a + b) \leftrightarrow a = b \) by (simp add: field-simps)+
with real-sqrt-le-mono [of \( 4 \cdot a \cdot b, (a + b) \cdot (a + b) \)]
and real-sqrt-eq-iff [of \( 4 \cdot a \cdot b, (a + b) \cdot (a + b) \)] have
\( \sqrt{4 \cdot a \cdot b} \leq \sqrt{(a + b) \cdot (a + b)} \)
and \( \sqrt{4 \cdot a \cdot b} = \sqrt{(a + b) \cdot (a + b)} \leftrightarrow a = b \) by simp+
with \( a \geq 0 \) and \( b \geq 0 \) have \( \sqrt{4 \cdot a \cdot b} \leq a + b \)
and \( \sqrt{4 \cdot a \cdot b} = a + b \leftrightarrow a = b \) by simp+
with real-sqrt-abs2 [of \( 2 \)] and real-sqrt-mult [of \( a, b \)] show
\( \sqrt{(a + b)} \leq (a + b) / 2 \)
and \( \sqrt{a \cdot b} = (a + b) / 2 \leftrightarrow a = b \) by (simp add: ac-simps)+
qued

lemma refl-on-allrel: refl-on \( A \times A \)
unfolding refl-on-def
by simp

7
lemma refl-on-restrict:
  assumes refl-on A r
  shows refl-on (A ∩ B) (r ∩ B × B)
proof –
  from (refl-on A r) and refl-on-allrel [of B] and refl-on-Int
  show ?thesis by auto
qed

lemma sym-allrel: sym (A × A)
  unfolding sym-def
  by simp

lemma sym-restrict:
  assumes sym r
  shows sym (r ∩ A × A)
proof –
  from (sym r) and sym-allrel and sym-Int
  show ?thesis by auto
qed

lemma trans-allrel: trans (A × A)
  unfolding trans-def
  by simp

lemma equiv-Int:
  assumes equiv A r and equiv B s
  shows equiv (A ∩ B) (r ∩ s)
proof –
  from assms and refl-on-Int [of A B] and sym-Int and trans-Int
  show ?thesis
    unfolding equiv-def
    by auto
qed

lemma equiv-allrel: equiv A (A × A)
  unfolding equiv-def
  by (simp add: refl-on-allrel sym-allrel trans-allrel)

lemma equiv-restrict:
  assumes equiv A r
  shows equiv (A ∩ B) (r ∩ B × B)
proof –
  from (equiv A r) and equiv-allrel [of B] and equiv-Int
  show ?thesis by auto
qed

lemma invertible-times-eq-zero:
  fixes x :: real"n and A :: real"n × n
  assumes invertible A and A *v x = 0
shows \( x = 0 \)
using assms invertible-def matrix-left-invertible-ker by blast

**lemma** times-invertible-eq-zero:
fixes \( x :: \text{real}^n \) and \( A :: \text{real}^{n \times n} \)
assumes invertible \( A \) and \( x \cdot A = 0 \)
shows \( x = 0 \)
using transpose-invertible assms invertible-times-eq-zero by fastforce

**lemma** matrix-id-invertible:
invertible \((\mat 1 :: ('a::semiring-1)^n \times n)\)
by (simp add: invertible-def)

**lemma** Image-refl-on-nonempty:
assumes refl-on \( A \) \( r \) and \( x \in A \)
shows \( x \in r'' \{ x \} \)
proof
from \((\text{refl-on } A) \) and \((x \in A) \) show \((x, x) \in r \)
  unfolding refl-on-def
by simp
qed

**lemma** quotient-element-nonempty:
assumes equiv \( A \) \( r \) and \( X \in A \)
shows \( \exists x. x \in X \)
using assms in-quotient-imp-non-empty by fastforce

**lemma** zero-3: \((3::3) = 0\)
by simp

**lemma** card-suc-ge-insert:
fixes \( A \) and \( x \)
shows \( \text{card } A + 1 \geq \text{card } (\text{insert } x A) \)
using card-insert-le-m1 by fastforce

**lemma** card-le-UNIV:
fixes \( A :: ('n::finite) \) set
shows \( \text{card } A \leq \text{CARD } ('n) \)
by (simp add: card-mono)

**lemma** partition-Image-element:
assumes equiv \( A \) \( r \) and \( X \in A//r \) and \( x \in X \)
shows \( r'' \{ x \} = X \)
by (metis Image-singleton-iff assms equiv-class-eq-iff quotientE)

**lemma** card-insert-ge: \( \text{card } (\text{insert } x A) \geq \text{card } A \)
by (metis card-infinite card-insert-le zero-le)

**lemma** choose-1:
assumes \( \text{card } S = 1 \)
shows \( \exists \ x. \ S = \{x\} \)
using \( \langle \text{card } S = 1 \rangle \) and \( \text{card-eq-SucD \ [of \ S \]} \)
by simp

**lemma choose-2:**
assumes \( \text{card } S = 2 \)
shows \( \exists \ x \ y. \ S = \{x,y\} \)
proof –
from \( \langle \text{card } S = 2 \rangle \) and \( \text{card-eq-SucD \ [of \ S \]} \)
obtain \( x \) and \( T \) where \( S = \text{insert } x \ T \) and \( \text{card } T = 1 \) by auto
from \( \langle \text{card } T = 1 \rangle \) and \( \text{choose-1 \ [of \ T \]} \) obtain \( y \) where \( T = \{y\} \) by auto
with \( \langle S = \text{insert } x \ T \rangle \) have \( S = \{x,y\} \) by simp
thus \( \exists \ x \ y. \ S = \{x,y\} \) by auto
qed

**lemma choose-3:**
assumes \( \text{card } S = 3 \)
shows \( \exists \ x \ y \ z. \ S = \{x,y,z\} \)
proof –
from \( \langle \text{card } S = 3 \rangle \) and \( \text{card-eq-SucD \ [of \ S \]} \)
obtain \( x \) and \( T \) where \( S = \text{insert } x \ T \) and \( \text{card } T = 2 \) by auto
from \( \langle \text{card } T = 2 \rangle \) and \( \text{choose-2 \ [of \ T \]} \) obtain \( y \) and \( z \) where \( T = \{y,z\} \) by auto
with \( \langle S = \text{insert } x \ T \rangle \) have \( S = \{x,y,z\} \) by simp
thus \( \exists \ x \ y \ z. \ S = \{x,y,z\} \) by auto
qed

**lemma card-gt-0-diff-singleton:**
assumes \( \text{card } S > 0 \) and \( x \in S \)
shows \( \text{card } (S - \{x\}) = \text{card } S - 1 \)
proof –
from \( \langle \text{card } S > 0 \rangle \) have finite \( S \) by (rule \( \text{card-ge-0-finite} \))
with \( \langle x \in S \rangle \)
show \( \text{card } (S - \{x\}) = \text{card } S - 1 \) by (simp add: \( \text{card-Diff-singleton} \))
qed

**lemma eq-3-or-of-3:**
fixes \( j :: 4 \)
shows \( j = 3 \lor (\exists \ j' :: 3. \ j = \text{of-int } (\text{Rep-bit1 } j')) \)
proof (induct \( j \))
fix \( j \)-int :: int
assume \( 0 \leq j \)-int
assume \( j \)-int < int \( \text{CARD(4)} \)
hence \( j \)-int \leq 3 \) by simp

show \( \text{of-int } j \)-int = \( 3 :: 4 \) \lor (\exists \ j' :: 3. \ \text{of-int } j \)-int = \( \text{of-int } (\text{Rep-bit1 } j') \))
proof cases
assume \( j \)-int = 3
thus
\[ \text{of-int } j\text{-int} = (3::4) \lor (\exists j'::3. \text{of-int } j\text{-int} = \text{of-int } (\text{Rep-bit1 } j')) \]
by simp

next
assume \( j\text{-int} \neq 3 \)
with \( j\text{-int} \leq 3 \) have \( j\text{-int} < 3 \) by simp
with \( 0 \leq j\text{-int} \) have \( j\text{-int} \in \{0..<3\} \) by simp
hence \( \text{Rep-bit1 } (\text{Abs-bit1 } j\text{-int} :: 3) = j\text{-int} \)
by (simp add: bit1.Abs-inverse)

hence \( \text{of-int } j\text{-int} = \text{of-int } (\text{Rep-bit1 } (\text{Abs-bit1 } j\text{-int} :: 3)) \) by simp
thus
\[ \text{of-int } j\text{-int} = (3::4) \lor (\exists j'::3. \text{of-int } j\text{-int} = \text{of-int } (\text{Rep-bit1 } j')) \]
by auto

qed

qed

lemma **sgn-plus**: fixes \( x \) \( y \) :: 'a::linordered-idom
assumes \( \text{sgn } x = \text{sgn } y \)
shows \( \text{sgn } (x + y) = \text{sgn } x \)
by (simp add: assms same-sgn-sgn-add)

lemma **sgn-div**: fixes \( x \) \( y \) :: 'a::linordered-field
assumes \( y \neq 0 \) and \( \text{sgn } x = \text{sgn } y \)
shows \( x / y > 0 \)
using assms sgn-1-pos sgn-eq-0-iff by fastforce

lemma **abs-plus**: fixes \( x \) \( y \) :: 'a::linordered-idom
assumes \( \text{sgn } x = \text{sgn } y \)
shows \( |x + y| = |x| + |y| \)
by (simp add: assms same-sgn-abs-add)

lemma **sgn-plus-abs**: fixes \( x \) \( y \) :: 'a::linordered-idom
assumes \( |x| > |y| \)
shows \( \text{sgn } (x + y) = \text{sgn } x \)
by (cases \( x > 0 \)) (use assms in auto)

end

3 Tarski’s geometry

theory Tarski
imports Complex-Main Miscellany Metric
begin

11
3.1 The axioms

The axioms, and all theorems beginning with \( th \) followed by a number, are based on corresponding axioms and theorems in [3].

locale tarski-first3 =  
  fixes \( C :: \'p \Rightarrow \'p \Rightarrow \'p \Rightarrow \'p \Rightarrow \text{bool} \)  
  assumes A1: : \( \forall a b. a b \equiv b a \)  
  and A2: : \( \forall a b p q r s. a b \equiv p q \land a b \equiv r s \rightarrow p q \equiv r s \)  
  and A3: : \( \forall a b c. a b \equiv c c \rightarrow a = b \)

locale tarski-first5 = tarski-first3 +  
  fixes \( B :: \'p \Rightarrow \'p \Rightarrow \'p \Rightarrow \text{bool} \)  
  assumes A4: : \( \forall a b c. \exists x. B q a x \land a x \equiv b c \)  
  and A5: : \( \forall a b c d a' b' c' d'. a \neq b \land B a b c \land B a' b' c' \land a d \equiv a' d' \land \)  
  \( b d \equiv b' d' \rightarrow c d \equiv c' d' \)

locale tarski-absolute-space = tarski-first5 +  
  assumes A6: : \( \forall a b. B a b a \rightarrow a = b \)  
  and A7: : \( \forall a b c p q. B a p c \land B b q c \rightarrow (\exists x. B p x b \land B q x a) \)  
  and A11: : \( \forall x y. (\exists a. \forall x y. x \in X \land y \in Y \rightarrow B a x y) \rightarrow (\exists b. \forall x y. x \in X \land y \in Y \rightarrow B x b y) \)

locale tarski-absolute = tarski-absolute-space +  
  assumes A8: : \( \exists a b c. \neg B a b c \land \neg B b c a \land \neg B c a b \)  
  and A9: : \( \forall p q a b c. p \neq q \land a p \equiv a q \land B p q c \equiv B q p c \equiv c q \rightarrow B a b c \lor B b c a \lor B c a b \)

locale tarski-space = tarski-absolute-space +  
  assumes A10: : \( \forall a b c d t. B a d t \land B b d c \land a \neq d \rightarrow (\exists x y. B a b x \land B a c y \land B x t y) \)

locale tarski = tarski-absolute + tarski-space

3.2 Semimetric spaces satisfy the first three axioms

context semimetric

begin
  definition smC :: \'p \Rightarrow \'p \Rightarrow \'p \Rightarrow \text{bool} \)  
  where simp: : \( a b \equiv c d \equiv \text{dist} a b = \text{dist} c d \)
end

sublocale semimetric < tarski-first3 smC

proof
  from symm show \( \forall a b. a b \equiv \text{by simp} \)
  show \( \forall a b p q r s. a b \equiv p q \land a b \equiv r s \rightarrow p q \equiv r s \) by simp
  show \( \forall a b c. a b \equiv c c \rightarrow a = b \) by simp
qed
3.3 Some consequences of the first three axioms

class context tarski-first3

begin

lemma A1': \( a \equiv b a \)
  by (simp add: A1)

lemma A2': \( [a \equiv p q; a \equiv r s] \Longrightarrow p \equiv r s \)
  proof
    assume a \equiv p q and a \equiv r s
    with A2 show ?thesis by blast
  qed

lemma A3': \( a \equiv c c \Longrightarrow a \equiv b \)
  by (simp add: A3)

theorem th2-1: \( a \equiv a b \)
  proof
    from A2' [of b a a b a b] and A1' [of a b] show ?thesis by simp
  qed

theorem th2-2: \( a \equiv c d \Longrightarrow c \equiv a b \)
  proof
    assume a \equiv c d
    with A2' [of a b c d a b] and th2-1 [of a b] show ?thesis by simp
  qed

theorem th2-3: \( [a \equiv c d; c \equiv e f] \Longrightarrow a \equiv c d \equiv e f \)
  proof
    assume a \equiv c d
    with th2-2 [of a b c d] have c \equiv a b by simp
    assume c \equiv e f
    with A2' [of c d a b e f] and \( c \equiv a b \) show ?thesis by simp
  qed

theorem th2-4: \( a \equiv c d \Longrightarrow b \equiv a c d \)
  proof
    assume a \equiv c d
    with th2-3 [of a b a b c d] and A1' [of b a] show ?thesis by simp
  qed

theorem th2-5: \( a \equiv c d \Longrightarrow a \equiv d c \)
  proof
    assume a \equiv c d
    with th2-3 [of a b c d d c] and A1' [of c d] show ?thesis by simp
  qed

definition is-segment :: 'p set \Rightarrow bool where
  is-segment X \( \triangleq \exists x y. X = \{x, y\} \)


definition segments :: 'p set set where
segments = {X. is-segment X}

definition SC :: 'p set ⇒ 'p set ⇒ bool where
SC X Y ≜ ∃ w x y z. X = {w, x} ∧ Y = {y, z} ∧ w x ≡ y z

definition SC-rel :: ('p set × 'p set) set where
SC-rel = { (X, Y). X Y. SC X Y }

lemma left-segment-congruence:
assumes {a, b} = {p, q} and p q ≡ c d
shows a b ≡ c d
proof cases
  assume a = p
  with unordered-pair-element-equality [of a b p q] and ⟨{a, b} = {p, q}⟩
  have b = q by simp
  with ⟨p q ≡ c d⟩ and ⟨a = p⟩ show ?thesis by simp
next
  assume a ≠ p
  with ⟨{a, b} = {p, q}⟩ have a = q by auto
  with unordered-pair-element-equality [of a b q p] and ⟨{a, b} = {p, q}⟩
  have b = p by auto
  with ⟨p q ≡ c d⟩ and ⟨a = q⟩ have b a ≡ c d by simp
  with th2-4 [of b a c d] show ?thesis by simp
qed

lemma right-segment-congruence:
assumes {c, d} = {p, q} and a b ≡ p q
shows a b ≡ c d
proof
  from th2-2 [of a b p q] and ⟨a b ≡ p q⟩ have p q ≡ a b by simp
  with left-segment-congruence [of c d p q a b] and ⟨{c, d} = {p, q}⟩
  have c d ≡ a b by simp
  with th2-2 [of c d a b] show ?thesis by simp
qed

lemma C-SC-equiv: a b ≡ c d = SC {a, b} {c, d}
proof
  assume a b ≡ c d
  with SC-def [of {a, b} {c, d}] show SC {a, b} {c, d} by auto
next
  assume SC {a, b} {c, d}
  with SC-def [of {a, b} {c, d}]
  obtain w x y z where {a, b} = {w, x} and {c, d} = {y, z} and w x ≡ y z
  by blast
  from left-segment-congruence [of a b w x y z] and
  ⟨{a, b} = {w, x}⟩ and ⟨w x ≡ y z⟩
  have a b ≡ y z by simp

14
with right-segment-congruence [of c d y z a b] and \{(c, d) = \{y, z\}\}
show a b \equiv c d by simp
qed

lemmas SC-refl = th2-1 [simplified]

lemma SC-rel-refl: refl on segments SC-rel
proof –
  note refl-on-def [of segments SC-rel]
  moreover
  { fix Z
    assume Z \in SC-rel
    with SC-refl obtain X Y where Z = (X, Y) and SC X Y by auto
    from \langle SC X Y \rangle and SC-def [of X Y]
    have \exists w x. X = \{w, x\} and \exists y z. Y = \{y, z\} by auto
    with is-segment-def [of X] and is-segment-def [of Y]
    have is-segment X and is-segment Y by auto
    with segments-def have X \in segments and Y \in segments by auto
    with \langle Z = (X, Y) \rangle have Z \in segments \times segments by simp }
  hence SC-rel \subseteq segments \times segments by auto
  moreover
  { fix X
    assume X \in segments
    with segments-def have is-segment X by auto
    with is-segment-def [of X] obtain x y where X = \{x, y\} by auto
    with SC-def [of X X] and SC-refl have SC X X by (simp add: C-SC-equiv)
    with SC-refl-def have (X, X) \in SC-rel by simp }
  hence \forall X. X \in segments \rightarrow (X, X) \in SC-rel by simp
  ultimately show \?thesis by simp
qed

lemma SC-sym:
  assumes SC X Y
  shows SC Y X
proof –
  from SC-def [of X Y] and \langle SC X Y \rangle
  obtain w x y z where X = \{w, x\} and Y = \{y, z\} and w x \equiv y z
  by auto
  from th2-2 [of w x y z] and \langle w x \equiv y z \rangle have y z \equiv w x by simp
  with SC-def [of Y X] and \langle X = \{w, x\} \rangle and \langle Y = \{y, z\} \rangle
  show SC Y X by (simp add: C-SC-equiv)
qed

lemma SC-sym': SC X Y = SC Y X
proof
  assume SC X Y
  with SC-sym [of X Y] show SC Y X by simp
next
  assume SC Y X
with SC-sym [of Y X] show SC X Y by simp
qed

lemma SC-rel-sym: sym SC-rel
proof −
{ fix X Y
  assume (X, Y) ∈ SC-rel
  with SC-rel-def have SC X Y by simp
  with SC-sym have SC Y X by simp
  with SC-rel-def have (Y, X) ∈ SC-rel by simp }
with sym-def [of SC-rel] show ?thesis by blast
qed

lemma SC-trans:
assumes SC X Y and SC Y Z
shows SC X Z
proof −
from SC-def [of X Y] and ⟨SC X Y⟩
obtain w x y z where X = {w, x} and Y = {y, z} and w x ≡ y z
by auto
from SC-def [of Y Z] and ⟨SC Y Z⟩
obtain p q r s where Y = {p, q} and Z = {r, s} and p q ≡ r s by auto
from ⟨Y = {y, z}⟩ and ⟨Y = {p, q}⟩ and ⟨p q ≡ r s⟩
have y z ≡ r s by (simp add: C-SC-equiv)
with th2-3 [of w x y z r s] and ⟨w x ≡ y z⟩ have w x ≡ r s by simp
with SC-def [of X Z] and ⟨X = {w, x}⟩ and ⟨Z = {r, s}⟩
show SC X Z by (simp add: C-SC-equiv)
qed

lemma SC-rel-trans: trans SC-rel
proof −
{ fix X Y Z
  assume (X, Y) ∈ SC-rel and (Y, Z) ∈ SC-rel
  with SC-rel-def have SC X Y and SC Y Z by auto
  with SC-trans [of X Y Z] have SC X Z by simp
  with SC-rel-def have (X, Z) ∈ SC-rel by simp }
with trans-def [of SC-rel] show ?thesis by blast
qed

lemma A3-reversed:
assumes a a ≡ b c
shows b = c
proof −
from ⟨a a ≡ b c⟩ have b c ≡ a a by (rule th2-2)
thus b = c by (rule A3)
qed

lemma equiv-segments-SC-rel: equiv segments SC-rel
by (simp add: equiv-def SC-rel-refl SC-rel-sym SC-rel-trans)
3.4 Some consequences of the first five axioms

context tarski-first5
begin

lemma A4': \exists x. B q a x \land a x \equiv b c
  by (simp add: A4 [simplified])

theorem th2-8: a a \equiv b b
proof -
  from A4' [of - a b b] obtain x where a x \equiv b b by auto
  with A3' [of a x b] have x = a by simp
  with \langle a x \equiv b b \rangle show ?thesis by simp
qed

definition OFS :: ![p',p',p',p',p',p',p'] \Rightarrow bool where
  OFS a b c d a' b' c' d' \equiv
  B a b c \land B a' b' c' \land a b \equiv a' b' \land b c \equiv b' c' \land a d \equiv a' d' \land b d \equiv b' d'

lemma A5': [OFS a b c d a' b' c' d'; a \neq b] \Longrightarrow c d \equiv c' d'
proof -
  assume OFS a b c d a' b' c' d' and a \neq b
  with A5 and OFS-def show ?thesis by blast
qed

theorem th2-11:
  assumes hypotheses:
    B a b c
    B a' b' c'
    a b \equiv a' b'
    b c \equiv b' c'
  shows a c \equiv a' c'
proof cases
  assume a = b
  with \langle a b \equiv a' b' \rangle have a' = b' by (simp add: A3-reversed)
  with \langle b c \equiv b' c' \rangle and \langle a = b \rangle show ?thesis by simp
next
  assume a \neq b
  moreover
  note A5' [of a b c a' b' c' a'] and
  unordered-pair-equality [of a c] and
  unordered-pair-equality [of a' c']
moreover
  from OFS-def [of a b c a' b' c' a'] and
  hypotheses and
  th2-8 [of a a'] and
  unordered-pair-equality [of a b] and
unordered-pair-equality \[ \text{(of } a' b') \]

have \( \text{OFS } a b c a' b' c' a' \) by (simp add: C-SC-eqv)

ultimately show ?thesis by (simp add: C-SC-eqv)

qed

lemma A4-unique:

assumes \( q \neq a \) and \( B q a x \) and \( a x \equiv b c \)

and \( B q a x' \) and \( a x' \equiv b c \)

shows \( x = x' \)

proof (assume SC-sym' and SC-trans and C-SC-eqv and \( a x \equiv b c \))

have \( a x \equiv a x' \) by blast

with th2-11 \([ q a x q a x'] \) and \( B q a x \) and \( B q a x' \) and SC-refl

have \( q x \equiv q x' \) by simp

with OFS-def \([ q a x q a x'] \) and \( B q a x \) and \( B q a x' \) and SC-refl

have \( q x \equiv q x' \) by simp

with A5' \([ q a x q a x'] \) and \( q \neq a \) have \( x x \equiv x x' \) by simp

thus \( x = x' \) by (rule A3-reversed)

qed

theorem th2-12:

assumes \( q \neq a \)

shows \( \exists! x. B q a x \land a x \equiv b c \)

using \( q \neq a \) and A4' and A4-unique

by blast

end

3.5 Simple theorems about betweenness

theorem (in tarski-first5) th3-1: \( B a b b \)

proof (assume A4 \([ \text{rule-format, of } a b b b ] \) obtain \( x \) where \( B a b x \) and \( b x \equiv b b \) by auto)

from A3 \([ \text{rule-format, of } b x b ] \) and \( b x \equiv b b \) have \( b = x \) by simp

with \( B a b x \) show \( B a b b \) by simp

qed

context tarski-absolute-space

begin

lemma A6' (assume A6 \([ \text{rule-format, of } a b b b ] \) obtain \( x \) where \( B a b x \) and \( b x \equiv b b \) by auto)

from A3 \([ \text{rule-format, of } b x b ] \) and \( b x \equiv b b \) have \( b = x \) by simp

with \( B a b x \) show \( B a b b \) by simp

qed
lemma \( A7' \):
assumes \( B \ a \ p \ c \) and \( B \ b \ q \ c \)
shows \( \exists \ x. \ B \ p \ x \ b \land B \ q \ x \ a \)
proof –
from \( A7 \) and \( \langle B \ a \ p \ c \rangle \) and \( \langle B \ b \ q \ c \rangle \) show thesis by blast
qed

lemma \( A11' \):
assumes \( \forall \ x \ y. \ x \in X \land y \in Y \longrightarrow B \ a \ x \ y \)
shows \( \exists \ b. \ \forall \ x \ y. \ x \in X \land y \in Y \longrightarrow B \ x \ b \ y \)
proof –
from asm have \( \exists \ a. \ \forall \ x \ y. \ x \in X \land y \in Y \longrightarrow B \ a \ x \ y \) by (rule exI)
thus \( \exists \ b. \ \forall \ x \ y. \ x \in X \land y \in Y \longrightarrow B \ x \ b \ y \) by (rule \( A11 \) [rule-format])
qed

theorem \( \text{th3-2} \):
assumes \( B \ a \ b \ c \)
shows \( B \ c \ b \ a \)
proof –
from \( \text{th3-1} \) have \( B \ b \ c \ c \) by simp
with \( A7' \) and \( \langle B \ a \ b \ c \rangle \) obtain \( x \) where \( B \ b \ x \ b \) and \( B \ c \ x \ a \) by blast
from \( A6' \) and \( \langle B \ b \ x \ b \rangle \) have \( x = b \) by auto
with \( \langle B \ c \ x \ a \rangle \) show \( B \ c \ b \ a \) by simp
qed

theorem \( \text{th3-4} \):
assumes \( B \ a \ b \ c \) and \( B \ b \ a \ c \)
shows \( a = b \)
proof –
from \( \langle B \ a \ b \ c \rangle \) and \( \langle B \ b \ a \ c \rangle \) and \( A7' \) [of \( a \ b \ c \ b \ a \)]
obtain \( x \) where \( B \ b \ x \ b \) and \( B \ a \ x \ a \) by auto
hence \( b = x \) and \( a = x \) by (simp-all add: \( A6' \))
thus \( a = b \) by simp
qed

theorem \( \text{th3-5-1} \):
assumes \( B \ a \ b \ d \) and \( B \ b \ c \ d \)
shows \( B \ a \ b \ c \)
proof –
from \( \langle B \ a \ b \ d \rangle \) and \( \langle B \ b \ c \ d \rangle \) and \( A7' \) [of \( a \ b \ d \ b \ c \)]
obtain \( x \) where \( B \ b \ x \ b \) and \( B \ c \ x \ a \) by auto
from \( \langle B \ b \ x \ b \rangle \) have \( b = x \) by (rule \( A6' \))
with \( \langle B \ c \ x \ a \rangle \) have \( B \ c \ b \ a \) by simp
thus \( B \ a \ b \ c \) by (rule \( \text{th3-2} \))
qed

theorem \( \text{th3-6-1} \):
assumes \( B \ a \ b \ c \) and \( B \ a \ c \ d \)
shows $B\, b\, c\, d$
proof −
from $\langle B\, a\, c\, d \rangle$ and $\langle B\, a\, b\, c \rangle$ and th3-2 have $B\, d\, c\, a$ and $B\, c\, b\, a$ by fast+
hence $B\, d\, c\, b$ by (rule th3-5-1)
thus $B\, b\, c\, d$ by (rule th3-2)
qed

theorem th3-7-1:
assumes $b \neq c$ and $B\, a\, b\, c$ and $B\, b\, c\, d$
shows $B\, a\, c\, d$
proof −
from $A4'$ obtain $x$ where $B\, a\, c\, x$ and $c\, x \equiv c\, d$ by fast
from $\langle B\, a\, b\, c \rangle$ and $\langle B\, a\, c\, x \rangle$ have $B\, b\, c\, x$ by (rule th3-6-1)
have $c\, d \equiv c\, d$ by (rule th2-1)
with $\langle b \neq c \rangle$ and $\langle B\, b\, c\, x \rangle$ and $\langle c\, x \equiv c\, d \rangle$ and $\langle B\, b\, c\, d \rangle$
have $x = d$ by (rule A4-unique)
with $\langle B\, a\, c\, x \rangle$ show $B\, a\, c\, d$ by simp
qed

theorem th3-7-2:
assumes $b \neq c$ and $B\, a\, b\, c$ and $B\, b\, c\, d$
shows $B\, a\, b\, d$
proof −
from $\langle B\, b\, c\, d \rangle$ and $\langle B\, a\, b\, c \rangle$ and th3-2 have $B\, d\, c\, b$ and $B\, c\, b\, a$ by fast+
with $\langle b \neq c \rangle$ and th3-7-1 [of $c\, b\, d\, a$] have $B\, d\, b\, a$ by simp
thus $B\, a\, b\, d$ by (rule th3-2)
qed

3.6 Simple theorems about congruence and betweenness

definition (in tarski-first5) Col :: ‘p ⇒ ‘p ⇒ ‘p ⇒ bool where
Col a b c ≜ B a b c ∨ B b c a ∨ B c a b

end

4 Real Euclidean space and Tarski’s axioms

theory Euclid-Tarski
imports Tarski
begin

4.1 Real Euclidean space satisfies the first five axioms

abbreviation
real-euclid-C :: [real¨(’n::finite), real¨(’n), real¨(’n), real¨(’n)] ⇒ bool
(- - ≡R - - [99,99,99,99] 50) where
real-euclid-C ≜ norm-metric.smC

20
definition real-euclid-B ⊆ [real^('n::finite), real^('n), real^('n)] ⇒ bool
(B_R - - [99.99,99.99] 50) where
B_R a b c ≜ ∃ l. 0 ≤ l ∧ l ≤ 1 ∧ b − a = l *R (c − a)

interpretation real-euclid: tarski-first5 real-euclid-C real-euclid-B

proof
By virtue of being a semimetric space, real Euclidean space is already known to satisfy the first three axioms.

{ fix q a b c
have ∃x. B_R q a x ∧ a x ≡I_R b c
proof cases
assume q = a
let ?x = a + c − b
have B_R q a ?x
proof
let ?l = 0 :: real
note real-euclid-B-def [of q a ?x]
moreover
have ?l ≥ 0 and ?l ≤ 1 by auto
moreover
from q = a have a − q = 0 by simp
hence a − q = ?l *R (?x − q) by simp
ultimately show ?thesis by auto
qed
moreover
have a − ?x = b − c by simp
hence a ?x ≡I_R b c by (simp add: field-simps)
ultimately show ?thesis by blast
next
assume q ≠ a
hence norm-dist q a > 0 by simp
let ?k = norm-dist b c / norm-dist q a
let ?x = a + ?k *R (a − q)
have B_R q a ?x
proof
let ?l = 1 / (1 + ?k)
have ?l > 0 by (simp add: add-pos-nonneg)
note real-euclid-B-def [of q a ?x]
moreover
have ?l ≥ 0 and ?l ≤ 1 by (auto simp add: add-pos-nonneg)
moreover
from scaleR-left-distrib [of 1 ?k a − q]
have (1 + ?k) *R (a − q) = ?l *R (?x − q) by simp
hence ?l *R ((1 + ?k) *R (a − q)) = ?l *R (?x − q) by simp
with ⟨?l > 0 ⟨scaleR-right-diff-distrib [of ?l ?x q]
have a − q = ?l *R (?x − q) by simp
ultimately show B_R q a ?x by blast
qed
moreover

have a ?x ≡_R b c

proof -

  from norm-scaleR [of ?k a − q] have
  norm-dist a ?x = |?k| * norm (a − q) by simp

  also have
  ... = ?k * norm (a − q) by simp

  also from norm-metric.symm [of q a] have
  ... = ?k * norm-dist q a by simp

finally have
  norm-dist a ?x = norm-dist b c / norm-dist q a
  by simp

with ⟨norm-dist q a > 0⟩ show a ?x ≡_R b c by auto

qed

ultimately show ?thesis by blast

qed }

thus ∀ q a b c. ∃ x. B_I R q a x ∧ a x ≡_R b c by auto

{ fix a b c d a' b' c' d'

  assume a ≠ b and
  B_I R a b c and
  B_I R a' b' c' and
  a b ≡_R a' b' and
  b c ≡_R b' c' and
  a d ≡_R a' d' and
  b d ≡_R b' d'

  have c d ≡_R c' d'

proof -

  { fix m
    fix p q r :: realˆ\langle n::finite⟩

    assume 0 ≤ m and
    m ≤ 1 and
    p ≠ q and
    q − p = m *_R (r − p)

from (p ≠ q) and (q − p = m *_R (r − p)) have m ≠ 0

proof -

  { assume m = 0
    with (q − p = m *_R (r − p)) have q − p = 0 by simp
    with (p ≠ q) have False by simp }

thus ?thesis ..

qed

with ⟨m ≥ 0⟩ have m > 0 by simp

from (q − p = m *_R (r − p)) and

scaleR-right-diff-distrib [of m r p]

have q − p = m *_R r − m *_R p by simp

hence q − p − q + p − m *_R r =

m *_R r − m *_R p − q + p − m *_R r

by simp

with scaleR-left-diff-distrib [of 1 m p] and

scaleR-left-diff-distrib [of 1 m q]

have (1 − m) *_R p − (1 − m) *_R q = m *_R q − m *_R r by auto

with scaleR-right-diff-distrib [of 1 − m p q] and


scaleR-right-diff-distrib \[\text{of } m \ q \ r\]

have \((1 - m) \cdot_R (p - q) = m \cdot_R (q - r)\) by simp

with norm-scaleR \[\text{of } 1 - m \ p - q\] and norm-scaleR \[\text{of } m \ q - r\]

have \(|1 - m| \cdot_R \|p - q\| = |m| \cdot_R \|q - r\|\) by simp

with \(|m| > 0\) and \(|m| \leq 1\)

have \(\|q - r\| = (1 - m) / \|m\| \cdot_R \|p - q\|\) by simp

moreover from: \(p \neq q\) have \(\|p - q\| \neq 0\) by simp

ultimately

have \(\|q - r\| / \|p - q\| = (1 - m) / \|m\|\) by simp

with \(|m| \neq 0\) have

\(\|\) norm-dist \(q \) \(/ / \|p - q\| = (1 - m) / \|m\|\) and \(|m| \neq 0\) by auto \}

note linelemma = this

from real-euclid-B-def \[\text{of } a \ b \ c\] and \(\langle BR \ a \ b \ c\)

obtain \(l\) where \(0 \leq l\) and \(l \leq 1\) and \(b - a = l \cdot_R (c - a)\) by auto

from real-euclid-B-def \[\text{of } a' \ b' \ c'\] and \(\langle BR \ a' \ b' \ c'\)

obtain \(l'\) where \(0 \leq l'\) and \(l' \leq 1\) and \(b' - a' = l' \cdot_R (c' - a')\) by auto

from \(\langle a \neq b\rangle\) and \(\langle a \equiv_R a' \ b' \rangle\) have \(a' \neq b'\) by auto

from linelemma \[\text{of } l \ a \ b \ c\] and

\(\langle l \geq 0\rangle\) and
\(\langle l \leq 1\rangle\) and
\(\langle a \neq b\rangle\) and
\(\langle b - a = l \cdot_R (c - a)\rangle\)

have \(l \neq 0\) and \((1 - l) / l = \|\) norm-dist \(b \ c\) \(/ / \|\) norm-dist \(a \ b\) by auto

from \(\langle (1 - l) / l = \|\) norm-dist \(b \ c\) \(/ / \|\) norm-dist \(a \ b\rangle\) and

\(\langle a \equiv_R a' \ b' \rangle\) and
\(\langle b \ c \equiv_R b' \ c'\rangle\)

have \((1 - l) / l = \|\) norm-dist \(b' \ c'\) \(/ / \|\) norm-dist \(a' \ b'\) by simp

with linelemma \[\text{of } l' \ a' \ b' \ c'\] and

\(\langle l' \geq 0\rangle\) and
\(\langle l' \leq 1\rangle\) and
\(\langle a' \neq b'\rangle\) and
\(\langle b' - a' = l' \cdot_R (c' - a')\rangle\)

have \(l' \neq 0\) and \((1 - l') / l' = (1 - l') / l'\) by auto

from \(\langle (1 - l') / l' = (1 - l') / l'\rangle\) and \(\langle l' \neq 0\rangle\) have \((1 - l') / l' = (1 - l') / l'\) by simp

with left-diff-distrib \[\text{of } 1 \ l \ l'\] and left-diff-distrib \[\text{of } 1 \ l \ l'\]

have \(l = l'\) by simp

\{ fix \(m\)

fix \(p \ q \ r \ s::\ real\langle\langle int\rangle\rangle\langle\langle int\rangle\rangle\)

assume \(|m| \neq 0\) and
\(q - p = m \cdot_R (r - p)\)

with scaleR-scaleR have \(r - p = (1/m) \cdot_R (q - p)\) by simp

with cosine-rule \[\text{of } r \ s \ p\]

have \(\|\) norm-dist \(r \ s\|^2 = \|\) norm-dist \(r \ p\|^2 + \|\) norm-dist \(p \ s\|^2 + 2 * ((1/m) \cdot_R (q - p)) \cdot_R (p - s)\) by simp

also from inner-scaleR-left \[\text{of } 1/m \ q \ - \ p \ p \ - \ s\]

have \(\ldots\) =

23
(\text{nrm-dist}\ r\ p)^2 + (\text{nrm-dist}\ p\ s)^2 + 2/m \ast ((q - p) \cdot (p - s))

by simp

also from \langle m \neq 0 \rangle \text{ and cosine-rule [of q s p]}

have \ldots = (\text{nrm-dist}\ r\ p)^2 + (\text{nrm-dist}\ p\ s)^2 +
\frac{1}{m} \ast ((\text{nrm-dist}\ q\ s)^2 - (\text{nrm-dist}\ q\ p)^2 - (\text{nrm-dist}\ p\ s)^2)

by simp

finally have (\text{nrm-dist}\ r\ s)^2 = (\text{nrm-dist}\ r\ p)^2 + (\text{nrm-dist}\ p\ s)^2 +
\frac{1}{m} \ast ((\text{nrm-dist}\ q\ s)^2 - (\text{nrm-dist}\ q\ p)^2 - (\text{nrm-dist}\ p\ s)^2) .

moreover

\{ from \text{nrm-dist-dot [of r p]} \text{ and } \langle r - p = (1/m) *_{R} (q - p) \rangle

have (\text{nrm-dist}\ r\ p)^2 = ((1/m) *_{R} (q - p)) \cdot ((1/m) *_{R} (q - p))

by simp

also from \text{inner-scaleR-left [of 1/m q - p]} \text{ and}

\text{inner-scaleR-right [of -1/m q - p]}

have \ldots = 1/m^2 \ast ((q - p) \cdot (q - p))

by (simp add: power2-eq-square)

also from \text{nrm-dist-dot [of q p]} have \ldots = 1/m^2 \ast (\text{nrm-dist}\ q\ p)^2

by simp

finally have (\text{nrm-dist}\ r\ p)^2 = 1/m^2 \ast (\text{nrm-dist}\ q\ p)^2 . \}

ultimately have

(\text{nrm-dist}\ r\ s)^2 = 1/m^2 \ast (\text{nrm-dist}\ q\ p)^2 + (\text{nrm-dist}\ p\ s)^2 +
\frac{1}{m} \ast ((\text{nrm-dist}\ q\ s)^2 - (\text{nrm-dist}\ q\ p)^2 - (\text{nrm-dist}\ p\ s)^2)

by simp

\text{with norm-metric.symm [of q p]}

have (\text{nrm-dist}\ r\ s)^2 = 1/m^2 \ast (\text{nrm-dist}\ p\ q)^2 + (\text{nrm-dist}\ p\ s)^2 +
\frac{1}{m} \ast ((\text{nrm-dist}\ q\ s)^2 - (\text{nrm-dist}\ q\ p)^2 - (\text{nrm-dist}\ p\ s)^2)

by simp \}

\text{note fivelemma} = \text{this}

from fivelemma [of l b a c d] \text{ and } (d \neq 0) \text{ and } (b = a = l *_{R} (c - a))

have (\text{nrm-dist}\ c\ d)^2 = 1/l^2 \ast (\text{nrm-dist}\ a\ b)^2 + (\text{nrm-dist}\ a\ d)^2 +
1/l \ast ((\text{nrm-dist}\ b\ d)^2 - (\text{nrm-dist}\ a\ b)^2 - (\text{nrm-dist}\ a\ d)^2)

by simp

also from (l = l') \text{ and}

\langle a\ b \equiv_{R} a'\ b' \rangle \text{ and}
\langle a\ d \equiv_{R} a'\ d' \rangle \text{ and}
\langle b\ d \equiv_{R} b'\ d' \rangle

have \ldots = 1/l'^{2} \ast (\text{nrm-dist}\ a'\ b')^2 + (\text{nrm-dist}\ a'\ d')^2 +
1/l' \ast ((\text{nrm-dist}\ b'\ d')^2 - (\text{nrm-dist}\ a'\ b')^2 - (\text{nrm-dist}\ a'\ d')^2)

by simp

also from fivelemma [of l' b' a' c' d'] \text{ and}

\langle l' \neq 0 \rangle \text{ and}
\langle b' - a' = l' *_{R} (c' - a') \rangle

have \ldots = (\text{nrm-dist}\ c'\ d')^2 \text{ by simp}

finally have (\text{nrm-dist}\ c\ d)^2 = (\text{nrm-dist}\ c'\ d')^2 .

\text{hence sqrt ((\text{nrm-dist}\ c\ d)^2) = sqrt ((\text{nrm-dist}\ c'\ d')^2)} \text{ by simp}

\text{with real-sqrt-abs show c d \equiv_{R} c'\ d'} \text{ by simp}

qed }

thus \forall a\ b\ c\ d\ a'\ b'\ c'\ d'.

a \neq b \land B_{R} a\ b\ c \land B_{R} a'\ b'\ c'\ \land
\[
\begin{align*}
\text{a} \ b & \equiv_{R} \text{a}' \ b' \land \text{b} \ c \equiv_{R} \text{b}' \ c' \land \text{a} \ d \equiv_{R} \text{a}' \ d' \land \text{b} \ d \equiv_{R} \text{b}' \ d' \\
\text{c} \ d & \equiv_{R} \text{c}' \ d' \\
\text{by blast}
\end{align*}
\]

4.2 Real Euclidean space also satisfies axioms 6, 7, and 11

**Lemma** `rearrange-real-euclid-B`:

**Fixes** \(w \ y \ z :: \text{real}'('n)\) and \(h\)

**Shows** \(y - w = h \ast_{R} (z - w) \iff y = h \ast_{R} z + (1 - h) \ast_{R} w\)

**Proof**

**Assume** \(y - w = h \ast_{R} (z - w)\)

**Hence** \(y - w + w = h \ast_{R} (z - w) + w\) by simp

**Hence** \(y = h \ast_{R} (z - w) + w\) by simp

**With** `scaleR-right-diff-distrib` [of \(h \ z \ w\)]

**Have** \(y = h \ast_{R} z + w - h \ast_{R} w\) by simp

**With** `scaleR-left-diff-distrib` [of \(1 \ h \ w\)]

**Show** \(y = h \ast_{R} z + (1 - h) \ast_{R} w\) by simp

**Next**

**Assume** \(y = h \ast_{R} z + (1 - h) \ast_{R} w\)

**With** `scaleR-left-diff-distrib` [of \(1 \ h \ w\)]

**Have** \(y = h \ast_{R} z + w - h \ast_{R} w\) by simp

**With** `scaleR-right-diff-distrib` [of \(h \ z \ w\)]

**Have** \(y = h \ast_{R} (z - w) + w\) by simp

**Hence** \(y - w + w = h \ast_{R} (z - w) + w\) by simp

**Thus** \(y - w = h \ast_{R} (z - w)\) by simp

**Qed**

**Interpretation** real-euclid: tarski-absolute-space real-euclid-C real-euclid-B

**Proof**

**Fix** \(a \ b\)

**Assume** \(B_{R} \ a \ b \ a\)

**With** `real-euclid-B-def` [of \(a \ b \ a\)]

**Obtain** \(l\) where \(b - a = l \ast_{R} (a - a)\) by auto

**Hence** \(a = b\) by simp

**Thus** \(\forall a \ b. B_{R} \ a \ b \ a \to a = b\) by auto

**Fix** \(a \ b \ c \ p \ q\)

**Assume** \(B_{R} \ a \ p \ c\) and \(B_{R} \ b \ q \ c\)

**From** `real-euclid-B-def` [of \(a \ p \ c\)] and \(\langle B_{R} \ a \ p \ c\rangle\)

**Obtain** \(i\) where \(i \geq 0\) and \(i \leq 1\) and \(p - a = i \ast_{R} (c - a)\) by auto

**Have** \(\exists x. B_{R} \ p \ x \ b \wedge B_{R} \ q \ x \ a\)

**Proof** **Cases**

**Assume** \(i = 0\)

**With** \(p - a = i \ast_{R} (c - a)\): \(\text{have } p = a\) by simp

**Hence** \(p - a = 0 \ast_{R} (b - p)\) by simp

**Moreover** \(\text{have } (0::\text{real}) \geq 0\) and \(\langle 0::\text{real} \rangle \leq 1\) by auto

**Moreover** **Note** `real-euclid-B-def` [of \(p \ a \ b\)]

**Ultimately** \(\text{have } B_{R} \ p \ a \ b\) by auto

**Moreover**
\{ have \ a-q = 1 \ast_R (a-q) \text{ by simp} \\
\text{moreover have } (I::real) \geq 0 \text{ and } (I::real) \leq 1 \text{ by auto} \\
\text{moreover note } real-euclid-B-def [of q a a] \\
\text{ultimately have } B_R \ q a a \text{ by blast } \}
ultimately have \ B_R \ p \ a b \land B_R \ q a a \text{ by simp}
thus \exists x. B_R \ p x b \land B_R \ q x a \text{ by auto}

next
assume \ i \neq 0
from \ real-euclid-B-def \ [of b q c] \text{ and } (B_R \ b q c)
\text{obtain } j \text{ where } j \geq 0 \text{ and } j \leq 1 \text{ and } q-b = j \ast_R (c-b) \text{ by auto}
from \ (i \geq 0) \text{ and } (i \leq 1)
\text{have } 1-i \geq 0 \text{ and } 1-i \leq 1 \text{ by auto}
from \ (j \geq 0) \text{ and } (1-i \geq 0)
\text{have } j \ast (1-i) \geq 0 \text{ by auto}
with \ (i \geq 0) \text{ and } (i \neq 0) \text{ have } i+j \ast (1-i) > 0 \text{ by simp}
hence \ i+j \ast (1-i) \neq 0 \text{ by simp}
let \ ?l = j \ast (1-i) / (i+j \ast (1-i))
from \ diff-divide-distrib \ [of i+j \ast (1-i) j \ast (1-i) i+j \ast (1-i)] \text{ and }
\text{have } 1-?l = i / (i+j \ast (1-i)) \text{ by simp}
let \ ?k = i \ast (1-j) / (j+i \ast (1-j))
from \ right-diff-distrib \ [of i \ast j \ast i] \text{ and }
\text{and } mult.commute \ [of i \ast j \ast i]
\text{and } add.commute \ [of i \ast j]
\text{have } j+i \ast (1-j) = i+j \ast (1-i) \text{ by simp}
with \ (i+j \ast (1-i) \neq 0) \text{ have } j+i \ast (1-j) \neq 0 \text{ by simp}
with \ diff-divide-distrib \ [of j+i \ast (1-j) i \ast (1-j) j+i \ast (1-j)]
\text{have } 1-?k = j / (j+i \ast (1-j)) \text{ by simp}
with \ (1-?l = i / (i+j \ast (1-i))) \text{ and }
\text{and } times-divide-eq-left \ [of -i+j \ast (1-i)] \text{ and }
\text{and } mult.commute \ [of i \ast j]
\text{have } (1-?l) \ast j = (1-?k) \ast i \text{ by simp}
moreover
\{ from \ (1-?k = j / (j+i \ast (1-j))) \text{ and }
\text{and } j+i \ast (1-j) = i+j \ast (1-i)\}
\text{have } ?l = (1-?k) \ast (1-i) \text{ by simp } \}
moreover
\{ from \ (1-?l = i / (i+j \ast (1-i))) \text{ and }
\text{and } j+i \ast (1-j) = i+j \ast (1-i)\}
\text{have } (1-?l) \ast (1-j) = ?k \text{ by simp } \}
ultimately
\text{have } \ ?l \ast_R a + ((1-?l) \ast j) \ast_R c + ((1-?l) \ast (1-j)) \ast_R b = 
?k \ast_R b + ((1-?k) \ast i) \ast_R c + ((1-?k) \ast (1-i)) \ast_R a
\text{by simp}
\text{with } scaleR-scaleR
\text{have } \ ?l \ast_R a + (1-?l) \ast_R j \ast_R c + (1-?l) \ast_R (1-j) \ast_R b = 
?k \ast_R b + (1-?k) \ast_R i \ast_R c + (1-?k) \ast_R (1-i) \ast_R a

26
by simp
with scaleR-right-distrib \[ (1 - \mathcal{A}) \cdot j + R \cdot c + (1 - j) \cdot R \cdot b \] and
scaleR-right-distrib \[ (1 - ?k) \cdot i + R \cdot c + (1 - i) \cdot R \cdot a \] and
add_assoc \[ ?l \cdot R \cdot a + (1 - \mathcal{A}) \cdot R \cdot j + R \cdot c + (1 - j) \cdot R \cdot b \] and
add_assoc \[ ?k \cdot R \cdot b + (1 - ?k) \cdot R \cdot i + R \cdot c + (1 - i) \cdot R \cdot a \]
have \[ ?l \cdot R \cdot a + (1 - \mathcal{A}) \cdot R \cdot (j + R \cdot c + (1 - j) \cdot R \cdot b) = ?k \cdot R \cdot b + (1 - ?k) \cdot R \cdot (i + R \cdot c + (1 - i) \cdot R \cdot a) \]
by arith
from \( ?l \cdot R \cdot a + (1 - \mathcal{A}) \cdot R \cdot (j + R \cdot c + (1 - j) \cdot R \cdot b) = ?k \cdot R \cdot b + (1 - ?k) \cdot R \cdot (i + R \cdot c + (1 - i) \cdot R \cdot a) \) and
\( p - a = i \cdot R \cdot (c - a) \) and
\( q - b = j \cdot R \cdot (c - b) \) and
\( \text{rearrange-real-euclid-B} \ [of \ p \ a \ i \ c] \) and
\( \text{rearrange-real-euclid-B} \ [of \ q \ b \ j \ c] \)
have \[ ?l \cdot R \cdot a + (1 - \mathcal{A}) \cdot R \cdot ?k \cdot R \cdot b = ?k \cdot R \cdot b + (1 - ?k) \cdot R \cdot p \) by simp
let \( \mathcal{E} = ?l \cdot R \cdot a + (1 - \mathcal{A}) \cdot R \cdot q \)
from \( \text{rearrange-real-euclid-B} \ [of \ \mathcal{E} \ q \ ?l \ a] \)
have \( \mathcal{E} - q = \mathcal{A} \cdot R \cdot (a - q) \) by simp
from \( \text{rearrange-real-euclid-B} \ [of \ \mathcal{E} \ q \ ?l \ a] \)
have \( \mathcal{E} = ?k \cdot R \cdot b + (1 - ?k) \cdot R \cdot p \) and
\( \text{rearrange-real-euclid-B} \ [of \ \mathcal{E} \ p \ ?k \ b] \)
have \( \mathcal{E} - p = ?k \cdot R \cdot (b - p) \) by simp
from \( i + j \cdot R \cdot (1 - i) > 0 \) and
\( j \cdot R \cdot (1 - i) \geq 0 \) and
\( \text{zero-le-divide-iff} \ [of \ j \cdot R \cdot (1 - i) \ i + j \cdot R \cdot (1 - i)] \)
have \( ?l \geq 0 \) by simp
from \( i + j \cdot R \cdot (1 - i) > 0 \) and
\( i \geq 0 \) and
\( \text{zero-le-divide-iff} \ [of \ i \ i + j \cdot R \cdot (1 - i)] \) and
\( (1 - \mathcal{A}) = i \cdot R \cdot (i + j \cdot R \cdot (1 - i)) \)
have \( 1 - \mathcal{A} \geq 0 \) by simp
hence \( ?l \leq 1 \) by simp
with \( ?l \geq 0 \) and
\( \mathcal{E} - q = ?l \cdot R \cdot (a - q) \) and
\( \text{real-euclid-B-def} \ [of \ q \ \mathcal{E} \ a] \)
have \( B_{?k}, \ q \ ?x \ a \) by auto
from \( j \geq 0 \) have \( 1 - j \geq 0 \) by simp
with \( i - ?l \geq 0 \) and
\( (1 - ?l) \cdot R \cdot (1 - j) = ?k \) and
\( \text{zero-le-mult-iff} \ [of \ 1 - \mathcal{A} \ 1 - j] \)
have \( ?k \geq 0 \) by simp
from \( j \geq 0 \) have \( 1 - j \leq 1 \) by simp
from \( ?l \geq 0 \) have \( 1 - \mathcal{A} \leq 1 \) by simp
with \( 1 - j \leq 1 \) and
\( 1 - j \geq 0 \) and
\( \text{multi-mono} \ [of \ 1 - ?l \ 1 \ 1 - j] \) and
\( (1 - ?l) \cdot R \cdot (1 - j) = ?k \)
have \( ?k \leq 1 \) by simp
with \( ?k \geq 0 \) and
\( \mathcal{E} - p = ?k \cdot R \cdot (b - p) \) and
real-euclid-B-def \[ \text{of } p \ ?x b \]

have \( B_R \ p \ ?x b \) by auto

with \( B_R \ q \ ?x a \) show \( ?\text{thesis} \) by auto

qed \}

thus \( \forall \ a \ b \ c \ p \ q, \ B_R \ a \ p \ c \wedge B_R \ b \ q \ c \longrightarrow (\exists x. \ B_R \ p \ x \ b \wedge B_R \ q \ x \ a) \) by auto

{ fix \( X \ Y \)

assume \( \exists a. \forall x y, x \in X \wedge y \in Y \longrightarrow B_R \ a \ x \ y \)

then obtain \( a \) where \( \forall x y, x \in X \wedge y \in Y \longrightarrow B_R \ a \ x \ y \) by auto

have \( \exists b, \forall x y, x \in X \wedge y \in Y \longrightarrow B_R \ x \ b \ y \)

proof cases

assume \( X \subseteq \{ a \} \wedge Y = \{ \} \)

let \( ?b = a \)

{ fix \( x \ y \)

assume \( x \in X \) and \( y \in Y \)

with \( \langle x \subseteq \{ a \} \wedge Y = \{ \} \rangle \) have \( x = a \) by auto

from \( \forall x y, x \in X \wedge y \in Y \longrightarrow B_R \ a \ x \ y \) and \( \langle x \in X \rangle \) and \( \langle y \in Y \rangle \)

have \( B_R \ a \ x \ y \) by simp

with \( \langle x = a \rangle \) have \( B_R \ x \ ?b \ y \) by simp \}

hence \( \forall x y, x \in X \wedge y \in Y \longrightarrow B_R \ x \ ?b \ y \) by simp

thus \( ?\text{thesis} \) by auto

next

assume \( \neg (X \subseteq \{ a \} \wedge Y = \{ \}) \)

hence \( X - \{ a \} \neq \{ \} \) and \( Y \neq \{ \} \) by auto

from \( \langle X - \{ a \} \neq \{ \} \rangle \) obtain \( c \) where \( c \in X \) and \( c \neq a \) by auto

from \( \langle c \neq a \rangle \) have \( c - a \neq 0 \) by simp

{ fix \( y \)

assume \( y \in Y \)

with \( \forall x y, x \in X \wedge y \in Y \longrightarrow B_R \ a \ x \ y \) and \( \langle c \in X \rangle \)

have \( B_R \ a \ c \ y \) by simp

with \( \text{real-euclid-B-def \[\text{of } a \ c \ y\]} \)

obtain \( l \) where \( l \geq 0 \) and \( l \leq 1 \) and \( c - a = l *_R (y - a) \) by auto

from \( \langle c - a = l *_R (y - a) \rangle \) and \( \langle c - a \neq 0 \rangle \) have \( l \neq 0 \) by simp

with \( \langle l \geq 0 \rangle \) have \( l > 0 \) by simp

with \( \langle c - a = l *_R (y - a) \rangle \) have \( y - a = (1/l) *_R (c - a) \) by simp

from \( \langle l > 0 \rangle \) and \( \langle l \leq 1 \rangle \) have \( 1/l \geq 1 \) by simp

with \( \langle y - a = (1/l) *_R (c - a) \rangle \)

have \( \exists j \geq 1. \ y - a = j *_R (c - a) \) by auto \}

note \( \text{ylemma = this} \)

from \( \langle Y \neq \{ \} \rangle \) obtain \( d \) where \( d \in Y \) by auto

with \( \text{ylemma \[\text{of } d\]} \)

obtain \( jd \) where \( jd \geq 1 \) and \( d - a = jd *_R (c - a) \) by auto

{ fix \( x \)

assume \( x \in X \)

with \( \forall x y, x \in X \wedge y \in Y \longrightarrow B_R \ a \ x \ y \) and \( \langle d \in Y \rangle \)

have \( B_R \ a \ x \ d \) by simp

with \( \text{real-euclid-B-def \[\text{of } a \ x \ d\]} \)

obtain \( l \) where \( l \geq 0 \) and \( x - a = l *_R (d - a) \) by auto

from \( \langle x - a = l *_R (d - a) \rangle \) and \( \langle d - a = jd *_R (c - a) \rangle \) and

28
scaleR-scaleR
have $x - a = (l \times jd) \times_R (c - a)$ by simp
hence $\exists i. x - a = i \times_R (c - a)$ by auto }

note zlemma = this
let $?S = \{ j, j \geq 1 \land (\exists y \in Y. y - a = j \times_R (c - a))\}$
from (d $\in Y$) and (j $\geq 1$) and (d $- a = jd \times_R (c - a))$
  have $?S \neq \{ \}$ by auto
let $?k = \lnf ?S$
let $?b = ?k \times_R c + (1 - ?k) \times_R a$
from rearrange-real-euclid-B [of $?b a ?k c]
  have $?b - a = ?k \times_R (c - a)$ by simp

{ fix $x y$
  assume $x \in X$ and $y \in Y$
  from zlemma [of $x$] and ($x \in X$)
    obtain i where $x - a = i \times_R (c - a)$ by auto
  from ylemma [of $y$] and ($y \in Y$)
    obtain j where $j \geq 1$ and $y - a = j \times_R (c - a)$ by auto
  with ($y \in Y$) have $j \in ?S$ by auto
  then have $?k \leq j$ by (auto intro: cInf-lower)

{ fix $h$
  assume $h \in ?S$
  hence $h \geq 1$ by simp
  from ($h \in ?S$)
    obtain $z$ where $z \in Y$ and $z - a = h \times_R (c - a)$ by auto
  from (\A x y. x $\in X$ \land y $\in Y$ $\rightarrow$ B$_R$ a x y) and ($x \in X$) and ($z \in Y$)
    have $B_R$ a x z by simp
  with real-euclid-B-def [of a x z]
    obtain $l$ where $l \leq 1$ and $x - a = l \times_R (z - a)$ by auto
  with ($z - a = h \times_R (c - a)$) and scaleR-scaleR
    have $x - a = (l \times h) \times_R (c - a)$ by simp
  with ($x - a = i \times_R (c - a)$)
    have $i \times_R (c - a) = (l \times h) \times_R (c - a)$ by auto
  with scaleR-cancel-right and ($c - a \neq 0$) have $i = l \times h$ by blast
  with ($l \leq 1$) and ($h \geq 1$) have $i \leq h$ by simp }
with ($\exists ?S = \{ \}$ and cInf-greatest [of $?S$] have $i \leq ?k$ by simp
have $y - x = (y - a) - (x - a)$ by simp
with ($y - a = j \times_R (c - a)$) and ($x - a = i \times_R (c - a)$)
  have $y - x = j \times_R (c - a) - i \times_R (c - a)$ by simp
with scaleR-left-diff-distrib [of $j i c - a$]
  have $y - x = (j - i) \times_R (c - a)$ by simp
have $?b - x = (?b - a) - (x - a)$ by simp
with ($?b - a = ?k \times_R (c - a)$) and ($x - a = i \times_R (c - a)$)
  have $?b - x = ?k \times_R (c - a) - i \times_R (c - a)$ by simp
with scaleR-left-diff-distrib [of $?k i c - a$]
  have $?b - x = (?k - i) \times_R (c - a)$ by simp
have $B_R x ?b y$
proof cases
  assume $i = j$
  with ($i \leq ?k$) and ($?k \leq j$) have $?k = i$ by simp
with \( ?b - x = (\forall k - i) \cdot R \cdot (c - a) \) have \( ?b - x = 0 \) by simp

hence \( ?b - x = 0 \cdot R \cdot (y - x) \) by simp

with real-euclid-B-def [of \( ?b \cdot y \)] show \( B_{\mathbb{R}} \cdot x \) by auto

next

assume \( i \neq j \)

with \( (i \leq \forall k) \) and \( (\forall k \leq j) \) have \( j - i > 0 \) by simp

with \( y - x = (j - i) \cdot R \cdot (c - a) \) and scaleR-scaleR

have \( c - a = (1 \cdot (j - i)) \cdot R \cdot (y - x) \) by simp

with \( ?b - x = (\forall k - i) \cdot R \cdot (c - a) \) and scaleR-scaleR

have \( ?b - x = ((\forall k - i) \cdot (j - i)) \cdot R \cdot (y - x) \) by simp

let \( ?l = (\forall k - i) / (j - i) \)

from \( (\forall k \leq j) \) have \( ?k - i \leq j - i \) by simp

with \( j - i > 0 \) have \( ?l \leq 1 \) by simp

from \( (i \leq \forall k) \) and \( (j - i > 0) \) and pos-le-divide-eq [of \( j - i \) \( 0 \) \( ?k - i \)]

have \( ?l \geq 0 \) by simp

with real-euclid-B-def [of \( x \) \( ?b \cdot y \)] and

\( \langle ?l \leq 1 \rangle \) and

\( \langle ?b - x = ?l \cdot R \cdot (y - x) \rangle \)

show \( B_{\mathbb{R}} \cdot x \) by auto

qed}

thus \( \exists b . \forall x . x \in X \land y \in Y \rightarrow B_{\mathbb{R}} \cdot x \cdot b \cdot y \) by auto

qed}

thus \( \forall X . \forall Y . (\exists b . \forall x . x \in X \land y \in Y \rightarrow B_{\mathbb{R}} \cdot a \cdot x \cdot y) \rightarrow

(\exists b . \forall x . y . x \in X \land y \in Y \rightarrow B_{\mathbb{R}} \cdot x \cdot b \cdot y) \)

by auto

qed

4.3 Real Euclidean space satisfies the Euclidean axiom

lemma rearrange-real-euclid-B-2:

fixes \( a \cdot b \cdot c :: \text{real}''(n::finite)

assumes \( l \neq 0 \)

shows \( b - a = l \cdot R \cdot (c - a) \leftrightarrow c = (1 / l) \cdot R \cdot b + (1 - 1 / l) \cdot R \cdot a \)

proof

from scaleR-right-diff-distrib [of \( 1 / l \cdot b \cdot a \)]

have \( (1 / l) \cdot R \cdot (b - a) = c - a \leftrightarrow (1 / l) \cdot R \cdot b - (1 / l) \cdot R \cdot a = c \) by auto

also with scaleR-left-diff-distrib [of \( 1 / l \cdot b \cdot a \)]

have \( c = (1 / l) \cdot R \cdot b + (1 - 1 / l) \cdot R \cdot a \) by auto

finally have eq:

\( (1 / l) \cdot R \cdot (b - a) = c - a \leftrightarrow c = (1 / l) \cdot R \cdot b + (1 - 1 / l) \cdot R \cdot a \)

\{ assume \( l \neq 0 \), have \( (1 / l) \cdot R \cdot (b - a) = c - a \) by simp

with eq show \( c = (1 / l) \cdot R \cdot b + (1 - 1 / l) \cdot R \cdot a \). \}

\{ assume \( c = (1 / l) \cdot R \cdot b + (1 - 1 / l) \cdot R \cdot a \)

with eq have \( (1 / l) \cdot R \cdot (b - a) = c - a \).

hence \( l \cdot R \cdot (b - a) = l \cdot R \cdot (c - a) \) by simp

with \( l \neq 0 \), show \( b - a = l \cdot R \cdot (c - a) \) by simp \}

qed
interpretation real-euclid: tarski-space real-euclid-C real-euclid-B

proof

\{ fix a b c d t \}
assume B_R a d t and B_R b d c and a \neq d
from real-euclid-B-def [of a d t] and \{B_R a d t\}
   obtain j where j \geq 0 and j \leq 1 and d - a = j \ast_R (t - a) by auto
from \{d - a = j \ast_R (t - a)\ and \(a \neq d\) have j \neq 0 by auto
with \{d - a = j \ast_R (t - a)\ and rearrange-real-euclid-B-2
   have t = (1/j) \ast_R d + (1 - 1/j) \ast_R a by auto
let ?x = (1/j) \ast_R b + (1 - 1/j) \ast_R a
let ?y = (1/j) \ast_R c + (1 - 1/j) \ast_R a
from \{j \neq 0\} and rearrange-real-euclid-B-2 have
   b - a = j \ast_R (?x - a) and c - a = j \ast_R (?y - a) by auto
with real-euclid-B-def and \(j \geq 0\) and \(j \leq 1\) have
   B_R a b ?x and B_R a c ?y by auto
from real-euclid-B-def and \{B_R b d c\} obtain k where
   k \geq 0 and k \leq 1 and d - b = k \ast_R (c - b) by blast
from \{t = (1/j) \ast_R d + (1 - 1/j) \ast_R a\} have
   t - ?x = (1/j) \ast_R d - (1/j) \ast_R b by simp
also from scaleR-right-diff-distrib [of 1/j d b] have
   \ldots = (1/j) \ast_R (d - b) by simp
also from \{d - b = k \ast_R (c - b)\} have
   \ldots = k \ast_R (1/j) \ast_R (c - b) by simp
also from scaleR-right-diff-distrib [of 1/j c b] have
   \ldots = k \ast_R (?y - ?x) by simp
finally have t - ?x = k \ast_R (?y - ?x) .
with real-euclid-B-def and \(k \geq 0\) and \(k \leq 1\) have B_R ?x t ?y by blast
with \{B_R a b ?x\} and \{B_R a c ?y\} have
   \exists x y. B_R a b x \land B_R a c y \land B_R x t y by auto \}
thus \forall a b c d t. B_R a d t \land B_R b d c \land a \neq d \rightarrow
   (\exists x y. B_R a b x \land B_R a c y \land B_R x t y)
by auto
qed

4.4 The real Euclidean plane

lemma Col-dep2:
real-euclid.Col a b c \iff dep2 (b - a) (c - a)

proof –
from real-euclid.Col-def have
   real-euclid.Col a b c \iff B_R a b c \land B_R b c a \land B_R c a b by auto
moreover from dep2-def have
   dep2 (b - a) (c - a) \iff (\exists w r s. b - a = r \ast_R w \land c - a = s \ast_R w)
by auto
moreover
\{ assume B_R a b c \land B_R b c a \land B_R c a b
moreover
\{ assume B_R a b c
   with real-euclid-B-def obtain l where b - a = l \ast_R (c - a) by blast

31
moreover have \( c - a = 1 \ast_R (c - a) \) by simp
ultimately have \( \exists \, w \, r \, s \, . \, b - a = r \ast_R w \land c - a = s \ast_R w \) by blast
case proof
assumption
moreover have \( B \_R \land b \ast_R c \ast_R a \)
with real-euclid-B-def obtain \( l \) where \( c - b = l \ast_R (a - b) \) by blast
moreover have \( c - a = (c - b) - (a - b) \) by simp
ultimately have \( c - a = l \ast_R (a - b) \) by simp
with scaleR-left-diff-distrib \( \langle l \ast_R 1 \ast_R a - b \rangle \) have
\( c - a = (l - 1) \ast_R (a - b) \) by simp
moreover have \( \exists \, w \, r \, s \, . \, b - a = r \ast_R w \land c - a = s \ast_R w \) by blast
case proof
assumption
moreover have \( B \_R \land c \ast_R a \ast_R b \)
with real-euclid-B-def obtain \( l \) where \( a - c = l \ast_R (b - c) \) by blast
moreover have \( c - a = -(a - c) \) by simp
ultimately have \( c - a = -(l \ast_R (b - c)) \) by simp
with scaleR-minus-left have \( c - a = (-l) \ast_R (b - c) \) by simp
moreover have \( b - a = (b - c) + (c - a) \) by simp
ultimately have \( b - a = 1 \ast_R (b - c) + (-l) \ast_R (b - c) \) by simp
with scaleR-left-distrib \( \langle \langle -l \ast_R (b - c) \rangle \rangle \) have
\( b - a = (1 + (-l)) \ast_R (b - c) \) by simp
with \( c - a = (-l) \ast_R (b - c) \) have
\( \exists \, w \, r \, s \, . \, b - a = r \ast_R w \land c - a = s \ast_R w \) by blast
ultimately have \( \exists \, w \, r \, s \, . \, b - a = r \ast_R w \land c - a = s \ast_R w \) by auto
next
assume \( s \neq 0 \)
with \( (c - a = s \ast_R w) \) have \( a = c \) by simp
with real-euclid.th3-1 have \( B \_R \land b \ast_R c \ast_R a \)
thus \(?thesis by simp

32
from $(r/s > 1)$ and inverse-positive-iff-positive [of $r/s$] have 
\[ s/r \geq 0 \] by simp
with real-euclid-B-def
\[ \langle c - a = (s/r) * (b - a) \rangle \]
and \[ (s/r) \leq 1 \]
have $B_R a b c$ by auto
with real-euclid-th3-2 have $B_R b a c$ by auto
hence $?thesis$ by simp 

moreover
\{ assume $r/s < 0$
have $b - c = (b - a) + (a - c)$ by simp
with $\langle b - a = (r/s) * (c - a) \rangle$ have
\[ b - c = (r/s) * (c - a) + (a - c) \] by simp
have $c - a = -(a - c)$ by simp
with scaleR-minus-right [of $r/s a - c$] have
\[ (r/s) * (c - a) = -(r/s * (a - c)) \] by arith
with $\langle b - c = (r/s) * (c - a) + (a - c) \rangle$ have
\[ b - c = -(r/s) * (a - c) + (a - c) \] by simp
with scaleR-left-distrib [of $-(r/s)$ $1 a - c$] have
\[ b - c = -(r/s) * (a - c) \] by simp
moreover from $(r/s < 0)$ have $-(r/s) + 1 > 1$ by simp
ultimately have $a - c = (1 / (-(r/s) + 1)) * (b - c)$ by auto
let $?l = 1 / (-(r/s) + 1)$
from $-(r/s) + 1 > 1$ and le-imp-inverse-le [of $1 - (r/s) + 1$] have
\[ \langle ?l \leq 1 \rangle \] by simp
from $-(r/s) + 1 > 1$
\[ \langle a - c = ?l * (b - c) \rangle \] have
\[ B_R c a b \] by blast
hence $?thesis$ by simp 
ultimately show $?thesis$ by auto
qed 

ultimately show $?thesis$ by blast

qed

lemma non-Col-example:
\[ -(\text{real-euclid.Col} 0 \ (\text{vector} [1/2,0]) :: \text{real}^2) \ (\text{vector} [0,1/2]) \]
(is $\neg (\text{real-euclid.Col} ?a ?b ?c)$)

proof
\{ assume $\langle ?b - ?a \rangle \ (\langle ?c - ?a \rangle$ with $\text{dep2-def} [\text{of} \ ?b - ?a \ ?c - ?a]$ obtain $w s$ where
\[ ?b - ?a = r \ * s \] and $\langle ?c - ?a = s \ * R \ w \rangle$ by auto
have $\langle ?b | ?s1 = 1/2 \rangle$ by simp
with $\langle ?b - ?a = r \ * R \ w \rangle$ have $r * (w | ?s1) = 1/2$ by simp
hence $w | ?s1 \neq 0$ by auto
have $\langle ?c | ?s1 = 0 \rangle$ by simp
with $\langle ?c - ?a = s \ * R \ w \rangle$ have $s * (w | ?s1) = 0$ by simp

33
with \( w \neq 1 \): have \( s = 0 \) by simp
have \( \cdot s = 1/2 \) by simp
with \( c - a = s \cdot 1 \) have \( s \cdot (w \cdot 1/2) = 1/2 \) by simp
with \( \cdot s = 0 \) have \( \text{False} \) by simp

\[ \text{hence } \neg(\text{dep}2 \ (b - a) \ (c - a)) \text{ by auto} \]
with \( \text{Col-dep}2 \) show \( \neg(\text{real-euclid.Col} \ a \ b \ c) \) by blast

qed

interpretation real-euclid:
\( \text{tarski real-euclid-C} \cdot \text{real-euclid-B} \)
proof
\{ let \( b = 0 \cdot \text{real}^2 \)
let \( c = \text{vector} \cdot [1/2, 0] \cdot \text{real}^2 \)
from \( \text{non-Col-example and real-euclid.Col-def} \)
\( \text{have } \neg B_R \ a \ b \ c \land \neg B_R \ b \ c \ a \land \neg B_R \ c \ a \ b \) by auto \}
thus \( \exists a \ b \ c \cdot \text{real}^2 \cdot \neg B_R \ a \ b \ c \land \neg B_R \ b \ a \ c \land \neg B_R \ c \ a \ b \)
by auto
\{ fix \( p \ a \ b \ c \cdot \text{real}^2 \)
assume \( p \neq q \) and \( a \ p \equiv_R \ a \ q \) and \( b \ p \equiv_R \ b \ q \) and \( c \ p \equiv_R \ c \ q \)
let \( m = (1/2) \cdot s_R \ (p + q) \)
from \( \text{scaleR-right-distrib} \) \( \cdot [1/2 \ p \ q] \) and
\( \text{scaleR-right-diff-distrib} \) \( \cdot [1/2 \ q \ p] \) and
\( \text{scaleR-left-diff-distrib} \) \( \cdot [1/2 \ 1 \ p] \)
\( \text{have } \cdot m - p = (1/2) \cdot s_R \ (q - p) \) by simp
with \( \cdot p \neq q \) have \( \cdot m - p \neq 0 \) by simp
from \( \text{scaleR-right-distrib} \) \( \cdot [1/2 \ p \ q] \) and
\( \text{scaleR-right-diff-distrib} \) \( \cdot [1/2 \ q \ p] \) and
\( \text{scaleR-left-diff-distrib} \) \( \cdot [1/2 \ 1 \ q] \)
\( \text{have } \cdot m - p = (1/2) \cdot s_R \ (q - p) \) by simp
with \( \cdot m - p = (1/2) \cdot s_R \ (q - p) \)
and \( \text{scaleR-minus-right} \) \( \cdot [1/2 \ q - p] \)
\( \text{have } \cdot m - q = - (\cdot m - p) \) by simp
with \( \text{norm-minus-cancel} \) \( \cdot [1/2 \ m - p] \)
\( \text{have } (\text{norm} \cdot (m - q))^2 = (\text{norm} \cdot (m - p))^2 \) by (simp only: norm-minus-cancel)
\}
fix \( d \)
assume \( d \ p \equiv_R \ d \ q \)
\( \text{hence } (\text{norm} \cdot (d - p))^2 = (\text{norm} \cdot (d - q))^2 \) by simp
have \( (d - \cdot m) \cdot (\cdot m - p) = 0 \)
proof —
\( \text{have } (\cdot m - q) = d - q \) by simp
\( \text{have } (\cdot m - p) = d - p \) by simp
with \( \text{dot-norm} \) \( \cdot [d - \cdot m \ m - p] \)
\( \text{have } (d - \cdot m) \cdot (\cdot m - p) =
\ (\text{norm} \cdot (d - p))^2 - (\text{norm} \cdot (d - \cdot m))^2 - (\text{norm} \cdot (\cdot m - p))^2 \) / 2
by simp
also from \( (\text{norm} \cdot (d - p))^2 = (\text{norm} \cdot (d - q))^2 \)
and \( (\text{norm} \cdot (\cdot m - q))^2 = (\text{norm} \cdot (\cdot m - p))^2 \)
have
\[ \ldots = (\text{norm } (d - q))^2 - (\text{norm } (d - m))^2 - (\text{norm } (m - q))^2) / 2 \]

by simp

also from \text{dot-norm } \{\text{of } d - m \ m - q\}

and \( d + (-q) = d - q \)

have

\[ \ldots = (d - m) \cdot (m - q) \] by simp

also from \text{inner-minus-right } \{\text{of } d - m \ m - p\}

and \( (m - q) = -(m - p) \)

have

\[ \ldots = -((d - m) \cdot (m - p)) \] by \text{(simp only: inner-minus-left)}

finally have \( (d - m) \cdot (m - p) = -(d - m) \cdot (m - p) \).

thus \( (d - m) \cdot (m - p) = 0 \) by arith

qed

\text{note } \text{m-lemma = this}

with \( a \ p \equiv \mathbb{R} \ a \ q \) have \( (a - m) \cdot (m - p) = 0 \) by simp

\{ fix \( d \)

\begin{align*}
& \text{assume } d \ p \equiv \mathbb{R} \ d \ q \\
& \text{with } \text{m-lemma } \text{have } (d - m) \cdot (m - p) = 0 \text{ by simp} \\
& \text{with } \text{dot-left-diff-distrib } \{\text{of } d - m \ a - m \ m - p\}

& \quad \text{and } ((a - m) \cdot (m - p) = 0) \\
& \quad \text{have } (d - a) \cdot (m - p) = 0 \text{ by } \text{(simp add: inner-diff-left inner-diff-right)} \}

\begin{align*}
& \text{with } (b \ p \equiv \mathbb{R} \ b \ q \ \text{and } c \ p \equiv \mathbb{R} \ c \ q) \text{ have} \\
& \quad (b - a) \cdot (m - p) = 0 \ \text{and } (c - a) \cdot (m - p) = 0 \text{ by simp+} \\
& \text{with } \text{real2-orthogonal-dep2 } \text{and } \text{?m - p} \neq 0 \text{ have dep2 } (b - a) (c - a)

& \quad \text{by blast} \\
& \text{with } \text{Col-dep2 } \text{have } \text{real-euclid} \text{.Col } a \ b \ c \text{ by auto} \\
& \text{with } \text{real-euclid} \text{.Col-def have } B_{\mathbb{R}} \ a \ b \ c \lor B_{\mathbb{R}} \ b \ a \ c \lor B_{\mathbb{R}} \ c \ a \ b \text{ by auto} \}

\text{thus } \forall \ p \ q \ a \ b \ c \ :: \ \text{real2}.

\begin{align*}
& p \neq q \land a \ p \equiv \mathbb{R} \ a \ q \land b \ p \equiv \mathbb{R} \ b \ q \land c \ p \equiv \mathbb{R} \ c \ q \rightarrow \\
& B_{\mathbb{R}} \ a \ b \ c \lor B_{\mathbb{R}} \ b \ c \ a \lor B_{\mathbb{R}} \ c \ a \ b

& \text{by blast}
\end{align*}

qed

4.5 Special cases of theorems of Tarski’s geometry

\text{lemma } \text{real-euclid-B-disjunction}:

\begin{align*}
& \text{assumes } l \geq 0 \ \text{and } b - a = l * (c - a) \\
& \text{shows } B_{\mathbb{R}} \ a \ b \ c \lor B_{\mathbb{R}} \ a \ c \ b
\end{align*}

proof \text{cases}

\begin{align*}
& \text{assume } l \leq 1 \\
& \text{with } \{ l \geq 0 \} \ \text{and } (b - a = l * (c - a)) \\
& \text{have } B_{\mathbb{R}} \ a \ b \ c \text{ by } \text{(unfold real-euclid-B-def) simp add: ezI } \{\text{of } - l\}

& \quad \text{thus } B_{\mathbb{R}} \ a \ b \ c \lor B_{\mathbb{R}} \ a \ c \ b \ ..
\end{align*}

next

\begin{align*}
& \text{assume } \neg (l \leq 1) \\
& \text{hence } 1/l \leq 1 \text{ by simp}
\end{align*}

from \( l \geq 0 \) have \( 1/l \geq 0 \) by simp

\vspace{1cm}

35
The following are true in Tarski’s geometry, but to prove this would
require much more development of it, so only the Euclidean case is proven
here.

**Theorem real-euclid-th5-1:**
assumes \(a \neq b\) and \(B_R a b c\) and \(B_R a b d\)
shows \(B_R a c d \lor B_R a d c\)

**Proof** —
from \(\langle B_R a b c \rangle \) and \(\langle B_R a b d \rangle \)
obtain \(l\) and \(m\) where \(l \geq 0\) and \(b - a = l * R (c - a)\)
and \(m \geq 0\) and \(b - a = m * R (d - a)\)
by \((\text{unfold real-euclid-B-def})\) auto
from \(\langle b - a = m * R (d - a) \rangle \) and \(\langle a \neq b \rangle\) have \(m \neq 0\) by auto

from \(l \geq 0\) and \((m \geq 0)\) have \(l/m \geq 0\) by \((\text{simp add: zero-le-divide-iff})\)

from \(\langle b - a = l * R (c - a) \rangle \) and \(\langle b - a = m * R (d - a) \rangle \)
have \(m * R (d - a) = l * R (c - a)\) by simp
hence \((1/m) * R (m * R (d - a)) = (1/l) * R (l * R (c - a))\) by simp
with \((m \neq 0)\) have \(d - a = (l/m) * R (c - a)\) by simp
with \((l/m \geq 0)\) and \((\text{real-euclid-B-disjunction})\)
show \(B_R a c d \lor B_R a d c\) by auto

**Qed**

**Theorem real-euclid-th5-3:**
assumes \(B_R a b d\) and \(B_R a c d\)
shows \(B_R a b c \lor B_R a c b\)

**Proof** —
from \(\langle B_R a b d \rangle \) and \(\langle B_R a c d \rangle \)
obtain \(l\) and \(m\) where \(l \geq 0\) and \(b - a = l * R (d - a)\)
and \(m \geq 0\) and \(c - a = m * R (d - a)\)
by \((\text{unfold real-euclid-B-def})\) auto

show \(B_R a b c \lor B_R a c b\)

**Proof** \textbf{Cases}
assume \(l = 0\)
with \(\langle b - a = l * R (d - a) \rangle\) have \(b - a = l * R (c - a)\) by simp
with \((l = 0)\)
have \(B_R a b c\) by \((\text{unfold real-euclid-B-def})\) \textbf{(simp add: exI [of - l])}
thus \(B_R a b c \lor B_R a c b\) ..

next
assume \(l \neq 0\)

36
from \( l \geq 0 \) and \( m \geq 0 \) have \( m/l \geq 0 \) by \( \text{simp add: zero-le-divide-iff} \)

from \( b - a = l *_{R} (d - a) \)
have \( (1/l) *_{R} (b - a) = (1/l) *_{R} (d - a) \) by \( \text{simp} \)
with \( d \neq 0 \) have \( d - a = (1/l) *_{R} (b - a) \) by \( \text{simp} \)
with \( c - a = m *_{R} (d - a) \) have \( c - a = (m/l) *_{R} (b - a) \) by \( \text{simp} \)
with \( m/l \geq 0 \) and \( \text{real-euclid-B-disjunction} \)
show \( B_{R} a b c \lor B_{R} a c b \) by \( \text{auto} \)
qed

end

5 Linear algebra

theory Linear-Algebra2
imports Miscellany
begin

lemma exhaust-4:
fixes \( x :: 4 \)
shows \( x = 1 \lor x = 2 \lor x = 3 \lor x = 4 \)
proof (induct \( x \))
  case (of-int \( z \))
  hence \( 0 \leq z \) and \( z < 4 \) by \( \text{simp-all} \)
  hence \( z = 0 \lor z = 1 \lor z = 2 \lor z = 3 \) by \( \text{arith} \)
  thus \( ?\text{case} \) by \( \text{auto} \)
qed

lemma forall-4: \((\forall i::4. P i) \iff P 1 \land P 2 \land P 3 \land P 4\)
by (metis exhaust-4)

lemma UNIV-4: \((UNIV::(4 set)) = \{1, 2, 3, 4\}\)
using exhaust-4
by \( \text{auto} \)

lemma vector-4:
fixes \( w :: 'a::zero \)
shows \( \text{(vector [w, x, y, z] :: 'a^4)$1 = w} \)
and \( \text{(vector [w, x, y, z] :: 'a^4)$2 = x} \)
and \( \text{(vector [w, x, y, z] :: 'a^4)$3 = y} \)
and \( \text{(vector [w, x, y, z] :: 'a^4)$4 = z} \)
unfolding \( \text{vector-def} \)
by \( \text{simp-all} \)

definition is-basis :: \((\text{real}^\sim\text{n}) \text{ set} \Rightarrow \text{bool} \) where
is-basis \( S \triangleq \text{independent} S \land \text{span} S = \text{UNIV} \)

37
lemma card-finite:
  assumes card S = CARD('n::finite)
  shows finite S
proof –
  from ⟨card S = CARD('n)⟩ have card S ≠ 0 by simp
  with card-eq-0-iff [of S] show finite S by simp
qed

lemma independent-is-basis:
  fixes B :: (real `'n::finite) set
  shows independent B ∧ card B = CARD('n) ←→ is-basis B
proof
  assume L: independent B ∧ card B = CARD('n)
  then have card (Basis :: (real `'n::finite) set) = card B
    by simp
  with L show is-basis B
    by (metis (no_types) card-eq-dim dim-UNIV independent-bound is-basis-def subset-antisym top-greatest)
next
  assume is-basis B
  then show independent B ∧ card B = CARD('n)
    by (metis DIM-cart DIM-real basis-card-eq-dim dim-UNIV is-basis-def mult.right-neutral top.extremum)
qed

lemma basis-finite:
  fixes B :: (real `'n::finite) set
  assumes is-basis B
  shows finite B
proof –
  from independent-is-basis [of B] and ⟨is-basis B⟩ have card B = CARD('n)
    by simp
  with card-finite [of B, where 'n = 'n] show finite B by simp
qed

lemma basis-expand:
  assumes is-basis B
  shows ∃ c. v = (∑ w∈B. (c w) *R w)
proof –
  from ⟨is-basis B⟩ have v ∈ span B unfolding is-basis-def by simp
  from basis-finite [of B] and ⟨is-basis B⟩ have finite B by simp
  with span-finite [of B] and ⟨v ∈ span B⟩
  show ∃ c. v = (∑ w∈B. (c w) *R w) by (simp add: scalar-equiv) auto
qed

lemma not-span-independent-insert:
  fixes v :: ('a::real_vector) `'n
  assumes independent S and v ∉ span S
shows independent (insert v S)
by (simp add: assms independent-insert)

lemma orthogonal-sum:
  fixes v :: real ^'n
  assumes \land w. w \in S \imp orthogonal v w
  shows orthogonal v (\sum w \in S. c w \ast s w)
  by (metis (no-types, lifting) assms orthogonal-clauses(1,2) orthogonal-rvsum scalar-equiv sum.infinite)

lemma orthogonal-self-eq-0:
  fixes v :: 'a :: real-inner ^'n
  assumes orthogonal v v
  shows v = 0
  using inner-eq-zero-iff [of v] and assms unfolding orthogonal-def
  by simp

lemma orthogonal-in-span-eq-0:
  fixes v :: real ^'n
  assumes v \in span S and \land w. w \in S \imp orthogonal v w
  shows v = 0
  using assms orthogonal-self orthogonal-to-span by blast

lemma orthogonal-independent:
  fixes v :: real ^'n
  assumes independent S and v \neq 0 and \land w. w \in S \imp orthogonal v w
  shows independent (insert v S)
  using assms not-span-independent-insert orthogonal-in-span-eq-0 by blast

lemma dot-scaleR-mult:
  shows (k \ast_R a) \cdot b = k \ast (a \cdot b) and a \cdot (k \ast_R b) = k \ast (a \cdot b)
  by auto

lemma dependent-explicit-finite:
  fixes S :: (('a::{real-vector, field}) ^'n) set
  assumes finite S
  shows dependent S \iff (\exists u. (\exists v \in S. u v \neq 0) \land (\sum v \in S. u v \ast_R v) = 0)
  by (simp add: assms dependent-finite)

lemma dependent-explicit-2:
  fixes v w :: ('a::{field, real-vector}) ^'n
  assumes v \neq w
  shows dependent {v, w} \iff (\exists i j. (i \neq 0 \lor j \neq 0) \land i \ast_R v + j \ast_R w = 0)
  proof
    let ?S = {v, w}
    have finite ?S by simp
    { assume dependent ?S

39
with dependent-explicit-finite [of ?S] and (finite ?S) and (v ≠ w)
show ∃ i j. (i ≠ 0 ∨ j ≠ 0) ∧ i *R v + j *R w = 0 by auto }

{ assume ∃ i j. (i ≠ 0 ∨ j ≠ 0) ∧ i *R v + j *R w = 0
then obtain i and j where i ≠ 0 ∨ j ≠ 0 and i *R v + j *R w = 0 by auto
let ?u = λ x. if x = v then i else j
from i ≠ 0 ∨ j ≠ 0 and (v ≠ w) have ∃ x∈?S. ?u x ≠ 0 by simp
from i *R v + j *R w = 0 and (v ≠ w)
have (∑ x∈?S. ?u x *R x) = 0 by simp
with dependent-explicit-finite [of ?S]
and (finite ?S) and (∃ x∈?S. ?u x ≠ 0)
show dependent ?S by best }
qed

5.1 Matrices

lemma zero-not-invertible:
¬ (invertible (0::realˆ′nˆ′m))
using invertible-times-eq-zero matrix-vector-mult-0 by blast

Based on matrix-vector-column in HOL/Multivariate_Analysis/Euclidean_Space.thy
in Isabelle 2009-1:

lemma vector-matrix-row:
fixes x :: ('a::comm-semiring-1) ^'m and A :: ('a ^'n ^'m)
shows x v * A = (∑ i∈UNIV. (x$ i) * (A$ i))
unfolding vector-matrix-mult-def
by (simp add: vec-eq-iff mult.commute)

lemma matrix-inv:
assumes invertible M
shows matrix-inv M ** M = mat 1
and M ** matrix-inv M = mat 1
using (invertible M) and someI-ex [of λ N. M ** N = mat 1 ∧ N ** M = mat 1]
unfolding invertible-def and matrix-inv-def
by simp-all

lemma matrix-inv-invertible:
assumes invertible M
shows invertible (matrix-inv M)
using (invertible M) and matrix-inv
unfolding invertible-def [of matrix-inv M]
by auto

lemma invertible-times-non-zero:
fixes M :: real ^'n ^'n
assumes invertible M and v ≠ 0
shows M *v v ≠ 0
using \langle invertible \ M \rangle \ \textbf{and} \ \langle \ v \neq 0 \ \rangle \ \textbf{and} \ \textit{invertible-times-eq-zero} \ [\text{of} \ M \ v] \\
\textbf{by} \ \textit{auto}

\textbf{lemma} \ \textit{matrix-right-invertible-ker}:
\textit{fixes} \ M :: \ \text{real}^{\langle n ::\ \text{finite} \rangle \times n} \\
\textit{shows} \ (\exists \ M'. \ M ** M' = \text{mat} \ \text{1}) \iff (\forall \ x. \ x v* M = 0 \rightarrow x = 0) \\
\textbf{using} \ \textit{left-invertible-transpose} \ \textit{matrix-left-invertible-ker} \ \textbf{by} \ \textit{force}

\textbf{lemma} \ \textit{left-invertible-iff-invertible}:
\textit{fixes} \ M :: \ \text{real}^{\langle n \times n \rangle} \\
\textit{shows} \ (\exists \ N. \ N ** M = \text{mat} \ \text{1}) \iff \text{invertible} \ M \\
\textbf{by} \ \textit{(simp add: invertible-def matrix-left-right-inverse)}

\textbf{lemma} \ \textit{right-invertible-iff-invertible}:
\textit{fixes} \ M :: \ \text{real}^{\langle n \times n \rangle} \\
\textit{shows} \ (\exists \ N. \ M ** N = \text{mat} \ \text{1}) \iff \text{invertible} \ M \\
\textbf{by} \ \textit{(simp add: invertible-def matrix-left-right-inverse)}

\textbf{definition} \ \textit{symmatrix} :: \ \langle a \times n \times n \rangle \Rightarrow \ \text{bool} \ \textbf{where} \\
\textit{symmatrix} M \ \text{≜} \ \text{transpose} M = M

\textbf{lemma} \ \textit{symmatrix-preserve}:
\textit{fixes} \ M N :: \ \langle a ::\ \text{comm-semiring-1} \rangle^{\langle n \times n \rangle} \\
\textit{assumes} \ \textit{symmatrix} M \\
\textit{shows} \ \textit{symmatrix} (N ** M ** \text{transpose} N) \\
\textbf{proof} - \\
\textbf{have} \ \text{transpose} (N ** M ** \text{transpose} N) = N ** (M ** \text{transpose} N) \\
\textbf{by} \ \textit{(metis (no-types) transpose-transpose \text{assms} matrix-transpose-mul symmatrix-def)} \\
\textbf{then show} \ ?thesis \\
\textbf{by} \ \textit{(simp add: matrix-mul-assoc symmatrix-def)}
\textbf{qed}

\textbf{lemma} \ \textit{non-zero-mult-invertible-non-zero}:
\textit{fixes} \ M :: \ \text{real}^{\langle n \times n \rangle} \\
\textit{assumes} \ v \neq 0 \ \textbf{and} \ \textit{invertible} \ M \\
\textit{shows} \ v v* M \neq 0 \\
\textbf{using} \ \langle v \neq 0 \ \rangle \ \textbf{and} \ \langle \text{invertible} \ M \rangle \ \textbf{and} \ \textit{times-invertible-eq-zero} \\
\textbf{by} \ \textit{auto}

\textbf{end}

\textbf{6} \ \textbf{Right group actions}

\textbf{theory} \ \textit{Action} \\
\textbf{imports} \ \textit{HOL-Algebra.Group} \\
\textbf{begin}

\textbf{locale} \ \textit{action} = \ \textit{group} + \\
\textit{fixes} \ \textit{act} :: \ \langle b \Rightarrow \ a \Rightarrow b \ (\text{infixl} < 69) \rangle

\textbf{41}
assumes \texttt{id-act [simp]}: \( b \circ \mathbf{1} = b \)
and \texttt{act-act\':}
\( g \in \text{carrier } G \land h \in \text{carrier } G \rightarrow (b \circ o g) \circ h = b \circ o (g \otimes h) \)

\begin{verbatim}
begin

lemma \texttt{act-act}:
assumes \( g \in \text{carrier } G \) and \( h \in \text{carrier } G \)
shows \( (b \circ o g) \circ h = b \circ o (g \otimes h) \)
proof –
from \( g \in \text{carrier } G \) and \( h \in \text{carrier } G \) and \texttt{act-act\'}
show \( (b \circ o g) \circ h = b \circ o (g \otimes h) \) by simp
qed

lemma \texttt{act-act-inv [simp]}:
assumes \( g \in \text{carrier } G \)
shows \( b \circ o g \circ o \text{ inv } g = b \)
proof –
from \( g \in \text{carrier } G \) have \text{ inv } g \in \text{carrier } G \ by \ (\text{rule } \text{ inv } - \text{ closed})
with \( g \in \text{carrier } G \) have \( b \circ o g \circ o \text{ inv } g = b \circ o g \otimes \text{ inv } g \) by \ (\text{rule } \text{ act } - \text{ act})
with \( g \in \text{carrier } G \) show \( b \circ o g \circ o \text{ inv } g = b \) by simp
qed

lemma \texttt{act-inv-act [simp]}:
assumes \( g \in \text{carrier } G \)
shows \( b \circ o \text{ inv } g \circ o g = b \)
using \( g \in \text{carrier } G \) and \texttt{act-act-inv [of inv } g\] by simp

lemma \texttt{act-inv-iff}:
assumes \( g \in \text{carrier } G \)
shows \( b \circ o \text{ inv } g = c \leftrightarrow b = c \circ o g \)
proof
assume \( b \circ o \text{ inv } g = c \)
hence \( b \circ o \text{ inv } g \circ o g = c \circ o g \) by simp
with \( g \in \text{carrier } G \) show \( b = c \circ o g \) by simp
next
assume \( b = c \circ o g \)
hence \( b \circ o \text{ inv } g = c \circ o g \circ o \text{ inv } g \) by simp
with \( g \in \text{carrier } G \) show \( b \circ o \text{ inv } g = c \) by simp
qed

end

end

7 Projective geometry

theory \texttt{Projective}
imports \texttt{Linear-Algebra2}
7.1 Proportionality on non-zero vectors

context vector-space

definition proportionality :: ('b × 'b) set where
  proportionality ≜ \{(x, y). x ≠ 0 ∧ y ≠ 0 ∧ (∃ k. x = scale k y)}

definition non-zero-vectors :: 'b set where
  non-zero-vectors ≜ \{x. x ≠ 0\}

lemma proportionality-refl-on: refl-on local.non-zero-vectors local.proportionality
proof –
  have local.proportionality ⊆ local.non-zero-vectors × local.non-zero-vectors
    unfolding proportionality-def non-zero-vectors-def
    by auto
  moreover have ∀ x∈ local.non-zero-vectors. (x, x) ∈ local.proportionality
  proof
    fix x
    assume x ∈ local.non-zero-vectors
    hence x ≠ 0 unfolding non-zero-vectors-def ..
    moreover have x = scale 1 x by simp
    ultimately show (x, x) ∈ local.proportionality
      unfolding proportionality-def
      by blast
    qed
  ultimately show refl-on local.non-zero-vectors local.proportionality
  unfolding refl-on-def ..
  qed

lemma proportionality-sym: sym local.proportionality
proof –
  { fix x y
    assume (x, y) ∈ local.proportionality
    hence x ≠ 0 and y ≠ 0 and ∃ k. x = scale k y
      unfolding proportionality-def
      by simp+
    from (∃ k. x = scale k y) obtain k where x = scale k y by auto
    with (x ≠ 0) have k ≠ 0 by simp
    with (x = scale k y) have y = scale (1/k) x by simp
    with (y ≠ 0) and (y ≠ 0) have (y, x) ∈ local.proportionality
      unfolding proportionality-def
      by auto
  }
  thus sym local.proportionality
unfolding \textit{sym-def} by blast qed

\textbf{lemma} \textit{proportionality-trans}: \textit{trans local.proportionality}
\begin{proof}
\{ fix \( x \), \( y \), \( z \) \\
assume \((x, y) \in \text{local.proportionality}\) \textbf{and} \((y, z) \in \text{local.proportionality}\) \\
hence \( x \neq 0 \) \textbf{and} \( z \neq 0 \) \textbf{and} \( \exists j. \) \( x = \text{scale} \) \( j \) \( y \) \textbf{and} \( \exists k. \) \( y = \text{scale} \) \( k \) \( z \) \\
unfolding \textit{proportionality-def} by simp+ \\
from \( \exists j. \) \( x = \text{scale} \) \( j \) \( y \) \textbf{and} \( \exists k. \) \( y = \text{scale} \) \( k \) \( z \) \\
obtain \( j \) \textbf{and} \( k \) where \( x = \text{scale} \) \( j \) \( y \) \textbf{and} \( y = \text{scale} \) \( k \) \( z \) by auto+ \\
hence \( x = \text{scale} \) \( (j \ast k) \) \( z \) by simp \\
with \((x \neq 0)\) \textbf{and} \((z \neq 0)\) \textbf{have} \((x, z) \in \text{local.proportionality}\) \\
unfolding \textit{proportionality-def} by auto \\
\}
thus \textit{trans local.proportionality} \\
unfolding \textit{trans-def} by blast qed

\textbf{theorem} \textit{proportionality-equiv}: \textit{equiv local.non-zero-vectors local.proportionality}
\begin{proof}
\textbf{unfolding} \textit{equiv-def} by (simp add:
\textit{proportionality-refl-on} \\
\textit{proportionality-sym} \\
\textit{proportionality-trans})
\end{proof}

\textbf{definition} \textit{invertible-proportionality} :: \\
\((\text{real}^{`\cdot`n::finite}^{`\cdot`n}) \times (\text{real}^{`\cdot`n}^{`\cdot`n})\) \textbf{set} where \\
\textit{invertible-proportionality} \(\triangleq\) \\
\textit{real-vector.proportionality} \(\cap\) \((\text{Collect invertible} \times \text{Collect invertible})\)

\textbf{lemma} \textit{invertible-proportionality-equiv}:
\(\text{equiv} \) \((\text{Collect invertible} :: (\text{real}^{`\cdot`n::finite}^{`\cdot`n}) \textbf{set})\) \\
\textit{invertible-proportionality} \\
(is \(\text{equiv}\) \(\_\text{invs}\) -) \\
\begin{proof}
from \textit{zero-not-invertible} \\
have \textit{real-vector.non-zero-vectors} \(\cap\) \(\_\text{invs} = \_\text{invs}\) \\
unfolding \textit{real-vector.non-zero-vectors-def} by auto \\
from \textit{equiv-restrict} \textbf{and} \textit{real-vector.proportionality-equiv} \\
have \textit{equiv} \((\text{real-vector.non-zero-vectors} \cap \_\text{invs})\) \textit{invertible-proportionality} \\
unfolding \textit{invertible-proportionality-def}
by auto
with (real-vector.non-zero-vectors ∩ ?invs = ?invs)
show equiv ?invs invertible-proportionality
  by simp
qed

7.2 Points of the real projective plane

typedef proj2 = (real-vector.non-zero-vectors :: (real^3) set) // real-vector.proportionality

proof
  have (axis 1 1 :: real^3) ∈ real-vector.non-zero-vectors
    unfolding real-vector.non-zero-vectors-def
    by (simp add: axis-def vec-eq-iff[where 'a=real])
  thus real-vector.proportionality " {axis 1 1} ∈ (real-vector.non-zero-vectors :: (real^3) set) // real-vector.proportionality
    unfolding quotient-def
    by auto
qed

definition proj2-rep :: proj2 ⇒ real^3 where
  proj2-rep x ≜ ϵ v. v ∈ Rep-proj2 x

definition proj2-abs :: real^3 ⇒ proj2 where
  proj2-abs v ≜ Abs-proj2 (real-vector.proportionality " v)

lemma proj2-rep-in: proj2-rep x ∈ Rep-proj2 x
proof
  let ?v = proj2-rep x
  from quotient-element-nonempty and
    real-vector.proportionality-equiv and
    Rep-proj2 [of x]
  have ∃ w. w ∈ Rep-proj2 x
    by auto
  with someI-ex [of λ z. z ∈ Rep-proj2 x]
  show ?v ∈ Rep-proj2 x
    unfolding proj2-rep-def
    by simp
qed

lemma proj2-rep-non-zero: proj2-rep x ≠ 0
proof
  from
    Union-quotient [of real-vector.non-zero-vectors real-vector.proportionality]
    and real-vector.proportionality-equiv
    and Rep-proj2 [of x] and proj2-rep-in [of x]
  have proj2-rep x ∈ real-vector.non-zero-vectors
    unfolding quotient-def
    by auto
  thus proj2-rep x ≠ 0

45
lemma proj2-rep-abs:
fixes v :: realˆ3
assumes v ∈ real-vector.non-zero-vectors
shows (v, proj2-rep (proj2-abs v)) ∈ real-vector.proportionality
proof –
from ⟨v ∈ real-vector.non-zero-vectors⟩ have real-vector.proportionality "\{v\} ∈ (real-vector.non-zero-vectors :: (realˆ3)) set)" by auto
unfolding quotient-def by simp
with Abs-proj2-inverse
have Proj-proj2 (proj2-abs v) = real-vector.proportionality "\{v\}" by auto
unfolding proj2-abs-def by simp
with proj2-rep-in
have proj2-rep (proj2-abs v) ∈ real-vector.proportionality "\{v\} by auto
thus ⟨v, proj2-rep (proj2-abs v)⟩ ∈ real-vector.proportionality by simp
qed

lemma proj2-abs-rep: proj2-abs (proj2-rep x) = x
proof –
from partition-Image-element [of real-vector.non-zero-vectors
real-vector.proportionality
Rep-proj2 x
proj2-rep x]
and real-vector.proportionality-equiv
and Rep-proj2 [of x] and proj2-rep-in [of x]
have real-vector.proportionality "\{proj2-rep x\} = Rep-proj2 x
by simp
with Rep-proj2-inverse show proj2-abs (proj2-rep x) = x
unfolding proj2-abs-def by simp
qed

lemma proj2-abs-mult:
assumes c ≠ 0
shows proj2-abs (c *R v) = proj2-abs v
proof cases
assume v = 0
thus proj2-abs (c *R v) = proj2-abs v by simp
next
assume v ≠ 0
with ⟨c ≠ 0⟩, have ⟨c *R v, v⟩ ∈ real-vector.proportionality
and $c \cdot v \in \text{real-vector}.\text{non-zero-vectors}$
and $v \in \text{real-vector}.\text{non-zero-vectors}$

unfolding real-vector.proportionality-def
and real-vector.non-zero-vectors-def
by simp-all
with eq-equiv-class-iff
[of real-vector.non-zero-vectors
real-vector.proportionality
c \cdot v
v]
and real-vector.proportionality-equiv
have real-vector.proportionality "{c \cdot v} =
real-vector.proportionality "{v}
by simp
thus proj2-abs (c \cdot v) = proj2-abs v
unfolding proj2-abs-def
by simp
qed

lemma proj2-abs-mult-rep:
assumes $c \neq 0$
shows proj2-abs (c \cdot proj2-rep x) = x
using proj2-abs-mult and proj2-abs-rep and assms
by simp

lemma proj2-rep-inj: inj proj2-rep
by (simp add: inj-on-inverseI [of UNIV proj2-abs proj2-rep] proj2-abs-rep)

lemma proj2-rep-abs2:
assumes $v \neq 0$
shows $\exists k. k \neq 0 \land proj2-rep (proj2-abs v) = k \cdot v$
proof -
from proj2-rep-abs [of v] and (v \neq 0)
have $(v, proj2-rep (proj2-abs v)) \in \text{real-vector.proportionality}$
unfolding real-vector.non-zero-vectors-def
by simp
then obtain c where $v = c \cdot R \text{proj2-rep (proj2-abs v)}$
unfolding real-vector.proportionality-def
by auto
with (v \neq 0) have $c \neq 0$ by auto
hence $1/c \neq 0$ by simp
from (v = c \cdot R \text{proj2-rep (proj2-abs v)})
have $(1/c) \cdot v = (1/c) \cdot R \text{ c \cdot R \text{proj2-rep (proj2-abs v)}}$
by simp
with (c \neq 0) have proj2-rep (proj2-abs v) = (1/c) \cdot R v by simp

with (1/c \neq 0) show $\exists k. k \neq 0 \land proj2-rep (proj2-abs v) = k \cdot v$
by blast

47
lemma proj2-abs-abs-mult:
  assumes proj2-abs v = proj2-abs w and w ≠ 0
  shows ∃ c. v = c *R w
proof
cases
  assume v = 0
  hence v = 0 *R w by simp
  thus ∃ c. v = c *R w ..
next
  assume v ≠ 0
  from ⟨proj2-abs v = proj2-abs w⟩
  have proj2-rep (proj2-abs v) = proj2-rep (proj2-abs w) by simp
  with proj2-rep-abs2 and ⟨w ≠ 0⟩
  obtain k where proj2-rep (proj2-abs v) = k *R w by auto
  with proj2-rep-abs2 [of v] and ⟨w ≠ 0⟩
  obtain j where j ≠ 0 and j *R v = k *R w by auto
  hence (1/j) *R j *R v = (1/j) *R k *R w by simp
  with ⟨j ≠ 0⟩ have v = (k/j) *R w by simp
  thus ∃ c. v = c *R w ..
qed

lemma dependent-proj2-abs:
  assumes p ≠ 0 and q ≠ 0 and i ≠ 0 ∨ j ≠ 0 and i *R p + j *R q = 0
  shows proj2-abs p = proj2-abs q
proof
  have i ≠ 0
  proof
    assume i = 0
    with ⟨i ≠ 0 ∨ j ≠ 0⟩ have j ≠ 0 by simp
    with ⟨i *R p + j *R q = 0; and ⟨q ≠ 0⟩ have i *R p ≠ 0 by auto
    with ⟨i = 0⟩ show False by simp
  qed
  with ⟨p ≠ 0⟩ and ⟨i *R p + j *R q = 0⟩ have j ≠ 0 by auto
  from ⟨i ≠ 0⟩
  have proj2-abs p = proj2-abs (i *R p) by (rule proj2-abs-mult [symmetric])
  also from ⟨i *R p + j *R q = 0; and proj2-abs-mult [of -1 j *R q]⟩
  have ... = proj2-abs (j *R q) by (simp add: algebra-simps [symmetric])
  also from ⟨j ≠ 0⟩ have ... = proj2-abs q by (rule proj2-abs-mult)
  finally show proj2-abs p = proj2-abs q .
qed

lemma proj2-rep-dependent:
  assumes i *R proj2-rep v + j *R proj2-rep w = 0
  (is i *R ?p + j *R ?q = 0)
  and i ≠ 0 ∨ j ≠ 0
  shows v = w
proof –
have \( ?p \neq 0 \) and \( ?q \neq 0 \) by (rule proj2-rep-non-zero)+
with \((i \neq 0 \lor j \neq 0)\) and \(i \cdot R ?p + j \cdot R ?q = 0\):
\[\text{have proj2-abs} \ ?p = \text{proj2-abs} \ ?q \ \text{by} \ (\text{simp add: dependent-proj2-abs})\]
\[\text{thus} \ v = \ w \ \text{by} \ (\text{simp add: proj2-abs-rep})\]
qed

\textbf{lemma proj2-rep-independent:}
assumes \( p \neq q \)
shows \( \text{independent} \ \{\text{proj2-rep} \ p, \text{proj2-rep} \ q\} \)
proof
let \( ?p' = \text{proj2-rep} \ p \)
let \( ?q' = \text{proj2-rep} \ q \)
let \( ?S = \{?p', ?q'\} \)
assume \( \text{dependent} \ ?S \)
\[\text{from proj2-rep-inj and} \ \langle p \neq q \rangle \ \text{have} \ ?p' \neq ?q' \]
\quad unfolding inj-on-def
by auto
with \( \text{dependent-explicit-2} \ [\text{of} \ ?p' \ ?q'] \ \text{and} \ \langle \text{dependent} \ ?S \rangle \)
obtain \( i \) and \( j \) where \( i \cdot R ?p' + j \cdot R ?q' = 0 \) and \( i \neq 0 \lor j \neq 0 \)
by (simp add: scalar-equiv) auto
with \( \text{proj2-rep-dependent} \ \text{have} \ p = q \ \text{by simp} \)
with \( \langle p \neq q \rangle \ \text{show} \ False .. \)
qed

\section{7.3 Lines of the real projective plane}

\textbf{definition proj2-Col ::} \([\text{proj2}, \text{proj2}, \text{proj2}] \Rightarrow \text{bool} \ where\)
\[\text{proj2-Col} \ p \ q \ r \ \triangleq \ \exists \ i \ j \ k. \ i \cdot R \text{proj2-rep} \ p + j \cdot R \text{proj2-rep} \ q + k \cdot R \text{proj2-rep} \ r = 0 \]
\[\land \ (i \neq 0 \lor j \neq 0 \lor k \neq 0)\]

\textbf{lemma proj2-Col-abs:}
assumes \( p \neq 0 \) and \( q \neq 0 \) and \( r \neq 0 \) and \( i \neq 0 \lor j \neq 0 \lor k \neq 0 \)
and \( i \cdot R \ p + j \cdot R \ q + k \cdot R \ r = 0 \)
shows \( \text{proj2-Col} \ \langle\text{proj2-abs} \ p\rangle \ \langle\text{proj2-abs} \ q\rangle \ \langle\text{proj2-abs} \ r\rangle \)
(\(\text{is proj2-Col} \ ?pp \ ?pq \ ?pr\))
proof --
from \( \langle p \neq 0 \rangle \ \text{and proj2-rep-abs2} \)
obtain \( i' \) where \( i' \neq 0 \) and \( \text{proj2-rep} \ ?pp = i' \cdot R \ p \) (\(\text{is} \ ?rp = \cdot\)) by auto
from \( \langle q \neq 0 \rangle \ \text{and proj2-rep-abs2} \)
obtain \( j' \) where \( j' \neq 0 \) and \( \text{proj2-rep} \ ?pq = j' \cdot R \ q \) (\(\text{is} \ ?rq = \cdot\)) by auto
from \( \langle r \neq 0 \rangle \ \text{and proj2-rep-abs2} \)
obtain \( k' \) where \( k' \neq 0 \) and \( \text{proj2-rep} \ ?pr = k' \cdot R \ r \) (\(\text{is} \ ?rr = \cdot\)) by auto
with \( \langle i \cdot R \ p + j \cdot R \ q + k \cdot R \ r = 0 \rangle \)
and \( \langle i' \neq 0 \rangle \ \text{and} \ \langle\text{proj2-rep} \ ?pp = i' \cdot R \ p\rangle \)
and \( \langle j' \neq 0 \rangle \ \text{and} \ \langle\text{proj2-rep} \ ?pq = j' \cdot R \ q\rangle \)
have \( \langle i/j' \cdot R \ ?rp + (j/j') \cdot R \ ?rq + (k/k') \cdot R \ ?rr = 0 \rangle \ \text{by simp} \)
from \( \langle i' \neq 0 \rangle \ \text{and} \ \langle j' \neq 0 \rangle \ \text{and} \ \langle k' \neq 0 \rangle \ \text{and} \ \langle i \neq 0 \lor j \neq 0 \lor k \neq 0 \rangle \)
have \( i/i' \neq 0 \lor j/j' \neq 0 \lor k/k' \neq 0 \) by simp

with \((i/i') * \mathbb{R} ?rp + (j/j') * \mathbb{R} ?rq + (k/k') * \mathbb{R} ?rr = 0\)

show proj2-Col \( ?pp \ ?pq \ ?pr \) by (unfold proj2-Col-def, best)

qed

lemma proj2-Col-permute:
  assumes proj2-Col a b c
  shows proj2-Col a c b
  and proj2-Col b a c

proof
  let \( ?a' = \text{proj2-rep } a \)
  let \( ?b' = \text{proj2-rep } b \)
  let \( ?c' = \text{proj2-rep } c \)
  from (proj2-Col a b c)
  obtain i and j and k where
    \( i * \mathbb{R} ?a' + j * \mathbb{R} ?b' + k * \mathbb{R} ?c' = 0 \)
    and \( i \neq 0 \lor j \neq 0 \lor k \neq 0 \)
  unfolding proj2-Col-def
  by auto

  from \((i * \mathbb{R} ?a' + j * \mathbb{R} ?b' + k * \mathbb{R} ?c' = 0)\)
  have \( i * \mathbb{R} ?a' + k * \mathbb{R} ?c' + j * \mathbb{R} ?b' = 0 \)
    and \( j * \mathbb{R} ?b' + i * \mathbb{R} ?a' + k * \mathbb{R} ?c' = 0 \)
    by (simp-all add: ac-simps)
  moreover from \( i \neq 0 \lor j \neq 0 \lor k \neq 0 \)
  have \( i \neq 0 \lor k \neq 0 \lor j \neq 0 \) and \( j \neq 0 \lor i \neq 0 \lor k \neq 0 \) by auto
  ultimately show proj2-Col a c b and proj2-Col b a c
    unfolding proj2-Col-def
    by auto

qed

lemma proj2-Col-coincide: proj2-Col a a c

proof
  have \( 1 * \mathbb{R} \text{proj2-rep } a + (-1) * \mathbb{R} \text{proj2-rep } a + 0 * \mathbb{R} \text{proj2-rep } c = 0 \)
    by simp
  moreover have \((1::\mathbb{R}) \neq 0\) by simp
  ultimately show proj2-Col a a c
    unfolding proj2-Col-def
    by blast

qed

lemma proj2-Col-iff:
  assumes \( a \neq r \)
  shows proj2-Col a r t \iff
    \( t = a \lor (3 \ i. \ t = \text{proj2-abs } (i * \mathbb{R} \ (\text{proj2-rep } a) + (\text{proj2-rep } r))) \)

proof
  let \( ?a' = \text{proj2-rep } a \)
  let \( ?r' = \text{proj2-rep } r \)
  let \( ?t' = \text{proj2-rep } t \)

50
\{ \text{assume } \text{proj2-Col } a \; r \; t \}
\text{then obtain } h \text{ and } j \text{ and } k \text{ where }
\begin{align*}
h \ast_R \; ?a' + j \ast_R \; ?r' + k \ast_R \; ?t' &= 0 \\
\text{and } h \neq 0 &\lor j \neq 0 \lor k \neq 0
\end{align*}
\text{unfolding } \text{proj2-Col-def} 
\text{by auto}
\text{show } t = a \lor (\exists \; i . \; t = \text{proj2-abs} \; (i \ast_R \; ?a' + \; ?r')) 
\text{proof cases}
\begin{itemize}
\item \text{assume } j = 0 
\begin{itemize}
\item \text{with } h \neq 0 \lor j \neq 0 \lor k \neq 0 \text{ have } h \neq 0 \lor k \neq 0 \text{ by simp}
\item \text{with } \text{proj2-rep-dependent} 
\begin{itemize}
\item \text{and } h \ast_R \; ?a' + j \ast_R \; ?r' + k \ast_R \; ?t' = 0 \\
\item \text{and } (j = 0)
\end{itemize}
\item \text{have } t = a \text{ by auto}
\item \text{thus } t = a \lor (\exists \; i . \; t = \text{proj2-abs} \; (i \ast_R \; ?a' + \; ?r')) ..
\end{itemize}
\end{itemize}
\text{next}
\begin{itemize}
\item \text{assume } j \neq 0 
\item \text{have } k \neq 0 
\item \text{proof (rule ccontr)}
\begin{itemize}
\item \text{assume } \neg k \neq 0 
\item \text{with } \text{proj2-rep-dependent} 
\begin{itemize}
\item \text{and } h \ast_R \; ?a' + j \ast_R \; ?r' + k \ast_R \; ?t' = 0 \\
\item \text{and } (j \neq 0)
\end{itemize}
\item \text{have } a = r \text{ by simp}
\item \text{with } (a \neq r) \text{ show False ..}
\end{itemize}
\end{itemize}
\text{qed}
\text{from } (h \ast_R \; ?a' + j \ast_R \; ?r' + k \ast_R \; ?t' = 0)
\text{have } h \ast_R \; ?a' + j \ast_R \; ?r' + k \ast_R \; ?t' - k \ast_R \; ?t' = -k \ast_R \; ?t' \text{ by simp}
\text{hence } h \ast_R \; ?a' + j \ast_R \; ?r' = -k \ast_R \; ?t' \text{ by simp}
\text{with } \text{proj2-abs-mult-rep} \; [\text{of } -k] \text{ and } (k \neq 0)
\text{have } \text{proj2-abs} \; (h \ast_R \; ?a' + j \ast_R \; ?r') = t \text{ by simp}
\text{with } \text{proj2-abs-mult} \; [\text{of } 1/j \; h \ast_R \; ?a' + j \ast_R \; ?r'] \text{ and } (j \neq 0)
\text{have } \text{proj2-abs} \; ((h/j) \ast_R \; ?a' + \; ?r') = t 
\text{by } (\text{simp add: scaleR-right-distrib})
\text{hence } (\exists \; i . \; t = \text{proj2-abs} \; (i \ast_R \; ?a' + \; ?r')) \text{ by auto}
\text{thus } t = a \lor (\exists \; i . \; t = \text{proj2-abs} \; (i \ast_R \; ?a' + \; ?r')) ..
\text{qed}
\} \\
\{ \text{assume } t = a \lor (\exists \; i . \; t = \text{proj2-abs} \; (i \ast_R \; ?a' + \; ?r')) 
\text{show } \text{proj2-Col } a \; r \; t 
\text{proof cases}
\begin{itemize}
\item \text{assume } t = a 
\item \text{with } \text{proj2-Col-coincide} \text{ and } \text{proj2-Col-permute}
\text{show } \text{proj2-Col } a \; r \; t \text{ by blast}
\end{itemize}
\text{next}
assume \( t \neq a \)
with \( (t = a \lor (\exists \ i. \ t = \text{proj2-abs} (i \star_R a' + \tau'))) \)
\begin{align*}
\text{obtain } i \text{ where } t &= \text{proj2-abs} (i \star_R a' + \tau') \text{ by auto} \\
\text{from } \text{proj2-rep-dependent} \ [\text{of } i \ a \ 1 \ r] &\text{ and } (a \neq r) \\
\text{have } i \star_R a' + \tau' \neq 0 \text{ by auto} \\
\text{with } \text{proj2-rep-abs2} &\text{ and } (t = \text{proj2-abs} (i \star_R a' + \tau')) \\
\text{obtain } j \text{ where } \tau'i' = j \star_R (i \star_R a' + \tau') \text{ by auto} \\
\text{hence } \tau'i' - \tau'i' = (j \star i) \star_R a' + j \star_R \tau' + (-1) \star_R \tau'i' \\
&\text{by (simp add: scaleR-right-distrib)} \\
\text{hence } (j \star i) \star_R a' + j \star_R \tau' + (-1) \star_R \tau'i' = 0 \text{ by simp} \\
\text{have } \exists \ h \ j. \ h \star_R a' + j \star_R \tau' + k \star_R \tau'i' = 0 \\
&\land \ (h \neq 0 \lor j \neq 0 \lor k \neq 0) \\
\text{proof standard}+ \\
&\text{from } ((j \star i) \star_R a' + j \star_R \tau' + (-1) \star_R \tau'i' = 0) \\
&\text{show } (j \star i) \star_R a' + j \star_R \tau' + (-1) \star_R \tau'i' = 0 . \\
&\text{show } j \star i \neq 0 \lor j \neq 0 \lor (-1::real) \neq 0 \text{ by simp} \\
\text{qed} \\
\text{thus } \text{proj2-Col a r t} \\
\text{unfolding proj2-Col-def} . \\
\text{qed} \\
\end{align*}

\text{definition } \text{proj2-Col-coeff} :: \text{proj2} \Rightarrow \text{proj2} \Rightarrow \text{proj2} \Rightarrow \text{real} \text{ where} \\
\text{proj2-Col-coeff } a \ r \ t \triangleq \ i. \ t = \text{proj2-abs} (i \star_R \text{proj2-rep a} + \text{proj2-rep r}) \\

\text{lemma } \text{proj2-Col-coeff}: \\
\text{assumes } \text{proj2-Col a r t} \text{ and } a \neq r \text{ and } t \neq a \\
\text{shows } t = \text{proj2-abs} ((\text{proj2-Col-coeff} a \ r \ t) \star_R \text{proj2-rep a} + \text{proj2-rep r}) \\
\text{proof} - \\
&\text{from } (a \neq r) \text{ and } (\text{proj2-Col a r t}) \text{ and } t \neq a \text{ and } \text{proj2-Col-iff} \\
&\text{have } \exists \ i. \ t = \text{proj2-abs} (i \star_R \text{proj2-rep a} + \text{proj2-rep r}) \text{ by simp } \\
&\text{thus } t = \text{proj2-abs} ((\text{proj2-Col-coeff} a \ r \ t) \star_R \text{proj2-rep a} + \text{proj2-rep r}) \\
&\text{by (unfold proj2-Col-coeff-def) (rule someI-ex)} \\
\text{qed} \\

\text{lemma } \text{proj2-Col-coeff-unique'}: \\
\text{assumes } a \neq 0 \text{ and } r \neq 0 \text{ and } \text{proj2-abs a} \neq \text{proj2-abs r} \\
\text{and } \text{proj2-abs} (i \star_R a + r) = \text{proj2-abs} (j \star_R a + r) \\
\text{shows } i = j \\
\text{proof} - \\
&\text{from } (a \neq 0) \text{ and } (r \neq 0) \text{ and } (\text{proj2-abs a} \neq \text{proj2-abs r}) \\
&\text{and } \text{dependent-proj2-abs} \ [\text{of } a \ r - 1] \\
&\text{have } i \star_R a + r \neq 0 \text{ and } j \star_R a + r \neq 0 \text{ by auto} \\
&\text{with } \text{proj2-rep-abs2} \ [\text{of } i \star_R a + r] \\
&\text{and } \text{proj2-rep-abs2} \ [\text{of } j \star_R a + r] \\
&\text{obtain } k \text{ and } l \text{ where } k \neq 0 \\
&\text{and } \text{proj2-rep} \ (\text{proj2-abs} (i \star_R a + r)) = k \star_R (i \star_R a + r) \\
&\text{and } \text{proj2-rep} \ (\text{proj2-abs} (j \star_R a + r)) = l \star_R (j \star_R a + r) \\
\text{52}
by auto
with ⟨proj2-abs (i *R a + r) = proj2-abs (j *R a + r)⟩
have (k * i) *R a + k *R r = (l * j) *R a + l *R r
  by (simp add: scaleR-right-distrib)
  hence (k * i - l * j) *R a + (k - l) *R r = 0
  by (simp add: algebra-simps vec-eq-iff)
with ⟨a ≠ 0 ⟩ and ⟨r ≠ 0 ⟩ and ⟨proj2-abs a ≠ proj2-abs r ⟩
  and ⟨proj2-abs-b (a r k * i - l * j k - l)⟩
  have k * i - l * j = 0 and k - l = 0 by auto
from ⟨k - l = 0 ⟩ have k = l by simp
with ⟨k * i - l * j = 0 ⟩ have k * i = k * j by simp
with ⟨k ≠ 0 ⟩ show i = j by simp
qed

lemma proj2-Col-coeff-unique:
  assumes a ≠ r
  and proj2-abs (i *R proj2-rep a + proj2-rep r) = proj2-abs (j *R proj2-rep a + proj2-rep r)
  shows i = j
proof -
  let ?a' = proj2-rep a
  let ?r' = proj2-rep r
  have ?a' ≠ 0 and ?r' ≠ 0 by (rule proj2-rep-non-zero)+
   from ⟨a ≠ r ⟩ have proj2-abs ?a' ≠ proj2-abs ?r' by (simp add: proj2-abs-rep)
  with ⟨?a' ≠ 0 ⟩ and ⟨?r' ≠ 0 ⟩
   and ⟨proj2-abs (i *R ?a' + ?r') = proj2-abs (j *R ?a' + ?r') ⟩
  show i = j by simp
qed

datatype proj2-line = P2L proj2

definition L2P :: proj2-line ⇒ proj2 where
  L2P l ≜ case l of P2L p ⇒ p

lemma L2P-P2L [simp]: L2P (P2L p) = p
  unfolding L2P-def
  by simp

lemma P2L-L2P [simp]: P2L (L2P l) = l
  by (induct l) simp

lemma L2P-inj [simp]:
  assumes L2P l = L2P m
  shows l = m
  using P2L-L2P [of l] and assms
  by simp
lemma \( P2L\)-to-L2P: \( P2L\ p = l \iff p = L2P\ l \)

proof
  assume \( P2L\ p = l \)
  hence \( L2P\ (P2L\ p) = L2P\ l \) by simp
  thus \( p = L2P\ l \) by simp
next
  assume \( p = L2P\ l \)
  thus \( P2L\ p = l \) by simp

qed

definition proj2-line-abs :: real\(^3\) ⇒ proj2-line
where
  proj2-line-abs \( v \) ≜ \( P2L\ (proj2-abs\ v) \)

definition proj2-line-rep :: proj2-line ⇒ real\(^3\)
where
  proj2-line-rep \( l \) ≜ proj2-rep\ ((L2P\ l))

lemma proj2-line-rep-abs:
  assumes \( v \neq 0 \)
  shows \( \exists k. k \neq 0 \land proj2-line-rep\ (proj2-line-abs\ v) = k \ast_R v \)
  unfolding proj2-line-rep-def and proj2-line-abs-def
  using proj2-rep-abs2 and \( \langle v \neq 0 \rangle \)
  by simp

lemma proj2-line-rep-abs-flip [simp]: proj2-line-abs\ (proj2-line-rep\ l) = l
  unfolding proj2-line-abs-def and proj2-line-rep-def
  by (simp add: proj2-abs-rep)

lemma proj2-line-rep-non-zero: proj2-line-rep\ l \neq 0
  unfolding proj2-line-rep-def
  using proj2-rep-non-zero
  by simp

lemma proj2-line-rep-dependent:
  assumes \( i \ast_R proj2-line-rep\ l + j \ast_R proj2-line-rep\ m = 0 \)
  and \( i \neq 0 \lor j \neq 0 \)
  shows \( l = m \)
  using proj2-rep-dependent [of \( i \) \( L2P\ l \) \( j \) \( L2P\ m \)] and assms
  unfolding proj2-line-rep-def
  by simp

lemma proj2-line-abs-mult:
  assumes \( k \neq 0 \)
  shows \( proj2-line-abs\ (k \ast_R v) = proj2-line-abs\ v \)
  unfolding proj2-line-abs-def
  using \( \langle k \neq 0 \rangle \)
  by (subst proj2-abs-mult) simp-all

lemma proj2-line-abs-abs-mult:
  assumes \( proj2-line-abs\ v = proj2-line-abs\ w \) and \( w \neq 0 \)
shows $\exists \ k. \ v = k *_R w$
using assms
by (unfold proj2-line-abs-def) (simp add: proj2-abs-abs-mult)

definition proj2-incident :: proj2 $\Rightarrow$ proj2-line $\Rightarrow$ bool where
proj2-incident p l $\triangleq$ $(\text{proj2-rep } p) \cdot (\text{proj2-line-rep } l) = 0$

lemma proj2-points-define-line:
shows $\exists \ l. \ \text{proj2-incident } p \ l \land \text{proj2-incident } q \ l$
proof
  let $?p' = \text{proj2-rep } p$
  let $?q' = \text{proj2-rep } q$
  let $?B = \{ ?p', ?q' \}$
  from card-suc-ge-insert $[\text{of } ?p' \{ ?q' \}]$ have $\text{card } ?B \leq 2$ by simp
  with dim-le-card $' [\text{of } ?B]$ have $\text{dim } ?B < 3$ by simp
  with lowdim-subset-hyperplane $' [\text{of } ?B]$ obtain $l' \ where \ l' \neq 0$ and span $?B \subseteq \{ x. \ l' \cdot x = 0 \}$ by auto
  let $?l = \text{proj2-line-abs } l'$
  let $?l'' = \text{proj2-line-rep } ?l$
  from proj2-line-rep-abs and $l' \neq 0$ obtain $k \ where \ ?l'' = k *_R l'$ by auto

have $?p' \in ?B$ and $?q' \in ?B$ by simp-all
with span-superset $[\text{of } ?B]$ and $(\text{span } ?B \subseteq \{ x. \ l' \cdot x = 0 \})$
hence $?p' \cdot l' = 0$ and $?q' \cdot l' = 0$ by (simp-all add: inner-commute)
with dot-scaleR-mult $(2) [\text{of } k l'']$ and $(?l'' = k *_R l')$
have proj2-incident $p ?l \land \text{proj2-incident } q ?l$
  unfolding proj2-incident-def
  by simp
thus $\exists \ l. \ \text{proj2-incident } p \ l \land \text{proj2-incident } q \ l$ by auto
qed

definition proj2-line-through :: proj2 $\Rightarrow$ proj2 $\Rightarrow$ proj2-line where
proj2-line-through $p \ q \triangleq \epsilon \ l. \ \text{proj2-incident } p \ l \land \text{proj2-incident } q \ l$

lemma proj2-line-through-incident:
shows proj2-incident $p \ (\text{proj2-line-through } p \ q)$
and proj2-incident $q \ (\text{proj2-line-through } p \ q)$
unfolding proj2-line-through-def
using proj2-points-define-line
  and someI-ex $[\text{of } \lambda l. \ \text{proj2-incident } p \ l \land \text{proj2-incident } q \ l]$
by simp-all

lemma proj2-line-through-unique:
assumes $p \neq q$ and proj2-incident $p \ l$ and proj2-incident $q \ l$
shows $l = \text{proj2-line-through } p \ q$
proof
  let $?l' = \text{proj2-line-rep } l$
let \( ?m = \text{proj2-line-through} \ p \ q \)
let \( ?m' = \text{proj2-line-rep} \ ?m \)
let \( ?p' = \text{proj2-rep} \ p \)
let \( ?q' = \text{proj2-rep} \ q \)
let \( ?A = \{ ?p', ?q' \} \)
let \( ?B = \text{insert} \ ?m' \ ?A \)

from \( \text{proj2-line-through-incident} \)
have \( \text{proj2-incident} \ p \ ?m \) and \( \text{proj2-incident} \ q \ ?m \) by simp-all
with \( \text{proj2-incident} \ p \ l \) and \( \text{proj2-incident} \ q \ l \)
have ortho: \( \forall w. w \in ?A \implies \text{orthogonal} \ ?m' \ w \)
\( \forall w. w \in ?A \implies \text{orthogonal} \ ?l' \ w \)

unfolding \( \text{proj2-incident-def} \) and \( \text{orthogonal-def} \)
by (metis empty-iff inner-commute insert-iff)+
from \( \text{proj2-rep-independent} \) and \( p \neq q \) have independent \( ?A \) by simp
from \( \text{proj2-line-rep-non-zero} \) have \( ?m' \neq 0 \) by simp
with \( \text{orthogonal-independent} \) (independent \( ?A \)) ortho
have independent \( ?B \) by auto

from \( \text{proj2-rep-inj} \) and \( p \neq q \) have \( ?p' \neq ?q' \)

unfolding inj-on-def
by auto

hence card \( ?A = 2 \) by simp
moreover have \( ?m' \notin ?A \)
using ortho(1) orthogonal-self \( \text{proj2-line-rep-non-zero} \) by auto
ultimately have card \( ?B = 3 \) by simp
with independent-is-basis \( \text{[of} ?B] \) and \( \text{(independent} ?B) \)

have is-basis \( ?B \) by simp
with basis-expand obtain \( c \) where \( ?l' = \sum v \in ?B. c \ v \ *_R \ v \) by auto
let \( ?l'' = ?l' - c \ ?m' *_R \ ?m' \)
from \( ?l' = \sum v \in ?B. c \ v \ *_R \ v \) and \( ?m' \notin ?A \)
have \( ?l'' = \sum v \in ?A. c \ v \ *_R \ v \) by simp
with orthogonal-sum \( \text{[of} ?A] \) ortho
have orthogonal \( ?l' \ ?l'' \) and orthogonal \( ?m' \ ?l'' \)

by (simp-all add: scalar-equiv)
from \( \text{(orthogonal} \ ?m' \ ?l'') \)
have orthogonal \( (c \ ?m' *_R \ ?m') \ ?l'' \) by (simp add: orthogonal-clauses)
with \( \text{(orthogonal} \ ?l' \ ?l'') \)
have orthogonal \( ?l'' \ ?l'' \) by (simp add: orthogonal-clauses)
with orthogonal-self-eq-0 \( \text{[of} ?l'\] have} \ ?l'' = 0 \) by simp
with \( \text{proj2-line-rep-dependent} \) \( \text{[of} l - c \ ?m' \ ?m] \) show \( l = ?m \) by simp

qed

lemma \( \text{proj2-incident-unique} \):

assumes \( \text{proj2-incident} \ p \ l \)
and \( \text{proj2-incident} \ q \ l \)
and \( \text{proj2-incident} \ p \ m \)
and \( \text{proj2-incident} \ q \ m \)

shows \( p = q \lor l = m \)

proof cases

assume \( p = q \)

56
thus \( p = q \lor l = m \).

next

assume \( p \neq q \)

with \( (\text{proj2-incident } p \ l) \) and \( (\text{proj2-incident } q \ l) \)

and \( \text{proj2-line-through-unique} \)

have \( l = \text{proj2-line-through } p \ q \) by simp

moreover from \( p \neq q \) and \( (\text{proj2-incident } p \ m) \) and \( (\text{proj2-incident } q \ m) \)

have \( m = \text{proj2-line-through } p \ q \) by (rule \( \text{proj2-line-through-unique} \))

ultimately show \( p = q \lor l = m \) by simp

qed

lemma \( \text{proj2-lines-define-point} \) : \( \exists \ p. \ \text{proj2-incident } p \ l \land \text{proj2-incident } p \ m \)

proof

let \( ?l' = \text{L2P } l \)

let \( ?m' = \text{L2P } m \)

from \( \text{proj2-points-define-line} \) [of \( ?l' \ ?m' \)]

obtain \( p' \) where \( \text{proj2-incident } ?l' \ p' \land \text{proj2-incident } ?m' \ p' \) by auto

hence \( \text{proj2-incident } (\text{L2P } p') \ l \land \text{proj2-incident } (\text{L2P } p') \ m \)

unfolding \( \text{proj2-incident-def} \) and \( \text{proj2-line-rep-def} \)

by (simp add: inner-commute)

thus \( \exists \ p. \ \text{proj2-incident } p \ l \land \text{proj2-incident } p \ m \) by auto

qed

definition \( \text{proj2-intersection} :: \text{proj2-line} \Rightarrow \text{proj2-line} \Rightarrow \text{proj2} \)

\[
\text{proj2-intersection } l \ m \triangleq \text{L2P} \ (\text{proj2-line-through } (\text{L2P } l) (\text{L2P } m))
\]

lemma \( \text{proj2-incident-switch} \):

assumes \( \text{proj2-incident } p \ l \)

shows \( \text{proj2-incident } (\text{L2P } l) (\text{P2L } p) \)

using assms

unfolding \( \text{proj2-incident-def} \) and \( \text{proj2-line-rep-def} \)

by (simp add: inner-commute)

lemma \( \text{proj2-intersection-incident} \):

shows \( \text{proj2-incident } (\text{proj2-intersection } l \ m) \ l \)

and \( \text{proj2-incident } (\text{proj2-intersection } l \ m) \ m \)

using \( \text{proj2-line-through-incident}(1) \) [of \( \text{L2P } l \ \text{L2P } m \)]

and \( \text{proj2-line-through-incident}(2) \) [of \( \text{L2P } m \ \text{L2P } l \)]

and \( \text{proj2-incident-switch} \) [of \( \text{L2P } l \)]

and \( \text{proj2-incident-switch} \) [of \( \text{L2P } m \)]

unfolding \( \text{proj2-intersection-def} \)

by simp-all

lemma \( \text{proj2-intersection-unique} \):

assumes \( l \neq m \) and \( \text{proj2-incident } p \ l \) and \( \text{proj2-incident } p \ m \)

shows \( p = \text{proj2-intersection } l \ m \)

proof

from \( l \neq m \) have \( \text{L2P } l \neq \text{L2P } m \) by auto

from \( (\text{proj2-incident } p \ l) \) and \( (\text{proj2-incident } p \ m) \)
and proj2-incident-switch
have proj2-incident (L2P l) (P2L p) and proj2-incident (L2P m) (P2L p)
by simp-all
with (L2P l ≠ L2P m); and proj2-line-through-unique
have P2L p = proj2-line-through (L2P l) (L2P m) by simp
thus p = proj2-intersection l m
unfolding proj2-intersection-def
by (simp add: P2L-to-L2P)
qed

lemma proj2-not-self-incident:
¬ (proj2-incident p (P2L p))
unfolding proj2-incident-def and proj2-line-rep-def
using proj2-rep-non-zero and inner-eq-zero-iff [of proj2-rep p]
by simp

lemma proj2-another-point-on-line:
∃ q. q ≠ p ∧ proj2-incident q l
proof –
let ?m = P2L p
let ?q = proj2-intersection l ?m
from proj2-intersection-incident
have proj2-incident ?q l and proj2-incident ?q ?m by simp-all
from (proj2-incident ?q ?m) and proj2-not-self-incident have ?q ≠ p by auto
with (proj2-incident ?q l) show ∃ q. q ≠ p ∧ proj2-incident q l by auto
qed

lemma proj2-another-line-through-point:
∃ m. m ≠ l ∧ proj2-incident p m
proof –
from proj2-another-point-on-line
obtain q where q ≠ L2P l ∧ proj2-incident q (P2L p) by auto
with proj2-incident-switch [of q P2L p]
have P2L q ≠ l ∧ proj2-incident p (P2L q) by auto
thus ∃ m. m ≠ l ∧ proj2-incident p m ..
qed

lemma proj2-incident-abs:
assumes v ≠ 0 and w ≠ 0
shows proj2-incident (proj2-abs v) (proj2-line-abs w) ⟷ v · w = 0
proof –
from (v ≠ 0) and proj2-rep-abs2
obtain j where j ≠ 0 and proj2-rep (proj2-abs v) = j ⋆R v by auto
from (w ≠ 0) and proj2-line-rep-abs
obtain k where k ≠ 0
   and proj2-line-rep (proj2-line-abs w) = k ⋆R w
   by auto
with (j ≠ 0) and (proj2-rep (proj2-abs v) = j ⋆R v)
show proj2-incident (proj2-abs v) (proj2-line-abs w) ⟷ v · w = 0
unfolding proj2-incident-def
by (simp add: dot-scaleR-mult)

lemma proj2-incident-left-abs:
assumes v ≠ 0
shows proj2-incident (proj2-abs v) l ⟷ v · (proj2-line-rep l) = 0
proof –
have proj2-line-rep l ≠ 0 by (rule proj2-line-rep-non-zero)
with ⟨v ≠ 0⟩ and proj2-incident-abs [of v proj2-line-rep l]
show proj2-incident (proj2-abs v) l ⟷ v · (proj2-line-rep l) = 0 by simp

lemma proj2-incident-right-abs:
assumes v ≠ 0
shows proj2-incident p (proj2-line-abs v) ⟷ (proj2-rep p) · v = 0
proof –
have proj2-rep p ≠ 0 by (rule proj2-rep-non-zero)
with ⟨v ≠ 0⟩ and proj2-incident-abs [of proj2-rep p v]
show proj2-incident p (proj2-line-abs v) ⟷ (proj2-rep p) · v = 0
by (simp add: proj2-abs-rep)

definition proj2-set-Col :: proj2 set ⇒ bool where
proj2-set-Col S ≜ ∃ l . ∀ p∈S . proj2-incident p l

lemma proj2-subset-Col:
assumes T ⊆ S and proj2-set-Col S
shows proj2-set-Col T
using ⟨T ⊆ S⟩ and ⟨proj2-set-Col S⟩
by (unfold proj2-set-Col-def) auto

definition proj2-no-3-Col :: proj2 set ⇒ bool where
proj2-no-3-Col S ≜ card S = 4 ∧ (∀ p∈S . ¬ proj2-set-Col (S − {p}))

lemma proj2-Col-iff-not-invertible:
proj2-Col p q r
⟷ ¬ invertible (vector [proj2-rep p, proj2-rep q, proj2-rep r] :: realˆ3ˆ3)
(is - ⟷ ¬ invertible (vector [?u, ?v, ?w]))

proof –
let ?M = vector [?u, ?v, ?w] :: realˆ3ˆ3
have proj2-Col p q r ⟷ (∃ x . x ≠ 0 ∧ x v* ?M = 0)
proof
assume proj2-Col p q r
then obtain i and j and k
  where i ≠ 0 ∨ j ≠ 0 ∨ k ≠ 0 and i *R ?u + j *R ?v + k *R ?w = 0
unfolding proj2-Col-def
by auto
let \(?x = \text{vector} \{ i, j, k \} :: \text{real}^3\) from \((i \neq 0 \lor j \neq 0 \lor k \neq 0)\)
have \(?x \neq 0\)
unfolding vector-def
by (simp add: vec-eq-iff forall-3)
moreover {
from \((i \cdot_R ?u + j \cdot_R ?v + k \cdot_R ?w = 0)\)
have \(?x v \cdot \?M = 0\)
unfolding vector-def and vector-matrix-mult-def
by (simp add: sum-3 vec-eq-iff algebra-simps )
}
ultimately show \(\exists \ x. \ x \neq 0 \land x v \cdot \?M = 0\) by auto

next
assume \(\exists \ x. \ x \neq 0 \land x v \cdot ?M = 0\)
then obtain \(x\) where \(x \neq 0\) and \(x v \cdot \?M = 0\)
by (simp add: vec-eq-iff forall-3)
moreover {
from \((x \neq 0)\)
have \(?i \neq 0 \lor ?j \neq 0 \lor ?k \neq 0\)
unfolding vector-def and sum-3 and vector-def
by (simp add: vec-eq-iff algebra-simps )
}
ultimately show proj2-Col \(p\) \(q\) \(r\)
unfolding proj2-Col-def
by auto

qed

also from matrix-right-invertible-ker [of \(?M\)]
have \(\ldots \iff \neg (\exists \ M'. \ ?M \cdot \?M' = \text{mat} 1)\) by auto
also from matrix-left-right-inverse
have \(\ldots \iff \neg \text{invertible} \ ?M\)
unfolding invertible-def
by auto
finally show proj2-Col \(p\) \(q\) \(r\) \iff \neg \text{invertible} \ ?M .

qed

lemma not-invertible-iff-proj2-set-Col:
\neg \text{invertible} (\text{vector} \{ \text{proj2-rep} \ p, \text{proj2-rep} \ q, \text{proj2-rep} \ r \} :: \text{real}^3^3)
\iff proj2-set-Col \{p,q,r\}
(is \neg \text{invertible} \ ?M \iff \_)

proof –
from left-invertible-iff-invertible
have \(\neg \text{invertible} \ ?M \iff \neg (\exists \ M'. \ ?M \cdot \?M' = \text{mat} 1)\) by auto
also from matrix-left-invertible-ker [of \(?M\)]
have \(\ldots \iff (\exists \ y. \ y \neq 0 \land \?M \cdot v y = 0)\) by auto
also have \(\ldots \iff (\exists \ l. \forall s \in \{p,q,r\}. \text{proj2-incident} s l)\)
proof
assume \(\exists \ y. \ y \neq 0 \land \?M \cdot v y = 0\)
then obtain \(y\) where \(y \neq 0\) and \(?M \cdot v y = 0\) by auto
let \(?l = \text{proj2-line-abs} \ y\)

from \(\langle ?M \ast v \ y = 0 \rangle\)

have \(\forall s \in \{p,q,r\}. \text{proj2-rep} \ s \ast y = 0\)

unfolding \(\text{vector-def}\)
and \(\text{matrix-vector-mult-def}\)
and \(\text{inner-vec-def}\)
and \(\text{sum-3}\)
by \((\text{simp add: vec-eq-iff forall-3})\)

with \(\langle y \neq 0 \rangle\) and \(\text{proj2-incident-right-abs}\)

have \(\forall s \in \{p,q,r\}. \text{proj2-incident} \ s \ ?l \ by \text{simp}\)

thus \(\exists l. \ \forall s \in \{p,q,r\}. \text{proj2-incident} \ s \ l \ ..\)

next
assume \(\exists l. \ \forall s \in \{p,q,r\}. \text{proj2-incident} \ s \ l\)
then obtain \(\ i \ where \ \forall s \in \{p,q,r\}. \text{proj2-incident} \ s \ l \ ..\)

let \(?y = \text{proj2-line-rep} \ l\)

have \(?y \neq 0\) by \((\text{rule proj2-line-rep-non-zero})\)

moreover \{
from \(\forall s \in \{p,q,r\}. \text{proj2-incident} \ s \ l\)

have \(?M \ast v \ ?y = 0\)

unfolding \(\text{vector-def}\)
and \(\text{matrix-vector-mult-def}\)
and \(\text{inner-vec-def}\)
and \(\text{sum-3}\)
and \(\text{proj2-incident-def}\)
by \((\text{simp add: vec-eq-iff})\) \}

ultimately show \(\exists y. \ y \neq 0 \land ?M \ast v \ y = 0\) by \text{auto}

qed

finally show \(\neg \ \text{invertible} \ ?M \longleftrightarrow \text{proj2-set-Col} \ \{p,q,r\}\)

unfolding \(\text{proj2-set-Col-def}\) .

qed

lemma \(\text{proj2-Col-iff-set-Col}\):

\(\text{proj2-Col} \ p \ q \ r \longleftrightarrow \text{proj2-set-Col} \ \{p,q,r\}\)

by \((\text{simp add: proj2-Col-iff-not-invertible not-invertible-iff-proj2-set-Col})\)

lemma \(\text{proj2-incident-Col}\):

assumes \(\text{proj2-incident} \ p \ l \ \text{and} \ \text{proj2-incident} \ q \ l \ \text{and} \ \text{proj2-incident} \ r \ l\)

shows \(\text{proj2-Col} \ p \ q \ r\)

proof

from \(\langle \text{proj2-incident} \ p \ l \rangle\) and \(\langle \text{proj2-incident} \ q \ l \rangle\) and \(\langle \text{proj2-incident} \ r \ l \rangle\)

have \(\text{proj2-set-Col} \ \{p,q,r\}\) by \((\text{unfold proj2-set-Col-def})\) \text{auto}\n
thus \(\text{proj2-Col} \ p \ q \ r\) by \((\text{subst proj2-Col-iff-set-Col})\)

qed

lemma \(\text{proj2-incident-iff-Col}\):

assumes \(p \neq q\) and \(\text{proj2-incident} \ p \ l \ \text{and} \ \text{proj2-incident} \ q \ l\)

shows \(\text{proj2-incident} \ r \ l \longleftrightarrow \text{proj2-Col} \ p \ q \ r\)

proof

61
assume proj2-incident r l
with ⟨proj2-incident p l⟩ and ⟨proj2-incident q l⟩
show proj2-Col p q r by (rule proj2-incident-Col)

next
assume proj2-Col p q r
hence proj2-Col p m and proj2-Col q m and proj2-Col r m
by simp
then obtain m where ∀ s∈{p,q,r}. proj2-incident s m
unfolding proj2-Col-def ..
hence proj2-incident p m and proj2-incident q m and proj2-incident r m
by simp-all
from ⟨p ≠ q⟩ and ⟨proj2-incident p l⟩ and ⟨proj2-incident q l⟩
and ⟨proj2-incident p m⟩ and ⟨proj2-incident q m⟩
and proj2-incident-unique
have m = l by auto
with ⟨proj2-incident r m⟩ show proj2-incident r l by simp
qed

lemma proj2-incident-iff:
assumes p ≠ q and proj2-incident p l and proj2-incident q l
shows proj2-incident r l
⟷ r = p ∨ (∃ k. r = proj2-abs (k *R proj2-rep p + proj2-rep q))
proof –
from ⟨p ≠ q⟩ and ⟨proj2-incident p l⟩ and ⟨proj2-incident q l⟩
have proj2-incident r l ⟷ proj2-Col p q r by (rule proj2-incident-iff-Col)
with ⟨p ≠ q⟩ and proj2-Col-iff
show proj2-incident r l
⟷ r = p ∨ (∃ k. r = proj2-abs (k *R proj2-rep p + proj2-rep q))
by simp
qed

lemma not-proj2-set-Col-iff-span:
assumes card S = 3
shows ¬ proj2-set-Col S ⟷ span (proj2-rep ` S) = UNIV
proof –
from ⟨card S = 3⟩ and choose-3 [of S]
obtain p and q and r where S = {p,q,r} by auto
let ?u = proj2-rep p
let ?v = proj2-rep q
let ?w = proj2-rep r
let ?M = vector [?u, ?v, ?w] :: real^3^3
from ⟨S = {p,q,r}⟩ and not-invertible-iff-proj2-set-Col [of p q r]
have ¬ proj2-set-Col S ⟷ invertible ?M by auto
also from left-invertible-iff-invertible
have ... ⟷ (∃ N. N ** ?M = mat 1) ..
also from matrix-left-invertible-span-rows
have ... ⟷ span (rows ?M) = UNIV by auto
finally have ¬ proj2-set-Col S ⟷ span (rows ?M) = UNIV .

have rows ?M = {?u, ?v, ?w}
proof
  { fix x
    assume x ∈ rows ?M
    then obtain i :: 3 where x = ?M $ i
      unfolding rows-def and row-def
      by (auto simp add: vec-lambda-beta vec-lambda-eta)
    with exhaust-3 have x = ?u ∨ x = ?v ∨ x = ?w
      unfolding vector-def
      by auto
    hence x ∈ {?u, ?v, ?w} by simp }
  thus rows ?M ⊆ {?u, ?v, ?w} ..
  { fix x
    assume x ∈ {?u, ?v, ?w}
    hence x = ?u ∨ x = ?v ∨ x = ?w by simp
    hence x = ?M $ 1 ∨ x = ?M $ 2 ∨ x = ?M $ 3
      unfolding vector-def
      by simp
    hence x ∈ rows ?M
      unfolding rows-def row-def vec-lambda-eta
      by blast }
  thus {?u, ?v, ?w} ⊆ rows ?M ..
qed

lemma proj2-no-3-Col-span:
  assumes proj2-no-3-Col S and p ∈ S
  shows span (proj2-rep ' S − {p}) = UNIV
proof −
  from proj2-no-3-Col S have card S = 4 unfolding proj2-no-3-Col-def ..
  with (p ∈ S) and (card S = 4) and card-gt-0-diff-singleton [of S p]
  have card (S − {p}) = 3 by simp

  from proj2-no-3-Col S and (p ∈ S)
  have − proj2-set-Col (S − {p})
    unfolding proj2-no-3-Col-def
    by simp
  with (card (S − {p}) = 3) and not-proj2-set-Col-iff-span
  show span (proj2-rep ' (S − {p})) = UNIV by simp
qed

lemma fourth-proj2-no-3-Col:
  assumes − proj2-Col p q r
  shows ∃ s. proj2-no-3-Col {s,r,p,q}
proof

from (¬ proj2-Col p q r; and proj2-Col-coincide have p ≠ q by auto
hence card {p,q} = 2 by simp

from (¬ proj2-Col p q r; and proj2-Col-coincide and proj2-Col-permute
have r ∉ {p,q} by fast
with (card {p,q} = 2; have card {r,p,q} = 3 by simp

have finite {r,p,q} by simp

let ?s = proj2-abs (∑ t∈{r,p,q}. proj2-rep t)
have ∃ j. (∑ t∈{r,p,q}. proj2-rep t) = j *R proj2-rep ?s
proof cases
  assume (∑ t∈{r,p,q}. proj2-rep t) = 0
  hence (∑ t∈{r,p,q}. proj2-rep t) = 0 *R proj2-rep ?s by simp
  thus ∃ j. (∑ t∈{r,p,q}. proj2-rep t) = j *R proj2-rep ?s ..
next
  assume (∑ t∈{r,p,q}. proj2-rep t) ≠ 0
  with proj2-rep-abs2
  obtain k where k ≠ 0
    and proj2-rep ?s = k *R (∑ t∈{r,p,q}. proj2-rep t)
    by auto
  hence (1/k) *R proj2-rep ?s = (∑ t∈{r,p,q}. proj2-rep t) by simp
  from this [symmetric]
  show ∃ j. (∑ t∈{r,p,q}. proj2-rep t) = j *R proj2-rep ?s ..
qed

then obtain j where (∑ t∈{r,p,q}. proj2-rep t) = j *R proj2-rep ?s ..
let ?c = λ t. if t = ?s then 1 - j else 1
from (p ≠ q) have ?c p ≠ 0 ∨ ?c q ≠ 0 by simp

let ?d = λ t. if t = ?s then j else -1

let ?S = {?s,r,p,q}

have ?s ∉ {r,p,q}

proof
  assume ?s ∈ {r,p,q}

from (r ∉ {p,q}; and (p ∉ q)
have ∀ c r *R proj2-rep r + ∀ c p *R proj2-rep p + ∀ c q *R proj2-rep q
= (∑ t∈{r,p,q}. ∀ c t *R proj2-rep t)
  by (simp add: sum.insert [of - λ t. t *R proj2-rep t])
also from finite {r,p,q} and (?s ∈ {r,p,q})
have . . . = ∀ c ?s *R proj2-rep ?s + (∑ t∈{r,p,q} - {?s}. ∀ c t *R proj2-rep t)
  by (simp only: sum.remove [of {r,p,q} ?s λ t. ?c t *R proj2-rep t])
also have . . . = ¬ j *R proj2-rep ?s + (proj2-rep ?s + (∑ t∈{r,p,q} - {?s}. proj2-rep t))
  by (simp add: algebra-simps)
also from \{finite \{r,p,q\}\} and \{?s \in \{r,p,q\}\}

have \ldots = \neg \sum_{r \in \{r,p,q\}} \proj2-rep \ ?s + \sum_{t \in \{r,p,q\}} \proj2-rep \ t

by (simp only:
  sum.remove [of \{r,p,q\} \ ?s \lambda t. \proj2-rep \ t, symmetric])
also from \{\sum_{t \in \{r,p,q\}} \proj2-rep \ t\} = \neg \sum_{r \in \{r,p,q\}} \proj2-rep \ ?s

have \ldots = 0 \ by \ simp
finally
have \ ?c \ r \ \proj2-rep \ r + \ ?c \ p \ \proj2-rep \ p + \ ?c \ q \ \proj2-rep \ q = 0

. with \{?c p \neq 0 \lor \ ?c q \neq 0\}

have \proj2-Col \ p \ q \ r
  by (unfold \proj2-Col-def) (auto simp add: algebra-simps)

with \(\neg \ \proj2-Col \ p \ q \ r\) show False ..

qed
with \(\card \{r,p,q\} = 3\) have \card \ ?S = 4 \ by \ simp

from \(\neg \ \proj2-Col \ p \ q \ r\) and \proj2-Col-permute

have \(\neg \ \proj2-Col \ p \ q \ r\) \ by \ fast

hence \(\neg \ \proj2-set-Col \ \{r,p,q\}\) by (subst \proj2-Col iff-set-Col [symmetric])

have \(\forall \ u \in \ ?S, \ \neg \ \proj2-set-Col \ (\ ?S - \{u\})\)
proof
Fix \ u
assume \ u \in \ ?S
with \(\card \ ?S = 4\) have \(\card \ (\ ?S - \{u\}) = 3\) \ by \ simp

show \(\neg \ \proj2-set-Col \ (\ ?S - \{u\})\)
proof cases
assume \ u = \ ?s
with \(\ ?s \notin \{r,p,q\}\) have \(\ ?S - \{u\} = \{r,p,q\}\) \ by \ simp

with \(\neg \ \proj2-set-Col \ \{r,p,q\}\) \ show \(\neg \ \proj2-set-Col \ (\ ?S - \{u\})\) \ by \ simp

next
assume \ u \neq \ ?s
hence \ insert \ ?s \ ((\?r,p,q\} - \{u\}) = \ ?S - \{u\} \ by \ auto

from \(\finite \{r,p,q\}\) have \ finite \((\{r,p,q\} - \{u\})\) \ by \ simp

from \(\ ?s \notin \{r,p,q\}\) have \(\ ?s \notin \{r,p,q\} - \{u\}\) \ by \ simp

hence \(\forall \ t \in \{r,p,q\} - \{u\}. \ ?d \ t = -1\) \ by \ auto

from \(\ u \neq \ ?s\) \ and \(\ u \in \ ?S\) \ have \(\ u \in \{r,p,q\}\) \ by \ simp

hence \(\sum_{t \in \{r,p,q\}} \proj2-rep \ t\)
  \begin{align*}
  & = \proj2-rep \ u + (\sum_{t \in \{r,p,q\} - \{u\}} \proj2-rep \ t) \\
  & \ by \ (simp \ add: \ sum.remove)
  \end{align*}

with \(\sum_{t \in \{r,p,q\}} \proj2-rep \ t\) \(= \proj2-rep \ ?s\)

have \(\proj2-rep \ u\)
  \begin{align*}
  & = \proj2-rep \ ?s - (\sum_{t \in \{r,p,q\} - \{u\}} \proj2-rep \ t) \\
  & \ by \ simp
  \end{align*}

also from \(\forall \ t \in \{r,p,q\} - \{u\}. \ ?d \ t = -1\)

have \ldots = \proj2-rep \ ?s + (\sum_{t \in \{r,p,q\} - \{u\}} \proj2-rep \ t)

\[65\]
by (simp add: sum-negf)
also from \(\{r.\ p.\ q\} - \{u\}\) and \(?s \notin \{r.\ p.\ q\} - \{u\}\)

have \(\ldots = (\sum t \in \text{insert} \ ?s (\{r.\ p.\ q\} - \{u\}). \ ?d \ t \in R \text{proj2-rep \ t})\)
by (simp add: sum.insert)
also from \(\text{insert} \ ?s (\{r.\ p.\ q\} - \{u\}) = \{S - \{u\}\}\)

have \(\ldots = (\sum t \in ?S - \{u\}. \ ?d \ t \in R \text{proj2-rep \ t})\) by simp
finally have \(\text{proj2-rep \ u} = (\sum t \in ?S - \{u\}. \ ?d \ t \in R \text{proj2-rep \ t})\).

moreover

have \(\forall t \in ?S - \{u\}. \ ?d \ t \in \text{proj2-rep \ t} \in \text{span} (\text{proj2-rep' \ (?S - \{u\})})\)
by (simp add: \text{span-clauses})
ultimately have \(\text{proj2-rep \ u} \in \text{span} (\text{proj2-rep' \ (?S - \{u\})})\)
by (metis \text{no-types, lifting} \text{span-sum})

have \(\forall t \in \{r.\ p.\ q\}. \ \text{proj2-rep \ t} \in \text{span} (\text{proj2-rep' \ (?S - \{u\})})\)

proof

fix \(t\)

assume \(t \in \{r.\ p.\ q\}\)

show \(\text{proj2-rep \ t} \in \text{span} (\text{proj2-rep' \ (?S - \{u\})})\)

proof cases

assume \(t = u\)

from \(\text{proj2-rep \ u} \in \text{span} (\text{image} \text{proj2-rep' \ (?S - \{u\})})\)

show \(\text{proj2-rep \ t} \in \text{span} (\text{proj2-rep' \ (?S - \{u\})})\)

by (subst \(t = u\))

next

assume \(t \neq u\)

with \(t \in \{r.\ p.\ q\}\)

have \(\text{proj2-rep \ t} \in \text{proj2-rep' \ (?S - \{u\})}\) by simp

with \(\text{span-superset \ [of proj2-rep' \ (?S - \{u\})]}\)

show \(\text{proj2-rep \ t} \in \text{span} (\text{proj2-rep' \ (?S - \{u\})})\) by fast

qed

qed

hence \(\text{proj2-rep' \ {r.\ p.\ q\} \subseteq \text{span} (\text{proj2-rep' \ (?S - \{u\})})}\)

by (simp only: \text{image-subset-iff})

hence

\(\text{span} (\text{proj2-rep' \ {r.\ p.\ q\}}) \subseteq \text{span} (\text{span} (\text{proj2-rep' \ (?S - \{u\}))))\)

by (simp only: \text{span-mono})

hence \(\text{span} (\text{proj2-rep' \ {r.\ p.\ q\}}) \subseteq \text{span} (\text{proj2-rep' \ (?S - \{u\}))))\)

by (simp only: \text{span-span})

moreover

from \(\neg \text{proj2-set-Col} \{r.\ p.\ q\}\)

and \(\text{card} \{r.\ p.\ q\} = 3\)

and \(\text{not-proj2-set-Col-iff-span} \)

have \(\text{span} (\text{proj2-rep' \ {r.\ p.\ q\}}) = \text{UNIV} \) by simp

ultimately have \(\text{span} (\text{proj2-rep' \ (?S - \{u\}))) = \text{UNIV} \) by auto

with \(\text{card} \{?S - \{u\}) = 3\) and \(\text{not-proj2-set-Col-iff-span} \)

show \(\neg \text{proj2-set-Col} \{?S - \{u\}) \) by simp

qed

qed

with \(\text{card} \{?S = 4\})
have proj2-no-3-Col ?S by (unfold proj2-no-3-Col-def) fast
thus ∃ s. proj2-no-3-Col {s,r,p,q} ..
qed

lemma proj2-set-Col-expand:
assumes proj2-set-Col S and {p,q,r} ⊆ S and p ≠ q and r ≠ p
shows ∃ k. r = proj2-abs (k *R proj2-rep p + proj2-rep q)
proof –
from ⟨proj2-set-Col S⟩ obtain l where ∀ t ∈ S. proj2-incident t l unfolding proj2-set-Col-def ..
with ⟨{p,q,r} ⊆ S⟩ and ⟨p ≠ q⟩ and ⟨r ≠ p⟩ and proj2-incident-iff [of p q l r]
show ∃ k. r = proj2-abs (k *R proj2-rep p + proj2-rep q) by simp
qed

7.4 Collineations of the real projective plane

typedef cltn2 :: (Collect invertible :: (realˆ3ˆ3) set)//invertible-proportionality
proof
from matrix-id-invertible have (mat 1 :: realˆ3ˆ3) ∈ Collect invertible
  by simp
thus invertible-proportionality “ {mat 1} ∈ (Collect invertible :: (realˆ3ˆ3) set)//invertible-proportionality
  unfolding quotient-def
  by auto
qed

definition cltn2-rep :: cltn2 ⇒ realˆ3ˆ3 where
  cltn2-rep A ≜ ϵ B. B ∈ Rep-cltn2 A

definition cltn2-abs :: realˆ3ˆ3 ⇒ cltn2 where
  cltn2-abs B ≜ Abs-cltn2 (invertible-proportionality “ {B})

definition cltn2-independent :: cltn2 set ⇒ bool where
  cltn2-independent X ≜ independent {cltn2-rep A | A. A ∈ X}

definition apply-cltn2 :: proj2 ⇒ cltn2 ⇒ proj2 where
  apply-cltn2 x A ≜ proj2-abs (proj2-rep x v* cltn2-rep A)

lemma cltn2-rep-in: cltn2-rep B ∈ Rep-cltn2 B
proof –
let ?A = cltn2-rep B
from quotient-element-nonempty and
  invertible-proportionality-equiv and
  Rep-cltn2 [of B]
have ∃ C. C ∈ Rep-cltn2 B
  by auto
with someI-ex [of λ C. C ∈ Rep-cltn2 B]
show ?A ∈ Rep-cltn2 B
lemma cltn2-rep-invertible: invertible (cltn2-rep A)
proof -
  from
      Union-quotient [of Collect invertible invertible-proportionality]
      and invertible-proportionality-equiv
      and Rep-cltn2 [of A] and cltn2-rep-in [of A]
have cltn2-rep A ∈ Collect invertible
    unfolding quotient-def
    by auto
thus invertible (cltn2-rep A)
  unfolding invertible-proportionality-def
  by simp
qed

lemma cltn2-rep-abs:
  fixes A :: real\(^3\times3\)
  assumes invertible A
  shows (A, cltn2-rep (cltn2-abs A)) ∈ invertible-proportionality
proof -
  from (invertible A)
have invertible-proportionality "\{A\} ∈ (Collect invertible :: (real\(^3\times3\) set)//invertible-proportionality
    unfolding quotient-def
    by auto
with Abs-cltn2-inverse
have Rep-cltn2 (cltn2-abs A) = invertible-proportionality "\{A\}
  unfolding cltn2-abs-def
  by simp
with cltn2-rep-in
have cltn2-rep (cltn2-abs A) ∈ invertible-proportionality "\{A\} by auto
thus (A, cltn2-rep (cltn2-abs A)) ∈ invertible-proportionality by simp
qed

lemma cltn2-rep-abs2:
  assumes invertible A
  shows \(\exists k. k \neq 0 \land cltn2-rep (cltn2-abs A) = k \cdot R A\)
proof -
  from (invertible A) and cltn2-rep-abs
  have (A, cltn2-rep (cltn2-abs A)) ∈ invertible-proportionality by simp
  then obtain c where A = c \cdot R cltn2-rep (cltn2-abs A)
    unfolding invertible-proportionality-def and real-vector.proportionality-def
    by auto
with (invertible A) and zero-not-invertible have c \neq 0 by auto
  hence \(1/c \neq 0\) by simp

let \(?k = 1/c\)
from \( (A = c \cdot_R \text{cltn2-rep} \text{(cltn2-abs } A)) \)

have \( ?k \cdot_R A = ?k \cdot_R c \cdot_R \text{cltn2-rep} \text{(cltn2-abs } A) \) by simp

with \((c \neq 0) \) have \text{cltn2-rep} \text{(cltn2-abs } A) = ?k \cdot_R A \) by simp

with \((?k \neq 0) \)

show \( \exists \ k. \ k \neq 0 \wedge \text{cltn2-rep} \text{(cltn2-abs } A) = k \cdot_R A \) by blast

qed

lemma \( \text{cltn2-abs-rep} \): \( \text{cltn2-abs} \text{(cltn2-rep } A) = A \)

proof –

from \( \text{partition-Image-element} \)

[of \( \text{Collect invertible} \)

\( \text{Rep-cltn2 } A \)

\( \text{cltn2-rep } A \)]

and \( \text{invertible-proportionality-equiv} \)

and \( \text{Rep-cltn2} \text{ [of } A \text{] and cltn2-rep-in [of } A \text{]} \)

have \( \text{invertible-proportionality} \{ \text{cltn2-rep } A \} = \text{Rep-cltn2 } A \)

by simp

with \( \text{Rep-cltn2-inverse} \)

show \( \text{cltn2-abs} \text{(cltn2-rep } A) = A \)

unfolding \( \text{cltn2-abs-def} \)

by simp

qed

lemma \( \text{cltn2-abs-mult} \):

assumes \( k \neq 0 \) and \( \text{invertible } A \)

shows \( \text{cltn2-abs} \text{(k } \cdot_R A) = \text{cltn2-abs } A \)

proof –

from \( (k \neq 0) \cdot (\text{invertible } A) \) and \( \text{scalar-invertible} \)

have \( \text{invertible} \text{(k } \cdot_R A) \) by auto

with \( \text{invertible } A \)

have \( (k \cdot_R A, A) \in \text{invertible-proportionality} \)

unfolding \( \text{invertible-proportionality-def} \)

and \( \text{real-vector.proportionality-def} \)

by \( (\text{auto simp add: zero-not-invertible}) \)

with \( \text{eq-equiv-class-iff} \)

[of \( \text{Collect invertible invertible-proportionality k } \cdot_R A \text{ } A \)]

and \( \text{invertible-proportionality-equiv} \)

and \( \text{invertible } A \) and \( \text{invertible} \text{(k } \cdot_R A) \)

have \( \text{invertible-proportionality} \{ k \cdot_R A \}

= \text{invertible-proportionality} \{ A \} \)

by simp

thus \( \text{cltn2-abs} \text{(k } \cdot_R A) = \text{cltn2-abs } A \)

unfolding \( \text{cltn2-abs-def} \)

by simp

qed

lemma \( \text{cltn2-abs-mult-rep} \):

assumes \( k \neq 0 \)
shows cltn2-abs \((k \cdot_R \text{cltn2-rep } A) = A\)
using cltn2-rep-invertible and cltn2-abs-mult and cltn2-abs-rep and assms
by simp

lemma apply-cltn2-abs:
  assumes \(x \neq 0\) and invertible \(A\)
  shows apply-cltn2 \((\text{proj2-abs } x) \ (\text{cltn2-abs } A) = \text{proj2-abs } (x \cdot v \cdot A)\)
proof –
  from proj2-rep-abs2 and \(x \neq 0\)
  obtain \(k\) where \(k \neq 0\) and proj2-rep \((\text{proj2-abs } x) = k \cdot_R x\) by auto
from cltn2-rep-abs2 and \(\text{invertible } A\)
obtain \(c\) where \(c \neq 0\) and cltn2-rep \((\text{cltn2-abs } A) = c \cdot_R A\) by auto
from \(k \neq 0\) and \(c \neq 0\) have \(k \cdot c \neq 0\) by simp
from \(\text{proj2-rep } (\text{proj2-abs } x) = k \cdot_R x\) and \(\text{cltn2-rep } \text{cltn2-abs } A = c \cdot_R A\)
  have proj2-rep \((\text{proj2-abs } x) \cdot v \cdot \text{cltn2-rep } \text{cltn2-abs } A = (k \cdot c) \cdot_R (x \cdot v \cdot A)\)
    by (simp add: scaleR-vector-matrix-assoc vector-scaleR-matrix-ac)
with \(k \cdot c \neq 0\)
  show apply-cltn2 \((\text{proj2-abs } x) \text{cltn2-abs } A) = \text{proj2-abs } (x \cdot v \cdot A)\)
  unfolding apply-cltn2-def
    by (simp add: proj2-abs-mult)
qed

lemma apply-cltn2-left-abs:
  assumes \(v \neq 0\)
  shows apply-cltn2 \((\text{proj2-abs } v) \ (\text{cltn2-abs } C) = \text{proj2-abs } (v \cdot v \cdot \text{cltn2-rep } C)\)
proof –
  have cltn2-abs \((\text{cltn2-rep } C) = C\) by (rule cltn2-abs-rep)
  with \(v \neq 0\) and cltn2-rep-invertible and apply-cltn2-abs \([of v \text{ cltn2-rep } C]\)
  show apply-cltn2 \((\text{proj2-abs } v) \ (\text{cltn2-abs } C) = \text{proj2-abs } (v \cdot v \cdot \text{cltn2-rep } C)\)
    by simp
qed

lemma apply-cltn2-right-abs:
  assumes invertible \(M\)
  shows apply-cltn2 \(p \ (\text{cltn2-abs } M) = \text{proj2-abs } (\text{proj2-rep } p \cdot v \cdot M)\)
proof –
  from proj2-rep-non-zero and \(\text{invertible } M\) and apply-cltn2-abs
  have apply-cltn2 \((\text{proj2-abs } (\text{proj2-rep } p)) \ (\text{cltn2-abs } M) = \text{proj2-abs } (\text{proj2-rep } p \cdot v \cdot M)\)
    by simp
  thus apply-cltn2 \(p \ (\text{cltn2-abs } M) = \text{proj2-abs } (\text{proj2-rep } p \cdot v \cdot M)\)
    by (simp add: proj2-abs-rep)
qed

lemma non-zero-mult-rep-non-zero:
  assumes \(v \neq 0\)
shows $v \times c l t n ^ 2 - r e p \ C \neq 0$
using $(v \neq 0)$ and $c l t n ^ 2 - r e p$-invertible and times-invertible-eq-zero
by auto

lemma rep-mult-rep-non-zero: proj2-rep $p \times v \times c l t n ^ 2 - r e p \ A \neq 0$
using proj2-rep-non-zero
by (rule non-zero-mult-rep-non-zero)

definition cltn2-image :: proj2 set $\Rightarrow$ cltn2 $\Rightarrow$ proj2 set
where
$\text{cltn2-image } P \ A \triangleq \{ \text{apply-cltn2 } p \ A \mid p. p \in P \}$

7.4.1 As a group

definition cltn2-id :: cltn2 where
$\text{cltn2-id } \triangleq \text{cltn2-abs } (\text{mat } 1 )$

definition cltn2-compose :: cltn2 $\Rightarrow$ cltn2 $\Rightarrow$ cltn2 where
$\text{cltn2-compose } A \ B \triangleq \text{cltn2-abs } (\text{cltn2-rep } A \star \star \text{cltn2-rep } B )$

definition cltn2-inverse :: cltn2 $\Rightarrow$ cltn2 where
$\text{cltn2-inverse } A \triangleq \text{cltn2-abs } (\text{matrix-inv } (\text{cltn2-rep } A ) )$

lemma cltn2-compose-abs:
assumes invertible $M$ and invertible $N$
shows $\text{cltn2-compose } (\text{cltn2-abs } M ) \ (\text{cltn2-abs } N ) = \text{cltn2-abs } (M \star \star N )$

proof –
from $\langle \text{invertible } M \rangle$ and $\langle \text{invertible } N \rangle$ and invertible-mult
have invertible $(M \star \star N )$ by auto
from $\langle \text{invertible } M \rangle$ and $\langle \text{invertible } N \rangle$ and $\text{cltn2-rep-abs2}$
obtain $j$ and $k$ where $j \neq 0$ and $k \neq 0$
and $\text{cltn2-rep } (\text{cltn2-abs } M ) = j \star _R M$
and $\text{cltn2-rep } (\text{cltn2-abs } N ) = k \star _R N$
by blast
from $\langle j \neq 0 \rangle$ and $\langle k \neq 0 \rangle$ have $j \times k \neq 0$ by simp
from $\langle \text{cltn2-rep } (\text{cltn2-abs } M ) = j \star _R M \rangle$ and $\langle \text{cltn2-rep } (\text{cltn2-abs } N ) = k \star _R N \rangle$
have $\text{cltn2-rep } (\text{cltn2-abs } M ) \star \star \text{cltn2-rep } (\text{cltn2-abs } N )$
$= (j \times k) \star _R (M \star \star N )$
by (simp add: matrix-scalar-ac scalar-matrix-assoc [symmetric])
with $\langle j \times k \neq 0 \rangle$ and invertible $(M \star \star N )$:
show $\text{cltn2-compose } (\text{cltn2-abs } M ) \ (\text{cltn2-abs } N ) = \text{cltn2-abs } (M \star \star N )$
unfolding $\text{cltn2-compose-def}$
by (simp add: cltn2-abs-mult)
qed

lemma cltn2-compose-left-abs:
assumes invertible \( M \)
shows \( \text{cltn2-compose} (\text{cltn2-abs} M) \; A = \text{cltn2-abs} (M \; \ast \ast \; \text{cltn2-rep} \; A) \)
proof
−
from (invertible \( M \)) and \( \text{cltn2-rep-invertible} \) and \( \text{cltn2-compose-abs} \)
have \( \text{cltn2-compose} (\text{cltn2-abs} (\text{cltn2-abs} M)) (\text{cltn2-abs} (\text{cltn2-rep} A)) = \text{cltn2-abs} (M \; \ast \ast \; \text{cltn2-rep} \; A) \)
by simp
thus \( \text{cltn2-compose} (\text{cltn2-abs} M) \; A = \text{cltn2-abs} (M \; \ast \ast \; \text{cltn2-rep} \; A) \)
by (simp add: \text{cltn2-abs-rep})
qed

lemma \( \text{cltn2-compose-right-abs} \):
assumes invertible \( M \)
shows \( \text{cltn2-compose} A (\text{cltn2-abs} M) = \text{cltn2-abs} (\text{cltn2-rep} A \; \ast \ast \; M) \)
proof
−
from (invertible \( M \)) and \( \text{cltn2-rep-invertible} \) and \( \text{cltn2-compose-abs} \)
have \( \text{cltn2-compose} (\text{cltn2-abs} (\text{cltn2-rep} A)) (\text{cltn2-abs} M) = \text{cltn2-abs} (\text{cltn2-rep} A \; \ast \ast \; M) \)
by simp
thus \( \text{cltn2-compose} A (\text{cltn2-abs} M) = \text{cltn2-abs} (\text{cltn2-rep} A \; \ast \ast \; M) \)
by (simp add: \text{cltn2-abs-rep})
qed

lemma \( \text{cltn2-abs-rep-abs-mult} \):
assumes invertible \( M \) and invertible \( N \)
shows \( \text{cltn2-abs} (\text{cltn2-rep} (\text{cltn2-abs} M) \; \ast \ast \; N) = \text{cltn2-abs} (M \; \ast \ast \; N) \)
proof
−
from (invertible \( M \)) and (invertible \( N \))
have invertible \( (M \; \ast \ast \; N) \) by (simp add: invertible-mult)
from (invertible \( M \)) and \( \text{cltn2-rep-abs2} \)
obtain \( k \) where \( k \neq 0 \) and \( \text{cltn2-rep} (\text{cltn2-abs} M) = k \; *R \; M \) by auto
from \( \text{cltn2-rep} (\text{cltn2-abs} M) = k \; *R \; M \)
have \( \text{cltn2-abs} (\text{cltn2-rep} (\text{cltn2-abs} M) \; \ast \ast \; N) = k \; *R \; M \; \ast \ast \; N \) by simp
with \( (k \neq 0) \) and (invertible \( (M \; \ast \ast \; N) \)) and \( \text{cltn2-abs-mult} \)
show \( \text{cltn2-abs} (\text{cltn2-rep} (\text{cltn2-abs} M) \; \ast \ast \; N) = \text{cltn2-abs} (M \; \ast \ast \; N) \)
by (simp add: scalar-matrix-assoc [symmetric])
qed

lemma \( \text{cltn2-assoc} \):
\( \text{cltn2-compose} (\text{cltn2-compose} A \; B) \; C = \text{cltn2-compose} A (\text{cltn2-compose} B \; C) \)
proof
−
let \( ?A' = \text{cltn2-rep} \; A \)
let \( ?B' = \text{cltn2-rep} \; B \)
let \( ?C' = \text{cltn2-rep} \; C \)
from \( \text{cltn2-rep-invertible} \)
have invertible \( ?A' \) and invertible \( ?B' \) and invertible \( ?C' \) by simp-all
with invertible-mult
have invertible \( (?A' \; \ast \ast \; ?B') \) and invertible \( (?B' \; \ast \ast \; ?C') \)

72
and invertible \((?A' ** ?B' ** ?C')\)
by auto

from (invertible \((?A' ** ?B')\)) and \((?C')\) and cltn2-abs-rep-abs-mult
have cltn2-abs (cltn2-rep (cltn2-abs \((?A' ** ?B')\)) ** \((?C')\))
  = cltn2-abs \((?A' ** ?B' ** ?C')\)
  by simp

from (invertible \((?B' ** ?C')\)) and cltn2-rep-abs2 [of \((?B' ** ?C')\)]
obtain \(k\) where \(k \neq 0\)
  and cltn2-rep (cltn2-abs \((?B' ** ?C')\)) = \(k \cdot_R \((?B' ** ?C')\)\)
  by auto

from (cltn2-rep (cltn2-abs \((?B' ** ?C')\)) = \(k \cdot_R \((?B' ** ?C')\)\))
and cltn2-rep-abs-mult [of \((?B' ** ?C')\)]
have \((?A' ** cltn2-rep (cltn2-abs \((?B' ** ?C')\))) = cltn2-abs \((?A' ** ?B' ** ?C')\)
  by (simp add: matrix-scalar-ac matrix-mul-assoc scalar-mul-assoc)

show
  cltn2-compose (cltn2-compose A B) C = cltn2-compose A (cltn2-compose B C)
  unfolding cltn2-compose-def
  by simp

qed

lemma cltn2-left-id: cltn2-compose cltn2-id A = A
proof
  let \(?A' = cltn2-rep A\)
  from cltn2-rep-invertible have invertible \(?A'\) by simp
  with matrix-id-invertible and cltn2-abs-rep-abs-mult [of mat \(?A'\)]
  have cltn2-compose cltn2-id A = cltn2-abs (cltn2-rep A)
  unfolding cltn2-compose-def and cltn2-id-def
  by (auto simp add: matrix-mul-lid)

  with cltn2-abs-rep show cltn2-compose cltn2-id A = A by simp

qed

lemma cltn2-left-inverse: cltn2-compose (cltn2-inverse A) A = cltn2-id
proof
  let \(?M = cltn2-rep A\)
  let \(?M' = matrix-inv \(?M\)\)
  from cltn2-rep-invertible have invertible \(?M\) by simp
  with matrix-inv-invertible have invertible \(?M'\) by auto
  with \((\text{invertible } \(?M\))\) and cltn2-abs-rep-abs-mult
  have cltn2-compose (cltn2-inverse A) A = cltn2-abs \((?M' ** \(?M\))\)
  unfolding cltn2-compose-def and cltn2-inverse-def
  by simp
  with \((\text{invertible } \(?M\))\).
show \(cltn2-compose (cltn2-inverse A) A = cltn2-id\)

unfolding \(cltn2-id-def\)

by (simp add: matrix-inv)

qed

lemma \(cltn2-left-inverse-ex\):

\(\exists B. cltn2-compose B A = cltn2-id\)

using \(cltn2-left-inverse \ldots\)

interpretation \(cltn2\):

\(\text{group (carrier = UNIV, mult = cltn2-compose, one = cltn2-id)}\)

using \(cltn2-assoc \text{ and } cltn2-left-id \text{ and } cltn2-left-inverse-ex\)

and \(\text{groupI [of (carrier = UNIV, mult = cltn2-compose, one = cltn2-id)]}\)

by simp-all

lemma \(cltn2-inverse-inv \text{ simp} [simp]\):

\(\text{inv[(carrier = UNIV, mult = cltn2-compose, one = cltn2-id)] A = cltn2-inverse A}\)

using \(cltn2-left-inverse [of A] \text{ and cltn2-inv-equality}\)

by simp

lemmas \(cltn2-inverse-id [simp] = cltn2.inv-one [simplified]\)

and \(cltn2-inverse-compose = cltn2.inv-mult-group [simplified]\)

7.4.2 As a group action

lemma \(apply-cltn2-id \text{ simp} [simp]: apply-cltn2 p cltn2-id = p\)

proof –

from \(\text{matrix-id-invertible and apply-cltn2-right-abs}\)

have \(apply-cltn2 p cltn2-id = proj2-abs (proj2-rep p v* mat1)\)

unfolding \(cltn2-id-def\) by blast

thus \(apply-cltn2 p cltn2-id = p\)

by (simp add: proj2-abs-rep)

qed

lemma \(apply-cltn2-compose\):

\(apply-cltn2 (apply-cltn2 p A) B = apply-cltn2 p (cltn2-compose A B)\)

proof –

from \(\text{rep-mult-rep-non-zero and cltn2-rep-invertible and apply-cltn2-abs}\)

have \(apply-cltn2 (apply-cltn2 p A) (cltn2-abs (cltn2-rep B)) = proj2-abs ((proj2-rep p v* cltn2-rep A) v* cltn2-rep B)\)

unfolding \(apply-cltn2-def [of p A]\)

by simp

hence \(apply-cltn2 (apply-cltn2 p A) B\)

= \(proj2-abs (proj2-rep p v* (cltn2-rep A ** cltn2-rep B))\)

by (simp add: cltn2-abs-rep vector-matrix-mul-assoc)

from \(cltn2-rep-invertible \text{ and invertible-mult}\)

have \(\text{invertible (cltn2-rep A ** cltn2-rep B)} \text{ by auto}\)
with apply-cltn2-right-abs
have apply-cltn2 p (cltn2-compose A B) = proj2-abs (proj2-rep p v* (cltn2-rep A ** cltn2-rep B))
  unfolding cltn2-compose-def
  by simp
with ⟨apply-cltn2 (apply-cltn2 p A) B⟩
show apply-cltn2 (apply-cltn2 p A) B = apply-cltn2 p (cltn2-compose A B)
  by simp
qed

interpretation cltn2:
action (carrier = UNIV, mult = cltn2-compose, one = cltn2-id) apply-cltn2
proof
let ?G = (carrier = UNIV, mult = cltn2-compose, one = cltn2-id)
fix p
show apply-cltn2 p 1 ?G = p by simp
fix A B
have apply-cltn2 (apply-cltn2 p A) B = apply-cltn2 p (A ⊗ ?G B)
  by simp (rule apply-cltn2-compose)
thus A ∈ carrier ?G ∧ B ∈ carrier ?G
  → apply-cltn2 (apply-cltn2 p A) B = apply-cltn2 p (A ⊗ ?G B)
.. qed

definition cltn2-transpose :: cltn2 ⇒ cltn2 where
cltn2-transpose A ≜ cltn2-abs (transpose (cltn2-rep A))

definition apply-cltn2-line :: proj2-line ⇒ cltn2 ⇒ proj2-line where
apply-cltn2-line l A
  ≜ P2L (apply-cltn2 (L2P l) (cltn2-inverse cltn2-rep (cltn2-inverse A)))

lemma cltn2-transpose-abs:
assumes invertible M
shows cltn2-transpose (cltn2-abs M) = cltn2-abs (transpose M)
proof —
  from ⟨invertible M and transpose-invertible⟩ have invertible (transpose M) by auto

  from ⟨invertible M and cltn2-2-abs⟩ obtain k where k ≠ 0 and cltn2-2-rep (cltn2-2-abs M) = k *R M by auto

  from ⟨cltn2-2-rep (cltn2-2-abs M) = k *R M⟩
  have transpose (cltn2-2-rep (cltn2-2-abs M)) = k *R transpose M
    by (simp add: transpose-scalar)
  with ⟨k ≠ 0 and invertible (transpose M)⟩
  show cltn2-transpose (cltn2-2-abs M) = cltn2-abs (transpose M)
    unfolding cltn2-transpose-def
    by (simp add: cltn2-2-abs-mult)
\textbf{qed}

\begin{verbatim}
lemma cltn2-transpose-compose:
  cltn2-transpose (cltn2-compose A B) = cltn2-compose (cltn2-transpose B) (cltn2-transpose A)
proof –
  from cltn2-rep-invertible have invertible (cltn2-rep A) \textbf{and} invertible (cltn2-rep B)
    by simp-all
  with transpose-invertible
  have invertible (transpose (cltn2-rep A))
    \textbf{and} invertible (transpose (cltn2-rep B))
    by auto
  from (invertible (cltn2-rep A)) \textbf{and} (invertible (cltn2-rep B))
    \textbf{and} invertible-mult
  have cltn2-transpose (cltn2-compose A B)
    = cltn2-abs (transpose (cltn2-rep A ** cltn2-rep B))
    unfolding cltn2-compose-def
    by simp
  also have \ldots = cltn2-abs (transpose (cltn2-rep B) ** transpose (cltn2-rep A))
    by (simp add: matrix-transpose-mul)
  also from (invertible (transpose (cltn2-rep B)))
    \textbf{and} (invertible (transpose (cltn2-rep A)))
    \textbf{and} cltn2-compose-abs
  have \ldots = cltn2-compose (cltn2-transpose B) (cltn2-transpose A)
    unfolding cltn2-compose-def
    by simp
  finally show cltn2-transpose (cltn2-compose A B) = cltn2-compose (cltn2-transpose B) (cltn2-transpose A).
\end{verbatim}

\textbf{qed}

\begin{verbatim}
lemma cltn2-transpose-transpose:
  cltn2-transpose (cltn2-transpose A) = A
proof –
  from cltn2-rep-invertible have invertible (cltn2-rep A) \textbf{by} simp
  with transpose-invertible have invertible (transpose (cltn2-rep A)) \textbf{by} auto
  with cltn2-compose-abs [of transpose (cltn2-rep A)]
  have cltn2-transpose (cltn2-transpose A) = cltn2-abs (transpose (transpose (cltn2-rep A)))
    unfolding cltn2-transpose-def [of A]
    by simp
  with cltn2-abs-rep and transpose-transpose [of cltn2-rep A]
  show cltn2-transpose (cltn2-transpose A) = A \textbf{by} simp
\end{verbatim}

\textbf{qed}

\begin{verbatim}
lemma cltn2-transpose-id [simp];
  cltn2-transpose cltn2-id = cltn2-id
\end{verbatim}
using cltn2-transpose-abs
unfolding cltn2-id-def
by (simp add: transpose-mat matrix-id-invertible)

lemma apply-cltn2-line-id [simp]: apply-cltn2-line l cltn2-id = l
unfolding apply-cltn2-line-def
by simp

lemma apply-cltn2-line-compose:
apply-cltn2-line (apply-cltn2-line l A) B
= apply-cltn2-line l (cltn2-compose A B)
proof –
have cltn2-compose
(cltn2-transpose (cltn2-inverse A)) (cltn2-transpose (cltn2-inverse B))
= cltn2-transpose (cltn2-inverse (cltn2-compose A B))
by (simp add: cltn2-transpose-compose cltn2-inverse-compose)
thus apply-cltn2-line (apply-cltn2-line l A) B
= apply-cltn2-line l (cltn2-compose A B)
unfolding apply-cltn2-line-def
by (simp add: apply-cltn2-compose)
qed

interpretation cltn2-line:
action
(|carrier = UNIV, mult = cltn2-compose, one = cltn2-id|)
apply-cltn2-line
proof
let ?G = (|carrier = UNIV, mult = cltn2-compose, one = cltn2-id|)
fix l
show apply-cltn2-line l 1 ?G = l by simp
fix A B
have apply-cltn2-line (apply-cltn2-line l A) B
= apply-cltn2-line l (A ⊗ ?G B)
by simp (rule apply-cltn2-line-compose)
thus A ∈ carrier ?G ∧ B ∈ carrier ?G
→ apply-cltn2-line (apply-cltn2-line l A) B
= apply-cltn2-line l (A ⊗ ?G B)
.. 
qed

lemmas apply-cltn2-inv [simp] = cltn2.act-act-inv [simplified]
lemmas apply-cltn2-line-inv [simp] = cltn2-line.act-act-inv [simplified]

lemma apply-cltn2-line-alt-def:
apply-cltn2-line l A
= proj2-line-abs (cltn2-rep (cltn2-inverse A) *v proj2-line-rep l)
proof –
have invertible (cltn2-rep (cltn2-inverse A)) by (rule cltn2-rep-invertible)
hence invertible (transpose (cltn2-rep (cltn2-inverse A)))
by (rule transpose-invertible)

hence

apply-cltn2 (L2P l) (cltn2-transpose (cltn2-inverse A))

= proj2-abs (proj2-rep (L2P l) v* transpose (cltn2-rep (cltn2-inverse A)))

by (rule apply-cltn2-right-abs)

hence apply-cltn2 (L2P l) (cltn2-transpose (cltn2-inverse A))

= proj2-abs (cltn2-rep (cltn2-inverse A) *v proj2-line-rep l)

by simp

thus apply-cltn2-line l A

= proj2-line-abs (cltn2-rep (cltn2-inverse A) *v proj2-line-rep l)

by simp

unfolding apply-cltn2-line-def and proj2-line-abs-def ...

qed

lemma rep-mult-line-rep-non-zero: cltn2-rep A *v proj2-line-rep l ≠ 0

using proj2-line-rep-non-zero and cltn2-rep-invertible

and invertible-times-eq-zero

by auto

lemma apply-cltn2-incident:

proj2-incident p (apply-cltn2-line l A)

⟷ proj2-incident (apply-cltn2 p (cltn2-inverse A)) l

proof

have proj2-rep p v* cltn2-rep (cltn2-inverse A) ≠ 0

by (rule rep-mult-rep-non-zero)

with proj2-rep-obs2

obtain j where j ≠ 0

and proj2-rep (proj2-abs (proj2-rep p v* cltn2-rep (cltn2-inverse A)))

= j *R (proj2-rep p v* cltn2-rep (cltn2-inverse A))

by auto

let ?v = cltn2-rep (cltn2-inverse A) *v proj2-line-rep l

have proj2-line-rep-obs2 [of ?v]

obtain k where k ≠ 0

and proj2-line-rep (proj2-line-rep-obs2 ?v) = k *R ?v

by auto

hence proj2-incident p (apply-cltn2-line l A)

⟷ proj2-rep p * (cltn2-rep (cltn2-inverse A) *v proj2-line-rep l) = 0

by simp

also from dot-lmul-matrix [of proj2-rep p cltn2-rep (cltn2-inverse A)]

have

... ⟷ (proj2-rep p v* cltn2-rep (cltn2-inverse A)) * proj2-line-rep l = 0

by simp

also from j ≠ 0:

and proj2-rep (proj2-abs (proj2-rep p v* cltn2-rep (cltn2-inverse A)))

= j *R (proj2-rep p v* cltn2-rep (cltn2-inverse A))
have ... \iff proj2-incident (apply-cltn2 p (cltn2-inverse A)) l
unfolding proj2-incident-def and apply-cltn2-def
by simp add: dot-scaleR-mult
finally show ?thesis.
qed

lemma apply-cltn2-preserve-incident [iff]:
proj2-incident (apply-cltn2 p A) (apply-cltn2-line l A)
\iff proj2-incident p l
by simp add: apply-cltn2-incident

lemma apply-cltn2-preserve-set-Col:
assumes proj2-set-Col S
shows proj2-set-Col \{ apply-cltn2 p C | p. p \in S \}
proof -
from (proj2-set-Col S)
obtain l where \forall p \in S. proj2-incident p l unfolding proj2-set-Col-def ..
hence \forall q \in \{ apply-cltn2 p C | p. p \in S \}.
proj2-incident q (apply-cltn2-line l C)
by auto
thus proj2-set-Col \{ apply-cltn2 p C | p. p \in S \}
unfolding proj2-set-Col-def ..
qed

lemma apply-cltn2-injective:
assumes apply-cltn2 p C = apply-cltn2 q C
shows p = q
proof -
from (apply-cltn2 p C = apply-cltn2 q C)
have apply-cltn2 (apply-cltn2 p C) (cltn2-inverse C)
  = apply-cltn2 (apply-cltn2 q C) (cltn2-inverse C)
by simp
thus p = q by simp
qed

lemma apply-cltn2-line-injective:
assumes apply-cltn2-line l C = apply-cltn2-line m C
shows l = m
proof -
from (apply-cltn2-line l C = apply-cltn2-line m C)
have apply-cltn2-line (apply-cltn2-line l C) (cltn2-inverse C)
  = apply-cltn2-line (apply-cltn2-line m C) (cltn2-inverse C)
by simp
thus l = m by simp
qed

lemma apply-cltn2-line-unique:
assumes p \neq q and proj2-incident p l and proj2-incident q l
and proj2-incident (apply-cltn2 p C) m

79
and \( \text{proj2-incident} \) \((\text{apply-cltn2} \ q \ C) \) \( m \)
shows \( \text{apply-cltn2-line} \ l \ C = m \)

proof –
from \((\text{proj2-incident} \ p \ l)\)
have \((\text{proj2-incident} \) \((\text{apply-cltn2} \ p \ C) \) \((\text{apply-cltn2-line} \ l \ C)\)) by simp

from \((\text{proj2-incident} \ q \ l)\)
have \((\text{proj2-incident} \) \((\text{apply-cltn2} \ q \ C) \) \((\text{apply-cltn2-line} \ l \ C)\)) by simp

from \((p \neq q) \) and \( \text{apply-cltn2-injective} \) \([\text{of} \ C \ q]\)
have \( \text{apply-cltn2} \ p \ C \neq \text{apply-cltn2} \ q \ C \) by auto
with \((\text{proj2-incident} \) \((\text{apply-cltn2} \ p \ C) \) \((\text{apply-cltn2-line} \ l \ C)\))
and \((\text{proj2-incident} \) \((\text{apply-cltn2} \ q \ C) \) \((\text{apply-cltn2-line} \ l \ C)\))
and \((\text{proj2-incident} \) \((\text{apply-cltn2} \ p \ C) \) \( m)\)
and \((\text{proj2-incident} \) \((\text{apply-cltn2} \ q \ C) \) \( m)\)
and \( \text{proj2-incident-unique} \)
show \( \text{apply-cltn2-line} \ l \ C = m \) by fast
qed

lemma \( \text{apply-cltn2-unique} \):
assumes \((l \neq m) \) and \( \text{proj2-incident} \ p \ l \) and \( \text{proj2-incident} \ p \ m \)
and \( \text{proj2-incident} \ q \) \((\text{apply-cltn2-line} \ l \ C)\)
and \( \text{proj2-incident} \ q \) \((\text{apply-cltn2-line} \ m \ C)\)
shows \( \text{apply-cltn2} \ p \ C = q \)

proof –
from \((\text{proj2-incident} \ p \ l)\)
have \((\text{proj2-incident} \) \((\text{apply-cltn2} \ p \ C) \) \((\text{apply-cltn2-line} \ l \ C)\)) by simp

from \((\text{proj2-incident} \ p \ m)\)
have \((\text{proj2-incident} \) \((\text{apply-cltn2} \ p \ C) \) \((\text{apply-cltn2-line} \ m \ C)\)) by simp

from \((l \neq m) \) and \( \text{apply-cltn2-line-injective} \) \([\text{of} \ C \ m]\)
have \( \text{apply-cltn2-line} \ l \ C \neq \text{apply-cltn2-line} \ m \ C \) by auto
with \((\text{proj2-incident} \) \((\text{apply-cltn2} \ p \ C) \) \((\text{apply-cltn2-line} \ l \ C)\))
and \((\text{proj2-incident} \) \((\text{apply-cltn2} \ p \ C) \) \((\text{apply-cltn2-line} \ m \ C)\))
and \((\text{proj2-incident} \ q \) \((\text{apply-cltn2-line} \ l \ C)\))
and \((\text{proj2-incident} \ q \) \((\text{apply-cltn2-line} \ m \ C)\))
and \( \text{proj2-incident-unique} \)
show \( \text{apply-cltn2} \ p \ C = q \) by fast
qed

7.4.3 Parts of some Statements from [1]

All theorems with names beginning with \textit{statement} are based on corresponding theorems in [1].

lemma \textit{statement52-existence}:
fixes \( a :: \text{proj2} ^{\mathbb{3}} \) and \( a^{\mathbb{3}} :: \text{proj2} \)
assumes \( \text{proj2-no-3-Col} \) \((\text{insert} \ a^{\mathbb{3}} \) \((\text{range} \) \((\$) \) \( a))\))
shows \( \exists \ A. \ \text{apply-cltn2} \) \((\text{proj2-abs} \) \((\text{vector} \) \([1,1,1])\)) \( A = a^{\mathbb{3}} \land \)
(∀ j. apply-cltn2 (proj2-abs (axis j 1))) A = a$j

proof −
let ?v = proj2-rep a3
let ?B = proj2-rep ' range (($) a)

from (proj2-no-3-Col (insert a3 (range (($) a))))
have card (insert a3 (range (($) a))) = 4 unfolding proj2-no-3-Col-def ..

from card-image-le [of UNIV ($) a]
have card (range (($) a)) ≤ 3 by simp
with card-insert-if [of range (($) a) a3]
and (card (insert a3 (range (($) a))) = 4)
have a3 ∉ range (($) a) by auto
hence (insert a3 (range (($) a))) − {a3} = range (($) a) by simp
with (proj2-no-3-Col (insert a3 (range (($) a))))
and proj2-no-3-Col-span [of insert a3 (range (($) a)) a3]
have span ?B = UNIV by simp

from card-suc-ge-insert [of a3 range (($) a)]
and (card (insert a3 (range (($) a))) = 4)
and (card (range (($) a)) ≤ 3)
have card (range (($) a)) = 3 by simp
with card-image [of proj2-rep range (($) a)]
and proj2-rep-inj
and subset-inj-on
have card ?B = 3 by auto
hence finite ?B by simp
with (span ?B = UNIV) and span-finite [of ?B]
obtain c where (∑ w ∈ ?B. (c w) *R w) = ?v
by (auto simp add: scalar-equiv) (metis (no-types, lifting) UNIV-I rangeE)
let ?C = χ i. c (proj2-rep (a$i)) *R (proj2-rep (a$i))
let ?A = cltn2-abs ?C

from proj2-rep-inj and (a3 ∉ range (($) a)) have ?v ∉ ?B
unfolding inj-on-def
by auto

have ∀ i. c (proj2-rep (a$i)) ≠ 0
proof
fix i
let ?Bi = proj2-rep ' (range (($) a) − {a$i})

have a$i ∈ insert a3 (range (($) a)) by simp

have proj2-rep (a$i) ∈ ?B by auto

from image-set-diff [of proj2-rep] and proj2-rep-inj
have ?Bi = ?B − {proj2-rep (a$i)} by simp
with sum-diff1 [of ?B λ w. (c w) *R w]
and (finite ?B)
and (proj2-rep (a$i) ∈ ?B)
have (∑ w ∈ ?Bi. (c w) *R w) =
(∑ w ∈ ?B. (c w) *R w) − c (proj2-rep (a$i)) *R proj2-rep (a$i)
by simp

from a3 /∈ range (($) a): have a3 ≠ a$i by auto
hence insert a3 (range (($) a)) − {a$i} =
insert a3 (range (($) a) − {a$i}) by auto
hence proj2-rep ' (insert a3 (range (($) a)) − {a$i}) = insert ?v ?Bi
by simp
moreover from (proj2-no-3-Col (insert a3 (range (($) a))):
and (a$i ∈ insert a3 (range (($) a))):
have span (proj2-rep ' (insert a3 (range (($) a)) − {a$i})) = UNIV
by (rule proj2-no-3-Col-span)
ultimately have span (insert ?v ?Bi) = UNIV by simp

from (?Bi = ?B − {proj2-rep (a$i)}):
and (proj2-rep (a$i) ∈ ?B):
and (card ?B = 3)
have card ?Bi = 2 by (simp add: card-gt-0-diff-singleton)
hence finite ?Bi by simp
with (card ?Bi = 2): and dim-le-card’ [of ?Bi] have dim ?Bi ≤ 2 by simp
hence dim (span ?Bi) ≤ 2 by (subst dim-span)
then have span ?Bi ≠ UNIV
by clarify (auto simp: dim-UNIV)
with (span (insert ?v ?Bi) = UNIV): and span-redundant
have ?v /∈ span ?Bi by auto

{ assume c (proj2-rep (a$i)) = 0
with (∑ w ∈ ?Bi. (c w) *R w) =
(∑ w ∈ ?B. (c w) *R w) − c (proj2-rep (a$i)) *R proj2-rep (a$i):
and (∑ w ∈ ?B. (c w) *R w) = ?v
have ?v = (∑ w ∈ ?Bi. (c w) *R w)
by simp
with span-finite [of ?Bi] and (finite ?Bi)
have ?v ∈ span ?Bi by (simp add: scalar-equiv)
with (?v /∈ span ?Bi) have False .. } thus c (proj2-rep (a$i)) ≠ 0 ..
qed

hence ∀ w ∈?B. c w ≠ 0
unfolding image-def
by auto

have rows ?C = (λ w. (c w) *R w) ' ?B
unfolding rows-def
and row-def
and image-def
by (auto simp: vec-lambda-eta)
have $\forall \, x. \, x \in \text{span} \, (\text{rows} \, ?C)$

proof
  fix $x :: \text{real}^3$
  from $\text{finite} \, ?B$, and $\text{span-finite} \, [\text{of} \, ?B]$ and $\langle \text{span} \, ?B = \text{UNIV} \rangle$
  obtain $ub$ where $(\sum \, w \in ?B, \, (ub \, w) \ast_{\text{R}} w) = x$
    by (auto simp add: scalar-equiv) (metis (no-types, lifting) $\text{UNIV-I}$ rangeE)
  hence $\langle \sum \, w \in ?B, \, (ub \, w) \ast_{\text{R}} w \in \text{span} \, (\text{rows} \, ?C) \rangle$
  proof
    fix $w$
    assume $w \in ?B$
    with $\text{span-superset} \, [\text{of} \, \text{rows} \, ?C]$ and $\langle \text{rows} \, ?C = \text{image} \, (\lambda \, w. \, (c \, w) \ast_{\text{R}} w) \rangle$
    show $\langle \sum \, w \in ?B, \, (ub \, w) \ast_{\text{R}} w \in \text{span} \, (\text{rows} \, ?C) \rangle$
      by simp
  qed
  hence $\langle \text{span} \, (\text{rows} \, ?C) = \text{UNIV} \rangle$ by auto
  with $\text{matrix-left-invertible-span-rows} \, [\text{of} \, ?C]$
  have $\exists \, C'. \, C' \ast_{\text{B}} ?C = \text{mat} \, 1$ ..
  with $\text{left-invertible-iff-invertible}$
  have $\text{invertible} \, ?C$ ..

have $\langle \text{vector} \, [1,1,1] :: \text{real}^3 \rangle \neq 0$
  unfolding $\text{vector-def}$
  by (simp add: vec-eq-iff forall-3)
  with $\text{apply-cltn2-abs}$ and $\langle \text{invertible} \, ?C \rangle$
  have $\text{apply-cltn2} \, (\text{proj2-abs} \, (\text{vector} \, [1,1,1])) \, ?A$ =
    $\text{proj2-abs} \, (\text{vector} \, [1,1,1] \ast_{\text{B}} ?C)$
    by simp
  from $\text{inj-on-iff-eq-card} \, [\text{of} \, \text{UNIV} \, (\$) \, a]$ and $\langle \text{card} \, (\text{range} \, (\$(\$) \, a)) = 3 \rangle$
  have $\text{inj} \, (\$(\$) \, a)$ by simp
  from $\text{exhaust-3}$ have $\forall \, i :: 3. \, (\text{vector} \, [1::\text{real},1,1])\$i = 1$
    unfolding $\text{vector-def}$
    by auto
  with $\text{vector-matrix-row} \, [\text{of} \, \text{vector} \, [1,1,1] \, ?C]$
  have $(\sum \, i \in \text{UNIV}. \, (\, e \, (\text{proj2-rep} \, (a\$i))) \ast_{\text{R}} (\text{proj2-rep} \, (a\$i)))$
    by simp
  also from $\text{sum.reindex}$
  $[\text{of} \, (\$) \, a \, \text{UNIV} \, \lambda \, x. \, (\text{proj2-rep} \, x) \ast_{\text{R}} (\text{proj2-rep} \, x)]$
  and $\langle \text{inj} \, (\$(\$) \, a) \rangle$

83
have \( \ldots = (\sum x \in (\text{range } ((\$) a)). (c (\text{proj2-rep } x)) *_R (\text{proj2-rep } x)) \)
by simp
also from sum.reindex
[of \text{proj2-rep range } ((\$) a) \lambda w. (c w) *_R w]
and \text{proj2-rep-inj} and subset-inj-on [of \text{proj2-rep UNIV range } ((\$) a)]
have \( \ldots = (\sum w \in \{B. (c w) *_R w\}) \) by simp
also from \( \{w \in \{B. (c w) *_R w\} \} = ?v \) have \( \ldots = ?v \) by simp
finally have (vector \([1,1,1]\)) v* ?C = ?v
with (apply-cltn2 (\text{proj2-abs (vector} \([1,1,1]\)) ?A = \text{proj2-abs (vector} \([1,1,1]\) v* ?C))
have apply-cltn2 (\text{proj2-abs (vector} \([1,1,1]\)) ?A = \text{proj2-abs ?v} by simp
with \text{proj2-abs-rep} have apply-cltn2 (\text{proj2-abs (vector} \([1,1,1]\)) ?A = a3
by simp
have \( \forall j. \text{apply-cltn2} (\text{proj2-abs (axis} \ j 1)) ?A = a\$j \)
proof
fix \ j :: 3
have ((\text{axis} \ j 1):real 3) \neq 0 by (simp add: vec-eq-iff axis-def)
with apply-cltn2-abs and (invertible ?C)
have apply-cltn2 (\text{proj2-abs (axis} \ j 1)) ?A = \text{proj2-abs (axis} \ j 1 v* ?C)
by simp
have \( \forall i \in (\text{UNIV} - \{j\}). \)
((\text{axis} \ j 1)\$i * c (\text{proj2-rep (a}\$i)) *_R (\text{proj2-rep (a}\$i)) = 0
by (simp add: axis-def)
with sum.mono-neutral-left [of \text{UNIV} \{j\}
\lambda i. ((\text{axis} \ j 1)\$i * c (\text{proj2-rep (a}\$i)) *_R (\text{proj2-rep (a}\$i))]
and vector-matrix-row [of \text{axis} \ j 1 ?C]
have (\text{axis} \ j 1) v* ?C = ?C\$j by (simp add: scalar-equiv)
hence (\text{axis} \ j 1) v* ?C = c (\text{proj2-rep (a}\$j)) *_R (\text{proj2-rep (a}\$j)) by simp
with \text{proj2-abs-rep-mult} and \( \forall i. c (\text{proj2-rep (a}\$i)) \neq 0 \)
and (apply-cltn2 (\text{proj2-abs (axis} \ j 1)) ?A = \text{proj2-abs (axis} \ j 1 v* ?C))
show apply-cltn2 (\text{proj2-abs (axis} \ j 1)) ?A = a\$j
by simp
qed
with (apply-cltn2 (\text{proj2-abs (vector} \([1,1,1]\)) ?A = a\$j)
show \( \exists A. \text{apply-cltn2} (\text{proj2-abs (vector} \([1,1,1]\)) \ A = a3 \wedge \)
(\( \forall j. \text{apply-cltn2} (\text{proj2-abs (axis} \ j 1)) \ A = a\$j) by auto
qed

lemma statement53-existence:
fixes \ p :: \text{proj2'4''2}
assumes \( \forall i. \text{proj2-no-3-Col (range } ((\$) (p\$i))) \)
shows \( \exists C. \forall j. \text{apply-cltn2} (p\$0\$j) C = p\$1\$j \)
proof
let \(?q = \chi i. \chi j :: 3. p\$i \$ (of-int (\text{Rep-bit1} \ j)) \)
let \(?D = \chi i. e D. \text{apply-cltn2} (\text{proj2-abs (vector} \([1,1,1]\)) \ D = p\$i\$3 \wedge \)
(\( \forall j'. \text{apply-cltn2} (\text{proj2-abs (axis} \ j' 1)) \ D = ?q\$i\$j') \)
have \( \forall i. \text{apply-cltn2} (\text{proj2-abs (vector} \([1,1,1]\)) ) (\forall D\$i) = p\$i\$3 \)
\( \forall j'. \text{apply-cltn2 (proj2-abs (axis j' 1))} (\forall D\$i) = \forall q\$i\$j') \)

**proof**

fix \( i \)

have range ((\$) (p\$i)) = insert (p\$i\$3) (range ((\$) (?q\$i)))

**proof**

show range ((\$) (p\$i)) \(\supseteq\) insert (p\$i\$3) (range ((\$) (?q\$i))) by auto

show range ((\$) (p\$i)) \(\subseteq\) insert (p\$i\$3) (range ((\$) (?q\$i)))

**proof**

fix \( r \)

assume \( r \in\) range ((\$) (p\$i))

then obtain \( j \) where \( r = p\$i\$j \) by auto

with eq-3-or-of-3 [of \( j \)]

show \( r \in\) insert (p\$i\$3) (range ((\$) (?q\$i))) by auto

qed

**moreover from** \( \forall i. \text{proj2-no-3-Col (range ((\$) (p\$i)))) ; \)

have proj2-no-3-Col (range ((\$) (p\$i))) ..

ultimately have proj2-no-3-Col (insert (p\$i\$3) (range ((\$) (?q\$i))))

by simp

**hence** \( \exists D. \text{apply-cltn2 (proj2-abs (vector [1,1,1]))} D = p\$i\$3 \)

\(\land (\forall j'. \text{apply-cltn2 (proj2-abs (axis j' 1))}) D = \forall q\$i\$j' \)

by (rule statement52-existence)

with some1-ex [of \( \lambda D. \text{apply-cltn2 (proj2-abs (vector [1,1,1]))} D = p\$i\$3 \)

\(\land (\forall j'. \text{apply-cltn2 (proj2-abs (axis j' 1))}) D = \forall q\$i\$j' \)]

show apply-cltn2 (proj2-abs (vector [1,1,1])) (\forall D\$i) = p\$i\$3

\(\land (\forall j'. \text{apply-cltn2 (proj2-abs (axis j' 1))}) (\forall D\$i) = \forall q\$i\$j' \)

by simp

**qed**

**hence** apply-cltn2 (proj2-abs (vector [1,1,1])) (?D\$0) = p\$0\$3

and apply-cltn2 (proj2-abs (vector [1,1,1])) (?D\$1) = p\$1\$3

and \( \forall j'. \text{apply-cltn2 (proj2-abs (axis j' 1))} (?D\$0) = \forall q\$0\$j' \)

and \( \forall j'. \text{apply-cltn2 (proj2-abs (axis j' 1))} (?D\$1) = \forall q\$1\$j' \)

by simp-all

let \( ?C = \text{cltn2-compose (cltn2-inverse (?D\$0))} (?D\$1) \)

have \( \forall j. \text{apply-cltn2 (p\$0\$j)} ?C = p\$1\$j \)

**proof**

fix \( j \)

show apply-cltn2 (p\$0\$j) ?C = p\$1\$j

**proof cases**

assume \( j = 3 \)

with apply-cltn2 (proj2-abs (vector [1,1,1])) (?D\$0) = p\$0\$3;

and cltn2.act-inv-iff

**have**

apply-cltn2 (p\$0\$j) (cltn2-inverse (?D\$0)) = proj2-abs (vector [1,1,1])

by simp

with apply-cltn2 (proj2-abs (vector [1,1,1])) (?D\$1) = p\$1\$3;

and \( j = 3 \)

and cltn2.act-act [of cltn2-inverse (?D\$0) ?D\$1 p\$0\$j]
show apply-cltn2 (p$0$q) $C = p$1$q$ by simp

next
assume $j \neq 3$
with eq-3-or-of-3 obtain $j' :: 3$ where $j$ = of-int (Rep-bit1 $j'$)
by metis
with $\forall j'$. apply-cltn2 (proj2-abs (axis $j' 1$)) (?$D$0) = ?$q$0$q$j'$
and $\forall j'$. apply-cltn2 (proj2-abs (axis $j' 1$)) (?$D$1) = ?$q$1$q$j'$
have p$0$q$j$ = apply-cltn2 (proj2-abs (axis $j' 1$)) (?$D$0)
and p$1$q$j$ = apply-cltn2 (proj2-abs (axis $j' 1$)) (?$D$1)
by simp-all
from (p$0$q$j$) = apply-cltn2 (proj2-abs (axis $j' 1$)) (?$D$0)
and cltn2.act-inv-iff
have apply-cltn2 (p$0$q$j$) (cltn2-inverse (?$D$0)) = proj2-abs (axis $j' 1$)
by simp
with (p$1$q$j$) = apply-cltn2 (proj2-abs (axis $j' 1$)) (?$D$1);
and cltn2.act-act [of cltn2-inverse (?$D$0) ?$D$1 p$0$q$j$]
show apply-cltn2 (p$0$q$j$) $C = p$1$q$j$ by simp
qed
qed
thus $\exists C. \forall j$. apply-cltn2 (p$0$q$j$) $C = p$1$q$j$ by (rule exI [of - $?C$])
qed

lemma apply-cltn2-linear:
assumes $j \ast_R v + k \ast_R w \neq 0$
shows $j \ast_R (v v* cltn2-rep C) + k \ast_R (w v* cltn2-rep C) \neq 0$
(is $?u \neq 0$)
and apply-cltn2 (proj2-abs (j $\ast_R v + k$ $\ast_R w)) C$
= proj2-abs (j $\ast_R (v v* cltn2-rep C) + k$ $\ast_R (w v* cltn2-rep C))$
proof –
have $?u = (j \ast_R v + k \ast_R w) v* cltn2-rep C$
by (simp only: vector-matrix-left-distrib scaleR-vector-matrix-assoc)
with (j $\ast_R v + k$ $\ast_R w \neq 0$); and non-zero-mult-rep-non-zero
show $?u \neq 0$ by simp
from ( $?u = (j \ast_R v + k \ast_R w) v* cltn2-rep C$);
and (j $\ast_R v + k \ast_R w \neq 0$)
and apply-cltn2-left-abs
show apply-cltn2 (proj2-abs (j $\ast_R v + k$ $\ast_R w)) C = proj2-abs $?u$
by simp
qed

lemma apply-cltn2-imp-mult:
assumes apply-cltn2 p $C = q$
shows $\exists k. k \neq 0 \land proj2-rep p v* cltn2-rep C = k \ast_R proj2-rep q$
proof –
have proj2-rep p v* cltn2-rep C \neq 0 by (rule rep-mult-rep-non-zero)
from (apply-cltn2 p $C = q$)
have proj2-abs (proj2-rep p v* cltn2-rep C) = q by (unfold apply-cltn2-def)
hence \( \text{proj2-rep} \ (\text{proj2-abs} \ (\text{proj2-rep} \ p \ v \ast \ \text{cltn2-rep} \ C)) = \text{proj2-rep} \ q \) 
by simp 
with \( \text{proj2-rep} \ p \ v \ast \ \text{cltn2-rep} \ C \neq 0 \) and \( \text{proj2-rep-abs2} \ [\text{of proj2-rep} \ p \ v \ast \ \text{cltn2-rep} \ C] \)
have \( \exists \ j. \ j \neq 0 \land \text{proj2-rep} \ q = j \ast_R (\text{proj2-rep} \ p \ v \ast \ \text{cltn2-rep} \ C) \) by simp 
then obtain \( j \) where \( j \neq 0 \) 
and \( \text{proj2-rep} \ q = j \ast_R (\text{proj2-rep} \ p \ v \ast \ \text{cltn2-rep} \ C) \) by auto 
hence \( \text{proj2-rep} \ p \ v \ast \ \text{cltn2-rep} \ C = (1/j) \ast_R \text{proj2-rep} \ q \) 
by \( \text{simp add: field-simps} \) 
with \( j \neq 0 \)
show \( \exists \ k. \ k \neq 0 \land \text{proj2-rep} \ p \ v \ast \ \text{cltn2-rep} \ C = k \ast_R \text{proj2-rep} \ q \) 
by \( \text{simp add: exI [of - 1/j]} \)
qed

lemma statement55:
assumes \( p \neq q \) and \( \text{apply-cltn2} \ p \ C = q \) and \( \text{apply-cltn2} \ q \ C = p \) and \( \text{proj2-incident} \ p \ l \) and \( \text{proj2-incident} \ q \ l \) and \( \text{proj2-incident} \ r \ l \) shows \( \text{apply-cltn2} \ (\text{apply-cltn2} \ r \ C) \ C = r \)
proof cases
assume \( r = p \) 
with \( \text{apply-cltn2} \ p \ C = q \) and \( \text{apply-cltn2} \ q \ C = p \) 
show \( \text{apply-cltn2} \ (\text{apply-cltn2} \ r \ C) \ C = r \) by simp
next
assume \( r \neq p \) 
from \( \text{apply-cltn2} \ p \ C = q \) and \( \text{apply-cltn2} \ C = p \) 
obtain \( i \) where \( i \neq 0 \) and \( \text{proj2-rep} \ p \ v \ast \ \text{cltn2-rep} \ C = i \ast_R \text{proj2-rep} \ q \) 
by auto
from \( \text{apply-cltn2} \ q \ C = p \) and \( \text{apply-cltn2} \ C = q \) 
obtain \( j \) where \( j \neq 0 \) and \( \text{proj2-rep} \ q \ v \ast \ \text{cltn2-rep} \ C = j \ast_R \text{proj2-rep} \ p \) 
by auto
from \( p \neq q \) and \( \text{proj2-incident} \ p \ l \) and \( \text{proj2-incident} \ q \ l \) and \( \text{proj2-incident} \ r \ l \) and \( \text{proj2-incident-iff} \)
have \( r = p \lor (\exists \ k. \ r = \text{proj2-abs} \ (k \ast_R \text{proj2-rep} \ p + \text{proj2-rep} \ q)) \) 
by fast 
with \( r \neq p \)
obtain \( k \) where \( r = \text{proj2-abs} \ (k \ast_R \text{proj2-rep} \ p + \text{proj2-rep} \ q) \) by auto
from \( p \neq q \) and \( \text{proj2-rep-dependent} \ [\text{of k p 1 q}] \) 
have \( k \ast_R \text{proj2-rep} \ p + \text{proj2-rep} \ q \neq 0 \) by auto

87
with \( r = \text{proj2-abs} \left( k \ast_R \text{proj2-rep} p + \text{proj2-rep} q \right) \)
and \( \text{apply-cltn2-linear} \) of \( k \text{proj2-rep} p I \text{proj2-rep} q \)
\[ \begin{align*}
\text{have} & \ k \ast_R \left( \text{proj2-rep} p v \ast \text{cltn2-rep} C \right) + \text{proj2-rep} q v \ast \text{cltn2-rep} C \neq 0 \\
\text{and} & \ \text{apply-cltn2} \ r \ C \\
& = \text{proj2-abs} \\
& \left( k \ast_R \left( \text{proj2-rep} p v \ast \text{cltn2-rep} C \right) + \text{proj2-rep} q v \ast \text{cltn2-rep} C \right) \\
\text{by} & \ \text{simp-all} \\
\text{with} & \ \left( \text{proj2-rep} p v \ast \text{cltn2-rep} C = i \ast_R \text{proj2-rep} q \right) \\
\text{and} & \ \left( \text{proj2-rep} q v \ast \text{cltn2-rep} C = j \ast_R \text{proj2-rep} p \right) \\
\end{align*} \]

also have \( \text{proj2-abs} \left( \left( k \ast i \right) \ast_R \text{proj2-rep} q + \left( j \ast i \right) \ast_R \text{proj2-rep} p \right) \)
\text{by} \ \text{simp-all} \\
\text{with} \ \text{apply-cltn2-linear} \\
\text{have} \ (\text{apply-cltn2} \ r \ C) \ C = \text{proj2-abs} \\
\left( \left( k \ast i \right) \ast_R \text{proj2-rep} q + \left( j \ast i \right) \ast_R \text{proj2-rep} p \right) \\
\left( \text{numsimpl: algebra-simps} \right) \\
\text{also from} \ \left( i \neq 0 \right) \ \text{and} \ \left( j \neq 0 \right) \ \text{and} \ \text{proj2-abs mult} \\
\text{have} \ \left( \text{numsimpl: algebra-simps} \right) \\
\text{also from} \ (r = \text{proj2-abs} \left( k \ast_R \text{proj2-rep} p + \text{proj2-rep} q \right)) \\
\text{finally show} \ \text{apply-cltn2} \ (\text{apply-cltn2} \ r \ C) \ C = r .
\]

declaration \( \text{cross-ratio} :: \text{proj2} \Rightarrow \text{proj2} \Rightarrow \text{proj2} \Rightarrow \text{proj2} \Rightarrow \text{real} \) \text{where} \( \text{cross-ratio} \ p q r s \triangleq \text{proj2-Col-coeff} \ p q s / \text{proj2-Col-coeff} \ p q r \)

declaration \( \text{cross-ratio-correct} :: \text{proj2} \Rightarrow \text{proj2} \Rightarrow \text{proj2} \Rightarrow \text{proj2} \Rightarrow \text{bool} \) \text{where} \( \text{cross-ratio-correct} \ p q r s \triangleq \text{proj2-set-Col} \ \{ p, q, r, s \} \land p \neq q \land r \neq p \land s \neq p \land r \neq q \)

lemma \( \text{proj2-Col-coeff-abs} \): \\
\text{assumes} \ p \neq q \ \text{and} \ j \neq 0 \\
\text{shows} \ \text{proj2-Col-coeff} \ p q \left( \text{proj2-abs} \left( i \ast_R \text{proj2-rep} p + j \ast_R \text{proj2-rep} q \right) \right) \\
= \left( i/j \right) \left( \text{is proj2-Col-coeff} \ p q ?r = i/j \right) \\
\text{proof} \ - \\
\text{from} \ \left( j \neq 0 \right) \\
\]
and proj2-abs-mult [of 1/j i *R proj2-rep p + j *R proj2-rep q]
have ?r = proj2-abs ((i/j) *R proj2-rep p + proj2-rep q)
  by (simp add: scaleR-right-distrib)
from (p ≠ q) and proj2-rep-dependent [of p 1 q]
have (i/j) *R proj2-rep p + proj2-rep q ≠ 0 by auto
with (?[r = proj2-abs ((i/j) *R proj2-rep p + proj2-rep q)]
  and proj2-rep-abs2
obtain k where k ≠ 0
  and proj2-rep ?r = k *R ((i/j) *R proj2-rep p + proj2-rep q)
  by auto
hence (k+i/j) *R proj2-rep p + k *R proj2-rep q - proj2-rep ?r = 0
  by (simp add: algebra-simps)
by (simp add: scaleR-right-distrib)
hence (k+i/j) *R proj2-rep p + k *R proj2-rep q = 0
  by (simp add: algebra-simps)
with (k ≠ 0) and proj2-rep-dependent have p = q by simp
with (p ≠ q) show False ..
qed
with (proj2-Col p q ?r) and (p ≠ q)
have ?r = proj2-abs (proj2-Col-coeff p q ?r *R proj2-rep p + proj2-rep q)
  by (rule proj2-Col-coeff)
with (p ≠ q) and (?r = proj2-abs ((i/j) *R proj2-rep p + proj2-rep q))
  and proj2-Col-coeff-unique
show proj2-Col-coeff p q ?r = i/j by simp
qed

lemma proj2-set-Col-coeff:
  assumes proj2-set-Col S and {p,q,r} ⊆ S and p ≠ q and r ≠ p
  shows r = proj2-abs (proj2-Col-coeff p q r *R proj2-rep p + proj2-rep q)
  (is r = proj2-abs (?i *R ?u + {?v}))
proof –
from {p,q,r} ⊆ S and (proj2-set-Col S)
have proj2-set-Col {p,q,r} by (rule proj2-subset-Col)
by (subst proj2-Col-iff-set-Col)
with (p ≠ q) and (r ≠ p) and proj2-Col-coeff
show r = proj2-abs (?i *R ?u + {?v}) by simp
qed

lemma cross-ratio-abs:
  fixes u v :: real and i j k l :: real
assumes $u \neq 0$ and $v \neq 0$ and $\text{proj2-abs } u \neq \text{proj2-abs } v$
and $j \neq 0$ and $l \neq 0$
shows cross-ratio $(\text{proj2-abs } u) (\text{proj2-abs } v)$
$(\text{proj2-abs } (i *_R u + j *_R v))$
$(\text{proj2-abs } (k *_R u + l *_R v))$
$= j * k / (i * l)$
(is cross-ratio $?p ?q ?r ?s =$ -)

proof –
from $(u \neq 0)$ and $\text{proj2-rep-abs2}$
obtain $g$ where $g \neq 0$ and $\text{proj2-rep } ?p = g *_R u$ by auto
from $(v \neq 0)$ and $\text{proj2-rep-abs2}$
obtain $h$ where $h \neq 0$ and $\text{proj2-rep } ?q = h *_R v$ by auto
with $(g \neq 0)$ and $(\text{proj2-rep } ?p = g *_R u)$
    have $?r = \text{proj2-abs } (((i/g) *_R \text{proj2-rep } ?p + (j/h) *_R \text{proj2-rep } ?q)$
    and $?s = \text{proj2-abs } (((k/g) *_R \text{proj2-rep } ?p + (l/h) *_R \text{proj2-rep } ?q)$
    by (simp-all add: field-simps)
with $(?p \neq ?q)$ and $(h \neq 0)$ and $(j \neq 0)$ and $(l \neq 0)$ and $\text{proj2-Col-coeff-abs}$
    have $\text{proj2-Col-coeff } ?p ?q ?r = h * i / (g * j)$
    and $\text{proj2-Col-coeff } ?p ?q ?s = h * k / (g * l)$
    by simp-all
with $(g \neq 0)$ and $(h \neq 0)$
    show cross-ratio $?p ?q ?r ?s = j * k / (i * l)$
    by (unfold cross-ratio-def) (simp add: field-simps)
qed

lemma cross-ratio-accept2:
assumes $p \neq q$
shows cross-ratio $p q$
$(\text{proj2-abs } (i *_R \text{proj2-rep } p + \text{proj2-rep } q))$
$(\text{proj2-abs } (j *_R \text{proj2-rep } p + \text{proj2-rep } q))$
$= j / i$
(is cross-ratio $p q ?r ?s =$ -)

proof –
let $?u = \text{proj2-rep } p$
let $?v = \text{proj2-rep } q$
    have $?u \neq 0$ and $?v \neq 0$ by (rule proj2-rep-non-zero)+
    have $\text{proj2-abs } ?u = p$ and $\text{proj2-abs } ?v = q$ by (rule proj2-abs-rep)+
with $(?u \neq 0)$ and $(?v \neq 0)$ and $(p \neq q)$ and $\text{cross-ratio-abs } [of ?u ?v 1 1 i j]$
    show cross-ratio $p q ?r ?s = j / i$ by simp
qed

lemma cross-ratio-correct-cltn2:
assumes $\text{cross-ratio-correct } p q r s$
shows cross-ratio-correct $(\text{apply-cltn2 } p C)$ $(\text{apply-cltn2 } q C)$
$(\text{apply-cltn2 } r C)$ $(\text{apply-cltn2 } s C)$
(is cross-ratio-correct $?pC ?qC ?rC ?sC)$

proof –

90
proof

lemma cross-ratio-cltn2

have proj2-set-Col \{p,q,r,s\}
  and p \neq q and r \neq p and s \neq p and r \neq q
  by (unfold cross-ratio-correct-def) simp-all

have \{apply-cltn2 t C \mid t \in \{p,q,r,s\}\} = \{\?pC,\?qC,\?rC,\?sC\} by auto
with \{proj2-set-Col \{p,q,r,s\}\}
  and apply-cltn2-preserve-set-Col [of \{p,q,r,s\} C]
have proj2-set-Col \{\?pC,\?qC,\?rC,\?sC\} by simp

have \{p \neq q\} and (r \neq p) and (s \neq p) and (r \neq q) and apply-cltn2-injective
with \{proj2-set-Col \{p,q,r,s\}\}
  and apply-cltn2-injective [of \{p,q,r,s\}\{p,q,r,s\} C]

show cross-ratio-correct \?pC \?qC \?rC \?sC
  by (unfold cross-ratio-correct-def) simp

qed

lemma cross-ratio-cltn2:

assumes proj2-set-Col \{p,q,r,s\} and p \neq q and r \neq p and s \neq p

shows cross-ratio (apply-cltn2 p C) (apply-cltn2 q C)
  (apply-cltn2 r C) (apply-cltn2 s C)
= cross-ratio p q r s

(is cross-ratio \?pC \?qC \?rC \?sC = -)

proof –

let \?u = proj2-rep p
let \?v = proj2-rep q
let \?i = proj2-Col-coeff p q r
let \?j = proj2-Col-coeff p q s
from \{proj2-set-Col \{p,q,r,s\}\} and \{p \neq q\} and \{r \neq p\} and \{s \neq p\}
  and proj2-set-Col-coeff
have r = proj2-abs (\?i * _\?R \?u + ?v) and s = proj2-abs (\?j * _\?R \?u + ?v)
  by simp-all

let \?uC = \?u v* cltn2-rep C
let \?vC = \?v v* cltn2-rep C
have \?uC \neq 0 and \?vC \neq 0 by (rule rep-mult-rep-non-zero)+

have proj2-abs \?uC = \?pC and proj2-abs \?vC = \?qC
  by (unfold apply-cltn2-def) simp-all

from \{p \neq q\} and apply-cltn2-injective have \?pC \neq \?qC by fast

from \{p \neq q\} and proj2-rep-dependent [of \{-p 1 q\}]
have \?i * _\?R \?u + ?v \neq 0 and \?j * _\?R \?u + ?v \neq 0 by auto
with \{r = proj2-abs (\?i * _\?R \?u + ?v)\} and \{s = proj2-abs (\?j * _\?R \?u + ?v)\}
  and apply-cltn2-linear [of \?i \?u 1 ?v]
  and apply-cltn2-linear [of \?j \?u 1 ?v]
have \?rC = proj2-abs (\?i * _\?R \?uC + ?vC)
  and \?sC = proj2-abs (\?j * _\?R \?uC + ?vC)

91
by simp-all
with (\?uC \neq 0) and (\?vC \neq 0) and (proj2-abs \?uC = \?uC)
and (proj2-abs \?uC = \?qC) and (\?pC \neq \?qC)
and cross-ratio-abs [of \?uC \?vC \?qC \?pC]
have cross-ratio \?uC \?qC \?rC \?pC = \?j/\?i
by simp
thus cross-ratio \?uC \?qC \?rC \?sC = cross-ratio \?pC \?qC \?rC \?sC
 unfolding cross-ratio-def [of \?pC \?qC \?rC \?sC].

qed

lemma cross-ratio-unique:
assumes cross-ratio-correct \?pC \?qC \?rC \?sC
and cross-ratio-correct \?pC \?qC \?rC \?tC
and cross-ratio \?pC \?qC \?rC \?sC = cross-ratio \?pC \?qC \?rC \?tC
shows \?sC = \?tC
proof
−
from (cross-ratio-correct \?pC \?qC \?rC \?sC)
and (cross-ratio-correct \?pC \?qC \?rC \?tC)
have proj2-set-Col \{\?pC, \?qC, \?rC, \?sC\}
and \?pC \neq \?qC
and \?rC \neq \?pC
and \?rC \neq \?qC
and \?sC \neq \?pC
and \?tC \neq \?pC
by (unfold cross-ratio-correct-def) simp-all
let \?u = proj2-rep \?pC
let \?v = proj2-rep \?qC
let \?i = proj2-Col-coeff \?pC \?qC \?rC
let \?j = proj2-Col-coeff \?pC \?qC \?sC
let \?k = proj2-Col-coeff \?pC \?qC \?tC
from (proj2-set-Col \{\?pC, \?qC, \?rC, \?sC\})
and (proj2-set-Col \{\?pC, \?qC, \?rC, \?tC\})
and \?pC \neq \?qC
and \?rC \neq \?pC
and \?rC \neq \?qC
and \?sC \neq \?pC
and \?tC \neq \?pC
have \?rC = proj2-abs (\?i \*R \?u + \?v)
and \?sC = proj2-abs (\?j \*R \?u + \?v)
and \?tC = proj2-abs (\?k \*R \?u + \?v)
by simp-all
from (\?rC \neq \?qC)
and (\?rC = proj2-abs (\?i \*R \?u + \?v))
have \?i \neq 0 by (auto simp add: proj2-abs-rep)
with (cross-ratio \?pC \?qC \?rC \?sC = cross-ratio \?pC \?qC \?rC \?tC)
have \?jC = \?kC by (unfold cross-ratio-def) simp
with (\?sC = proj2-abs (\?j \*R \?u + \?v))
and (\?tC = proj2-abs (\?k \*R \?u + \?v))
show \?sC = \?tC by simp
qed

lemma cltn2-three-point-line:
assumes \?pC \neq \?qC
and \?pC \neq \?pC
and \?rC \neq \?qC
and proj2-incident \?lC \?pC and proj2-incident \?qC \?lC
and proj2-incident \?lC \?rC
and proj2-incident \?lC \?tC
shows proj2-cltn2 \?sC \?C = \?sC
(is \?sC = \?sC)
proof
−
assume \?sC = \?sC
with (proj2-cltn2 \?pC \?lC \?pC) show \?sC = \?sC by simp
next

92
assume \( s \neq p \)

let \( ?pC = \text{apply-cltn2} \ p \ C \)

let \( ?qC = \text{apply-cltn2} \ q \ C \)

let \( ?rC = \text{apply-cltn2} \ r \ C \)

from \( \langle \text{proj2-incident} \ p \ l \rangle \) and \( \langle \text{proj2-incident} \ q \ l \rangle \) and \( \langle \text{proj2-incident} \ r \ l \rangle \) and \( \langle \text{proj2-incident} \ s \ l \rangle \)

have \( \text{proj2-set-Col} \ \{p,q,r,s\} \) by (unfold \text{proj2-set-Col-def}) auto

with \( \langle p \neq q \rangle \) and \( \langle r \neq p \rangle \) and \( \langle s \neq p \rangle \) and \( \langle r \neq q \rangle \)

have \( \text{cross-ratio-correct} \ p \ q \ r \ s \) by (unfold \text{cross-ratio-correct-def}) simp

hence \( \text{cross-ratio-correct} \ ?pC \ ?qC \ ?rC \ ?sC \)

by (rule \text{cross-ratio-correct-cltn2})

with \( \langle ?pC = p \rangle \) and \( \langle ?qC = q \rangle \) and \( \langle ?rC = r \rangle \)

have \( \text{cross-ratio-correct} \ p \ q \ r \ ?sC \) by simp

from \( \langle \text{proj2-set-Col} \ \{p,q,r,s\} \rangle \) and \( \langle p \neq q \rangle \) and \( \langle r \neq p \rangle \) and \( \langle s \neq p \rangle \)

have \( \text{cross-ratio} \ ?pC \ ?qC \ ?rC \ ?sC = \text{cross-ratio} \ p \ q \ r \ s \)

by (rule \text{cross-ratio-cltn2})

with \( \langle ?pC = p \rangle \) and \( \langle ?qC = q \rangle \) and \( \langle ?rC = r \rangle \)

have \( \text{cross-ratio} \ p \ q \ r \ ?sC = \text{cross-ratio} \ p \ q \ r \ s \) by simp

with \( \langle \text{cross-ratio-correct} \ p \ q \ r \ ?sC \rangle \) and \( \langle \text{cross-ratio-correct} \ p \ q \ r \ s \rangle \)

show \( ?sC = s \) by (rule \text{cross-ratio-unique})

qed

lemma \text{cross-ratio-equal-cltn2}:

assumes \( \text{cross-ratio-correct} \ p \ q \ r \ s \)

and \( \text{cross-ratio-correct} \ (\text{apply-cltn2} \ p \ C) \ (\text{apply-cltn2} \ q \ C) \)

(apply-cltn2 \ r \ C) \ t

(is \( \text{cross-ratio-correct} \ ?pC \ ?qC \ ?rC \ t \))

and \( \text{cross-ratio} \ (\text{apply-cltn2} \ p \ C) \ (\text{apply-cltn2} \ q \ C) \) (apply-cltn2 \ r \ C) \ t

= \( \text{cross-ratio} \ p \ q \ r \ s \)

shows \( t = \text{apply-cltn2} \ s \ C \) (is \( t = ?sC \))

proof –

from \( \langle \text{cross-ratio-correct} \ p \ q \ r \ s \rangle \)

have \( \text{cross-ratio-correct} \ ?pC \ ?qC \ ?rC \ ?sC \) by (rule \text{cross-ratio-correct-cltn2})

from \( \langle \text{cross-ratio-correct} \ p \ q \ r \ s \rangle \)

have \( \text{proj2-set-Col} \ \{p,q,r,s\} \) and \( p \neq q \) and \( r \neq p \) and \( s \neq p \)

by (unfold \text{cross-ratio-correct-def}) simp-all

hence \( \text{cross-ratio} \ ?pC \ ?qC \ ?rC \ ?sC = \text{cross-ratio} \ p \ q \ r \ s \)

by (rule \text{cross-ratio-cltn2})

with \( \langle \text{cross-ratio} \ ?pC \ ?qC \ ?rC \ t = \text{cross-ratio} \ p \ q \ r \ s \rangle \)

have \( \text{cross-ratio} \ ?pC \ ?qC \ ?rC \ t = \text{cross-ratio} \ ?pC \ ?qC \ ?rC \ ?sC \) by simp

with \( \langle \text{cross-ratio-correct} \ ?pC \ ?qC \ ?rC \ t \rangle \)

and \( \langle \text{cross-ratio-correct} \ ?pC \ ?qC \ ?rC \ ?sC \rangle \)

show \( t = ?sC \) by (rule \text{cross-ratio-unique})

qed
lemma proj2-Col-distinct-coeff-non-zero:
  assumes proj2-Col p q r and p ≠ q and r ≠ p and r ≠ q
  shows proj2-Col-coeff p q r ≠ 0
proof
  assume proj2-Col-coeff p q r = 0
  from ⟨proj2-Col p q r⟩ and ⟨p ≠ q⟩ and ⟨r ≠ p⟩
  have r = proj2-abs ((proj2-Col-coeff p q r) *_R proj2-rep p + proj2-rep q)
  by (rule proj2-Col-coeff)
  with ⟨proj2-Col-coeff p q r = 0⟩ have r = q by (simp add: proj2-abs-rep)
  with ⟨r ≠ q⟩ show False ..
qed

lemma cross-ratio-product:
  assumes proj2-Col p q s and p ≠ q and s ≠ p and s ≠ q
  shows cross-ratio p q r s ≈ cross-ratio p q s t = cross-ratio p q r t
proof
  from ⟨proj2-Col p q s⟩ and ⟨p ≠ q⟩ and ⟨s ≠ p⟩ and ⟨s ≠ q⟩
  have proj2-Col-coeff p q s ≠ 0 by (rule proj2-Col-distinct-coeff-non-zero)
  thus cross-ratio p q r s ≈ cross-ratio p q s t = cross-ratio p q r t
  by (unfold cross-ratio-def) simp
qed

lemma cross-ratio-equal-1:
  assumes proj2-Col p q r and p ≠ q and r ≠ p and r ≠ q
  shows cross-ratio p q r r = 1
proof
  from ⟨proj2-Col p q r⟩ and ⟨p ≠ q⟩ and ⟨r ≠ p⟩ and ⟨r ≠ q⟩
  have proj2-Col-coeff p q r ≠ 0 by (rule proj2-Col-distinct-coeff-non-zero)
  thus cross-ratio p q r r = 1 by (unfold cross-ratio-def) simp
qed

lemma cross-ratio-1-equal:
  assumes cross-ratio-correct p q r s and cross-ratio p q r s = 1
  shows r = s
proof
  from ⟨cross-ratio-correct p q r s⟩
  have proj2-set-Col {p,q,r,s} and p ≠ q and r ≠ p and r ≠ q
  by (unfold cross-ratio-correct-def) simp-all
  from ⟨proj2-set-Col {p,q,r,s}⟩
  have proj2-set-Col {p,q,r}
  by (simp add: proj2-subset-Col [of {p,q,r} {p,q,r,s}])
  with ⟨p ≠ q⟩ and ⟨r ≠ p⟩ and ⟨r ≠ q⟩
  have cross-ratio-correct p q r r by (unfold cross-ratio-correct-def) simp
  from ⟨proj2-set-Col {p,q,r}⟩
  have proj2-Col p q r by (subst proj2-Col-iff-set-Col)
  with ⟨p ≠ q⟩ and ⟨r ≠ p⟩ and ⟨r ≠ q⟩
have cross-ratio $p\ q\ r\ r = 1$ by (simp add: cross-ratio-equal-1)
with (cross-ratio $p\ q\ r\ s = 1$)
have cross-ratio $p\ q\ r\ r = cross-ratio\ p\ q\ r\ s$ by simp
with (cross-ratio-correct $p\ q\ r\ r$ and (cross-ratio-correct $p\ q\ r\ s$)
show $r = s$ by (rule cross-ratio-unique)
qed

lemma cross-ratio-swap-34:
shows cross-ratio $p\ q\ r\ s = 1$ / (cross-ratio $p\ q\ r\ s$)
by (unfold cross-ratio-def) simp

lemma cross-ratio-swap-13-24:
assumes cross-ratio-correct $p\ q\ r\ s$ and $r \neq s$
shows cross-ratio $r\ s\ p\ q = cross-ratio\ p\ q\ r\ s$
proof –
from (cross-ratio-correct $p\ q\ r\ s$)
have proj2-set-Col $\{p,q,r,s\}$ and $p \neq q$ and $r \neq p$ and $s \neq p$ and $r \neq q$
by (unfold cross-ratio-correct-def, simp-all)

have proj2-rep $p \neq 0$ (is $?u \neq 0$) and proj2-rep $q \neq 0$ (is $?v \neq 0$)
by (rule proj2-rep-non-zero)+

have $p = proj2-abs\ ?u$ and $q = proj2-abs\ ?v$
by (simp-all add: proj2-abs-rep)
with ($p \neq q$) have proj2-abs $?u \neq proj2-abs\ ?v$ by simp

let $?i = proj2-Col-coeff\ p\ q\ r$
let $?j = proj2-Col-coeff\ p\ q\ s$
from (proj2-set-Col $\{p,q,r,s\}$) and ($p \neq q$) and ($r \neq p$) and ($s \neq p$)
have $r = proj2-abs\ (?i \ast\ ?u + ?v)$ (is $r = proj2-abs\ ?u$)
and $s = proj2-abs\ (?j \ast\ ?u + ?v)$ (is $s = proj2-abs\ ?x$)
by (simp-all add: proj2-set-Col-coef)
with ($r \neq s$) have $?i \neq ?j$ by auto

from ($?u \neq 0$) and ($?v \neq 0$) and (proj2-abs $?u \neq proj2-abs\ ?v$)
and dependent-proj2-abs [of $?u\ ?v - 1$]
have $?w \neq 0$ and $?x \neq 0$ by auto

from ($r = proj2-abs\ (?i \ast\ ?u + ?v)$) and ($r \neq q$)
have $?i \neq 0$ by (auto simp add: proj2-abs-rep)

have $?w - ?x = (?i - ?j) \ast\ ?u$ by (simp add: algebra-simps)
with ($?i \neq ?j$)

have $p = proj2-abs\ (?w - ?x)$ by (simp add: proj2-abs-mult-rep)

have $?j \ast\ ?w - ?i \ast\ ?x = (?j - ?i) \ast\ ?v$ by (simp add: algebra-simps)
with ($?i \neq ?j$)

have $q = proj2-abs\ (?j \ast\ ?w - ?i \ast\ ?x)$ by (simp add: proj2-abs-mult-rep)
with ($?w \neq 0$) and ($?x \neq 0$) and ($r \neq s$) and ($?i \neq 0$) and ($r = proj2-abs\ ?w$)
and \( s = \text{proj2-abs} \ ?x \) and \( p = \text{proj2-abs} \ (\ ?w - \ ?x) \)
and cross-ratio-abs [of \( \ ?w \ ?x - 1 - ?i \ ?j \)]

have cross-ratio \( r \ s \ p \ q = \ ?j / \ ?i \) by (simp add: algebra-simps)

thus cross-ratio \( r \ s \ p \ q = \text{cross-ratio} \ p \ q \ r \ s \)

by (unfold cross-ratio-def [of \( p \ q \ r \ s \)], simp)

qed

lemma cross-ratio-swap-12:
assumes cross-ratio-correct \( p \ q \ r \ s \) and cross-ratio-correct \( q \ p \ r \ s \)
shows cross-ratio \( q \ p \ r \ s = 1 / \text{(cross-ratio} \ p \ q \ r \ s \)

proof cases
assume \( r = s \)

from (cross-ratio-correct \( p \ q \ r \ s \))

have proj2-set-Col \{p, q, r, s\} and \( p \neq q \) and \( r \neq p \) and \( r \neq q \)

by (unfold cross-ratio-correct-def) simp-all

from (proj2-set-Col \{p, q, r, s\}) and \( r = s \)

have proj2-Col \( p \ q \ r \) by (simp-all add: proj2-Col-iff-set-Col)

hence proj2-Col \( q \ p \ r \) by (rule proj2-Col-permute)

with (proj2-Col \( p \ q \ r \) and \( p \neq q \) and \( r \neq p \) and \( r \neq q \) and \( r = s \))

have cross-ratio \( p \ q \ r \ s = 1 \) and cross-ratio \( p \ q \ r \ s = 1 \)

by (simp-all add: cross-ratio-equal-1)

thus cross-ratio \( q \ p \ r \ s = 1 / \text{(cross-ratio} \ p \ q \ r \ s \) by simp

next
assume \( r \neq s \)

with (cross-ratio-correct \( q \ p \ r \ s \))

have cross-ratio \( q \ p \ r \ s = \text{cross-ratio} \ r \ s \ q \ p \)

by (simp add: cross-ratio-swap-13-24)

also have \ldots = 1 / \text{(cross-ratio} \ r \ s \ p \ q \) by (rule cross-ratio-swap-34)

also from (cross-ratio-correct \( p \ q \ r \ s \) and \( r \neq s \))

have \ldots = 1 / \text{(cross-ratio} \ p \ q \ r \ s \) by (simp add: cross-ratio-swap-13-24)

finally show cross-ratio \( p \ q \ r \ s = 1 / \text{(cross-ratio} \ p \ q \ r \ s \)

qed

7.6 Cartesian subspace of the real projective plane

definition vector2-append1 :: \text{real}^2 \Rightarrow \text{real}^3 where
vector2-append1 \( v = \text{vector} \{v\$1, v\$2, 1\} \)

lemma vector2-append1-non-zero: vector2-append1 \( v \neq 0 \)

proof
have (vector2-append1 \( v\$3 \neq 0\$3 \)

unfolding vector2-append1-def and vector-def

by simp

thus vector2-append1 \( v \neq 0 \) by auto

qed

definition proj2-pt :: \text{real}^2 \Rightarrow \text{proj2} where
proj2-pt $v \triangleq$ proj2-abs (vector2-append1 $v$)

**Lemma:** proj2-pt-scalar:
\[ \exists \ c. \ c \neq 0 \land \text{proj2-rep (proj2-pt $v$)} = c \ast_R \text{vector2-append1} \ v \]

**Unfolding:**
proj2-pt-def
by (simp add: proj2-rep-abs2 vector2-append1-non-zero)

**Abbreviation:**
$z$-non-zero :: proj2 $\Rightarrow$ bool where
$z$-non-zero $p \triangleq (\text{proj2-rep} \ p)_3 \neq 0$

**Definition:**
cart2-pt :: proj2 $\Rightarrow$ real$^2$ where
cart2-pt $p \triangleq \text{vector} [(\text{proj2-rep} \ p)_1 / (\text{proj2-rep} \ p)_3, (\text{proj2-rep} \ p)_2 / (\text{proj2-rep} \ p)_3]$

definition cart2-append1 :: proj2 $\Rightarrow$ real$^3$ where
cart2-append1 $p \triangleq \left( \frac{1}{((\text{proj2-rep} \ p)_3)} \right) \ast_R \text{proj2-rep} \ p$

**Lemma:**
cart2-append1-z:
assumes $z$-non-zero $p$
shows $(\text{cart2-append1} \ p)_3 = 1$
using (z-non-zero $p$)
by (unfold cart2-append1-def) simp

**Lemma:**
cart2-append1-non-zero:
assumes $z$-non-zero $p$
shows cart2-append1 $p \neq 0$
proof
  from (z-non-zero $p$) have $(\text{cart2-append1} \ p)_3 = 1$ by (rule cart2-append1-z)
  thus cart2-append1 $p \neq 0$ by (simp add: vec-eq-iff exI [of - 3])
qed

**Lemma:**
proj2-rep-cart2-append1:
assumes $z$-non-zero $p$
shows $\text{proj2-rep} \ p = ((\text{proj2-rep} \ p)_3) \ast_R \text{cart2-append1} \ p$
using (z-non-zero $p$)
by (unfold cart2-append1-def) simp

**Lemma:**
proj2-abs-cart2-append1:
assumes $z$-non-zero $p$
shows $\text{proj2-abs} \ (\text{cart2-append1} \ p) = p$
proof
  from (z-non-zero $p$)
  have proj2-abs (cart2-append1 $p$) = proj2-abs (proj2-rep $p$)
    by (unfold cart2-append1-def) (simp add: proj2-abs-mult)
  thus proj2-abs (cart2-append1 $p$) = $p$ by (simp add: proj2-abs-rep)
qed

**Lemma:**
cart2-append1-inj:
assumes $z$-non-zero $p$ and cart2-append1 $p = \text{cart2-append1} \ q$

\[97\]
shows $p = q$

proof –

from (z-non-zero $p$) have (cart2-append1 $p$)$^3 = 1$ by (rule cart2-append1-z)

with (cart2-append1 $p = cart2-append1 q$)

have (cart2-append1 $q$)$^3 = 1$ by simp

hence z-non-zero $q$ by (unfold cart2-append1-def) auto

from (cart2-append1 $p = cart2-append1 q$)

have proj2-abs (cart2-append1 $p$) = proj2-abs (cart2-append1 $q$) by simp

with (z-non-zero $p$) and (z-non-zero $q$)

show $p = q$ by (simp add: proj2-abs-cart2-append1)

qed

lemma cart2-append1:

assumes z-non-zero $p$

shows vector2-append1 (cart2-pt $p$) = cart2-append1 $p$

using (z-non-zero $p$)

unfolding vector2-append1-def

and cart2-append1-def

and cart2-pt-def

and vector-def

by (simp add: vec-eq-iff forall-3)

lemma cart2-proj2:

cart2-pt (proj2-pt $v$) = $v$

proof –

let $?v' = vector2-append1 v$

let $?p = proj2-pt v$

from proj2-pt-scalar

obtain $c$ where $c \neq 0$ and proj2-rep $?p = c \ast_R ?v'$ by auto

hence (cart2-pt $?p$)$^1 = v^1$ and (cart2-pt $?p$)$^2 = v^2$

unfolding cart2-pt-def and vector2-append1-def and vector-def

by simp+

thus cart2-pt $?p = v$ by (simp add: vec-eq-iff forall-2)

qed

lemma z-non-zero-proj2-pt:

z-non-zero (proj2-pt $v$)

proof –

from proj2-pt-scalar

obtain $c$ where $c \neq 0$ and proj2-rep (proj2-pt $v$) = $c \ast_R$ (vector2-append1 $v$)

by auto

from (proj2-rep (proj2-pt $v$) = $c \ast_R$ (vector2-append1 $v$))

have (proj2-rep (proj2-pt $v$))$^3 = c$

unfolding vector2-append1-def and vector-def

by simp

with ($c \neq 0$) show z-non-zero (proj2-pt $v$) by simp

qed

lemma cart2-append1-proj2:

cart2-append1 (proj2-pt $v$) = vector2-append1 $v$

proof –
from z-non-zero-proj2-pt
have cart2-append1 (proj2-pt v) = vector2-append1 (cart2-pt (proj2-pt v))
  by (simp add: cart2-append1)
thus cart2-append1 (proj2-pt v) = vector2-append1 v
  by (simp add: cart2-proj2)
qed

lemma proj2-pt-inj: inj proj2-pt
  by (simp add: inj-on-inverseI [of UNIV cart2-pt proj2-pt] cart2-proj2)

lemma proj2-cart2:
  assumes z-non-zero p
  shows proj2-pt (cart2-pt p) = p
proof
  from ⟨z-non-zero p⟩
  have (proj2-rep p)$\exists R \cdot vector2-append1 (cart2-pt p) = proj2-rep p
    unfolding vector2-append1-def and cart2-pt-def and vector-def
    by (simp add: vec-eq-iff forall-3)
  with ⟨z-non-zero p⟩
  have proj2-abs (vector2-append1 (cart2-pt p)) = proj2-abs (proj2-rep p)
    by simp
  thus proj2-pt (cart2-pt p) = p
    by (unfold proj2-pt-def) (simp add: proj2-abs-rep)
qed

lemma cart2-injective:
  assumes z-non-zero p and z-non-zero q and cart2-pt p = cart2-pt q
  shows p = q
proof
  from ⟨z-non-zero p⟩ and ⟨z-non-zero q⟩
  have proj2-pt (cart2-pt p) = p and proj2-pt (cart2-pt q) = q
    by (simp-all add: proj2-cart2)
  from ⟨proj2-pt (cart2-pt p) = p⟩ and ⟨cart2-pt p = cart2-pt q⟩
  have proj2-pt (cart2-pt q) = p by simp
  with ⟨proj2-pt (cart2-pt q) = q⟩ show p = q by simp
qed

lemma proj2-Col-iff-euclid:
  proj2-Col (proj2-pt a) (proj2-pt b) (proj2-pt c) ←→ real-euclid.Col a b c
(is proj2-Col ?p ?q ?r ←→ -)
proof
  let ?a' = vector2-append1 a
  let ?b' = vector2-append1 b
  let ?c' = vector2-append1 c
  let ?a'' = proj2-rep ?p
  let ?b'' = proj2-rep ?q
  let ?c'' = proj2-rep ?r
from proj2-pt-scalar obtain i and j and k where
  i ≠ 0 and ?a'' = i *R ?a'
  and j ≠ 0 and ?b'' = j *R ?b'
  and k ≠ 0 and ?c'' = k *R ?c'
by metis
hence ?a'' = (1/i) *R ?a'
and ?b'' = (1/j) *R ?b'
and ?c'' = (1/k) *R ?c'
by simp-all

{ assume proj2-Col ?p ?q ?r
  then obtain i' and j' and k' where
    i' *R ?a'' + j' *R ?b'' + k' *R ?c'' = 0
and i' ≠ 0 \lor j' ≠ 0 \lor k' ≠ 0
  unfolding proj2-Col-def
by auto

let ?i'' = i * i'
let ?j'' = j * j'
let ?k'' = k * k'
from i ≠ 0 \land (j ≠ 0) \land (k ≠ 0) \land (i' ≠ 0 \lor j' ≠ 0 \lor k' ≠ 0)
  have ?i'' ≠ 0 \lor ?j'' ≠ 0 \lor ?k'' ≠ 0 by simp

from i' *R ?a'' + j' *R ?b'' + k' *R ?c'' = 0
  and (?a'' = i *R ?a'
  and (?b'' = j *R ?b'
  and (?c'' = k *R ?c'
  have ?i'' *R ?a'' + ?j'' *R ?b'' + ?k'' *R ?c'' = 0
  by (simp add: ac-simps)
  by simp
  hence ?i'' + ?j'' + ?k'' = 0
  unfolding vector2-append1-def and vector-def
  by simp

  have (?i'' *R ?a'' + ?j'' *R ?b'' + ?k'' *R ?c'\$1)
    = (?i'' *R a + ?j'' *R b + ?k'' *R c)\$1
    = (?i'' *R a + ?j'' *R b + ?k'' *R c)\$2
  unfolding vector2-append1-def and vector-def
  by simp+
  have ?i'' *R a + ?j'' *R b + ?k'' *R c = 0
  by (simp add: vec-eq-iff forall-2)

  have dep2 (b - a) (c - a)
proof cases
  assume ?k'' = 0
  with (?i'' + ?j'' + ?k'' = 0; have ?j'' = -?i'' by simp
  with (?i'' ≠ 0 \lor ?j'' ≠ 0 \lor ?k'' ≠ 0) \land (?k'' = 0; have ?i'' ≠ 0 by simp
from (\exists i'' *R a + \exists j'' *R b + \exists k'' *R c = 0)
  and (\exists k'' = 0) and (\exists j'' = -\exists i''),
have (\exists i'' *R a + (-(\exists i'') *R b)) = 0 by simp
with (\exists i'' \neq 0) have a = b by (simp add: algebra-simps)
hence b - a = 0 *R (c - a) by simp
moreover have c - a = 1 *R (c - a) by simp
ultimately have \exists x t s. b - a = t *R x \land c - a = s *R x
  by blast
thus dep2 (b - a) (c - a) unfolding dep2-def .
next
assume \exists k'' \neq 0
from (\exists i'' + \exists j'' + \exists k'' = 0) have \exists i'' = -(\exists j'' + \exists k'') by simp
with (\exists i'' *R a + \exists j'' *R b + \exists k'' *R c = 0)
have -(\exists j'' + \exists k'') *R a + \exists j'' *R b + \exists k'' *R c = 0 by simp
hence \exists k'' *R (c - a) = -(\exists j'' *R (b - a))
  by (simp add: scaleR-left-distrib
    scaleR-right-diff-distrib
    scaleR-left-diff-distrib
    algebra-simps)
hence (1 / \exists k'') *R \exists k'' *R (c - a) = -(\exists j'' / \exists k'') *R (b - a)
  by simp
with (\exists k'' \neq 0) have c - a = -(\exists j'' / \exists k'') *R (b - a) by simp
moreover have b - a = 1 *R (b - a) by simp
ultimately have \exists x t s. b - a = t *R x \land c - a = s *R x by blast
thus dep2 (b - a) (c - a) unfolding dep2-def .
qed
with Col-dep2 show real-euclid.Col a b c by auto
}

{ assume real-euclid.Col a b c
with Col-dep2 have dep2 (b - a) (c - a) by auto
then obtain x and t and s where b - a = t *R x and c - a = s *R x
  unfolding dep2-def
  by auto
show proj2-Col \exists p \exists q \exists r
proof cases
  assume t = 0
  with (b - a = t *R x) have a = b by simp
  with proj2-Col-coincide show proj2-Col \exists p \exists q \exists r by simp
next
  assume t \neq 0

  from (b - a = t *R x) and (c - a = s *R x)
  have s *R (b - a) = t *R (c - a) by simp
  hence (s - t) *R a + (-s) *R b + t *R c = 0
    by (simp add: scaleR-right-diff-distrib
      scaleR-left-diff-distrib
      algebra-simps
      scaleR-left-diff-distrib
      scaleR-right-diff-distrib
      scaleR-left-distrib
      scaleR-right-distrib
      algebra-simps)
  thus proj2-Col (b - a) (c - a) unfolding proj2-Col-def .
qed
with Col-dep2 show real-euclid.Col a b c by auto

101
algebra-simps)

hence \((s-t) \cdot R \cdot a' + (-s) \cdot R \cdot b' + t \cdot R \cdot c'\)$1 = 0  
and \((s-t) \cdot R \cdot a' + (-s) \cdot R \cdot b' + t \cdot R \cdot c'\)$2 = 0  
unfolding vector2-append1-def and vector-def  
by (simp-all add: vec-eq-iff)

moreover have \((s-t) \cdot R \cdot a' + (-s) \cdot R \cdot b' + t \cdot R \cdot c'\)$3 = 0  
unfolding vector2-append1-def and vector-def  
by simp

ultimately have \((s-t) \cdot R \cdot a' + (-s) \cdot R \cdot b' + t \cdot R \cdot c'\) = 0  
by (simp add: vec-eq-iff forall-3)

with \(\langle a' = (1/i) \cdot R \cdot a'' \rangle\)  
and \(\langle b' = (1/j) \cdot R \cdot b'' \rangle\)  
and \(\langle c' = (1/k) \cdot R \cdot c'' \rangle\)  
have \((s-t)/i \cdot R \cdot a'' + (-s/j) \cdot R \cdot b'' + (t/k) \cdot R \cdot c'' = 0\)  
by simp

moreover from \(t \neq 0\) and \(k \neq 0\) have \(t/k \neq 0\) by simp

ultimately show proj2-Col ?a ?b ?c  
unfolding proj2-Col-def  
by blast

qed

lemma proj2-Col-iff-euclid-cart2:

assumes z-non-zero p and z-non-zero q and z-non-zero r  
shows proj2-Col p q r \(\iff\) real-euclid.Col (cart2-pt p) (cart2-pt q) (cart2-pt r)  
(is - \(\iff\) real-euclid.Col ?a ?b ?c)

proof –

from \(\langle z\text{-non-zero } p \rangle\) and \(\langle z\text{-non-zero } q \rangle\) and \(\langle z\text{-non-zero } r \rangle\)  
have proj2-pt ?a = p and proj2-pt ?b = q and proj2-pt ?c = r  
by (simp-all add: proj2-cart2)

with proj2-Col-iff-euclid [of ?a ?b ?c]  
show proj2-Col p q r \(\iff\) real-euclid.Col ?a ?b ?c by simp

qed

lemma euclid-Col-cart2-incident:

assumes z-non-zero p and z-non-zero q and z-non-zero r and \(p \neq q\)  
and proj2-incident p l and proj2-incident q l  
and real-euclid.Col (cart2-pt p) (cart2-pt q) (cart2-pt r)  
(is real-euclid.Col ?cp ?cq ?cr)

shows proj2-incident r l

proof –

from \(\langle z\text{-non-zero } p \rangle\) and \(\langle z\text{-non-zero } q \rangle\) and \(\langle z\text{-non-zero } r \rangle\)  
and \(\langle \text{real-euclid.Col } ?cp ?cq ?cr \rangle\)  
have proj2-Col p q r by (subst proj2-Col-iff-euclid-cart2, simp-all)

hence proj2-set-Col \(\{p,q,r\}\) by (simp add: proj2-Col-iff-set-Col)

then obtain m where proj2-incident p m and proj2-incident q m and proj2-incident r m

102
by (unfold proj2-set-Col-def, auto)

from (p ≠ q) and (proj2-incident p l) and (proj2-incident q l)
and (proj2-incident p m) and (proj2-incident q m) and proj2-incident-unique
have l = m by auto
with (proj2-incident r m) show proj2-incident r l by simp

qed

lemma euclid-B-cart2-common-line:
assumes z-non-zero p and z-non-zero q and z-non-zero r
and B_R (cart2-pt p) (cart2-pt q) (cart2-pt r)
(is B_R ?cp ?cq ?cr)
sows ∃ l. proj2-incident p l ∧ proj2-incident q l ∧ proj2-incident r l
proof −
from (z-non-zero p) and (z-non-zero q) and (z-non-zero r)
and (B_R ?cp ?cq ?cr) and proj2-Col-iff-euclid-cart2
have proj2-Col p q r by (unfold real-euclid.Col-def) simp
hence proj2-set-Col {p,q,r} by (simp add: proj2-Col-iff-set-Col)
thus ∃ l. proj2-incident p l ∧ proj2-incident q l ∧ proj2-incident r l
by (unfold proj2-set-Col-def) simp

qed

lemma cart2-append1-between:
assumes z-non-zero p and z-non-zero q and z-non-zero r
shows B_R (cart2-pt p) (cart2-pt q) (cart2-pt r)
←→ (∃ k ≥ 0. k ≤ 1 ∧ cart2-append1 q = k *_R cart2-append1 r + (1 - k) *_R cart2-append1 p)
proof −
let ?cp = cart2-pt p
let ?cq = cart2-pt q
let ?cr = cart2-pt r
let ?cp1 = vector2-append1 ?cp
let ?cq1 = vector2-append1 ?cq
let ?cr1 = vector2-append1 ?cr
from (z-non-zero p) and (z-non-zero q) and (z-non-zero r)
have ?cp1 = cart2-append1 p
and ?cq1 = cart2-append1 q
and ?cr1 = cart2-append1 r
by (simp-all add: cart2-append1)

have ∀ k. ?cq - ?cp = k *_R (?cr - ?cp) ←→ ?cq = k *_R ?cr + (1 - k) *_R ?cp
by (simp add: algebra-simps)
hence ∀ k. ?cq - ?cp = k *_R (?cr - ?cp)
←→ ?cq1 = k *_R ?cr1 + (1 - k) *_R ?cp1
unfolding vector2-append1-def and vector-def
by (simp add: vec-eq-iff forall-2 forall-3)
with (?cp1 = cart2-append1 p)
and (?cq1 = cart2-append1 q)
and \( \langle ?cr1 = \text{cart2-append1 } r \rangle \)

have \( \forall k. \ ?cq - ?cp = k \ast R (\ ?cr - ?cp) \)

\[ \iff \text{cart2-append1 } q = k \ast R \text{cart2-append1 } r + (1 - k) \ast R \text{cart2-append1 } p \]

by simp

thus \( B_R (\text{cart2-pt } p) (\text{cart2-pt } q) (\text{cart2-pt } r) \)

\[ \iff (\exists k \geq 0. \ k < 1) \land \text{cart2-append1 } q = k \ast R \text{cart2-append1 } r + (1 - k) \ast R \text{cart2-append1 } p \]

by \( \text{(unfold real-euclid-B-def) simp} \)

qed

lemma \text{cart2-append1-between-right-strict}:

assumes \( \text{z-non-zero } p \) and \( \text{z-non-zero } q \) and \( \text{z-non-zero } r \)

and \( B_R (\text{cart2-pt } p) (\text{cart2-pt } q) (\text{cart2-pt } r) \)

and \( q \neq r \)

shows \( \exists k \geq 0, k < 1 \land \text{cart2-append1 } q = k \ast R \text{cart2-append1 } r + (1 - k) \ast R \text{cart2-append1 } p \)

proof

from \( \langle \text{z-non-zero } p \rangle \) and \( \langle \text{z-non-zero } q \rangle \) and \( \langle \text{z-non-zero } r \rangle \)

and \( \langle B_R (\text{cart2-pt } p) (\text{cart2-pt } q) (\text{cart2-pt } r): \text{cart2-append1-between} \rangle \)

obtain \( k \) where \( k \geq 0 \) and \( k < 1 \)

and \( \text{cart2-append1 } q = k \ast R \text{cart2-append1 } r + (1 - k) \ast R \text{cart2-append1 } p \)

by auto

have \( k \neq 1 \)

proof

assume \( k = 1 \)

with \( \langle \text{cart2-append1 } q = k \ast R \text{cart2-append1 } r + (1 - k) \ast R \text{cart2-append1 } p \rangle \)

have \( \text{cart2-append1 } q = \text{cart2-append1 } r \) by simp

with \( \langle \text{z-non-zero } q: \text{have } q = r \text{ by (rule cart2-append1-inj) \rangle} \)

with \( \langle q \neq r: \text{show False ..} \rangle \)

qed

with \( \langle k \leq 1 \rangle \) have \( k < 1 \) by simp

with \( \langle k \geq 0 \rangle \)

and \( \langle \text{cart2-append1 } q = k \ast R \text{cart2-append1 } r + (1 - k) \ast R \text{cart2-append1 } p \rangle \)

show \( \exists k \geq 0, k < 1 \land \text{cart2-append1 } q = k \ast R \text{cart2-append1 } r + (1 - k) \ast R \text{cart2-append1 } p \)

by \( \text{(simp add: exI [of - k])} \)

qed

lemma \text{cart2-append1-between-strict}:

assumes \( \text{z-non-zero } p \) and \( \text{z-non-zero } q \) and \( \text{z-non-zero } r \)

and \( B_R (\text{cart2-pt } p) (\text{cart2-pt } q) (\text{cart2-pt } r) \)

and \( q \neq p \) and \( q \neq r \)

shows \( \exists k \geq 0, k < 1 \land \text{cart2-append1 } q = k \ast R \text{cart2-append1 } r + (1 - k) \ast R \text{cart2-append1 } p \)

proof

from \( \langle \text{z-non-zero } p \rangle \) and \( \langle \text{z-non-zero } q \rangle \) and \( \langle \text{z-non-zero } r \rangle \)

and \( \langle B_R (\text{cart2-pt } p) (\text{cart2-pt } q) (\text{cart2-pt } r): \text{cart2-append1-between-right-strict} [\text{of } p q r] \rangle \)

obtain \( k \) where \( k \geq 0 \) and \( k < 1 \)

and \( \text{cart2-append1 } q = k \ast R \text{cart2-append1 } r + (1 - k) \ast R \text{cart2-append1 } p \)
by auto

have \( k \neq 0 \)
proof
  assume \( k = 0 \)
  with \( \langle \text{cart2-append1 } q = k \ast_R \text{cart2-append1 } r + (1 - k) \ast_R \text{cart2-append1 } p \rangle \)
  have \( \text{cart2-append1 } q = \text{cart2-append1 } p \) by simp
  with \( \langle \neg\text{non-zero } q \rangle \) have \( q = p \) by (rule cart2-append1-inj)
  with \( \langle q \neq p \rangle \) show False ..
qed
with \( \langle k \geq 0 \rangle \) have \( k > 0 \) by simp
with \( \langle k < 1 \rangle \)
  and \( \langle \text{cart2-append1 } q = k \ast_R \text{cart2-append1 } r + (1 - k) \ast_R \text{cart2-append1 } p \rangle \)
  show \( \exists k > 0. k < 1 \)
  \( \wedge \text{cart2-append1 } q = k \ast_R \text{cart2-append1 } r + (1 - k) \ast_R \text{cart2-append1 } p \)
  by (simp add: exI [of - k])
qed

8 The hyperbolic plane and Tarski’s axioms

theory Hyperbolic-Tarski
imports Euclid-Tarski
Projective
HOL-Library.Quadratic-Discriminant
begin

8.1 Characterizing a specific conic in the projective plane

definition \( M : real^{3 \times 3} \) where
  \( M \triangleq \text{vector } [1, 0, 0], \text{vector } [0, 1, 0], \text{vector } [0, 0, -1]] \)

lemma \( M\text{-symmatrix} : \text{symmatrix } M \)
  unfolding symmatrix-def and transpose-def and M-def
  by (simp add: vec-eq-iff forall-3 vector-3)

lemma \( M\text{-self-inverse} : M \ast\ast M = \text{mat } 1 \)
  unfolding M-def and matrix-matrix-mult-def and mat-def and vector-def
  by (simp add: sum-3 vec-eq-iff forall-3)

lemma \( M\text{-invertible} : \text{invertible } M \)
  unfolding invertible-def
  using M-self-inverse
  by auto
**definition** polar :: proj2 ⇒ proj2-line where
polar p ≜ proj2-line-abs (M *v proj2-rep p)

**definition** pole :: proj2-line ⇒ proj2 where
pole l ≜ proj2-abs (M *v proj2-line-rep l)

**lemma** polar-abs:
assumes v ≠ 0
shows polar (proj2-abs v) = proj2-line-abs (M *v v)

**proof**
- from ⟨v ≠ 0⟩ and proj2-rep-abs2 obtain k where k ≠ 0 and proj2-rep (proj2-abs v) = k *R v by auto
- have polar (proj2-abs v) = proj2-line-abs (k *R (M *v v)) unfolding polar-def
  by (simp add: matrix-scaleR-vector-ac scaleR-matrix-vector-assoc)
- with ⟨k ≠ 0⟩ and proj2-line-abs-mult
- show polar (proj2-abs v) = proj2-line-abs (M *v v) by simp

qed

**lemma** pole-abs:
assumes v ≠ 0
shows pole (proj2-line-abs v) = proj2-abs (M *v v)

**proof**
- from ⟨v ≠ 0⟩ and proj2-line-rep-abs obtain k where k ≠ 0 and proj2-line-rep (proj2-line-abs v) = k *R v by auto
- from ⟨proj2-line-rep (proj2-line-abs v) = k *R v⟩
  have pole (proj2-line-abs v) = proj2-abs (k *R (M *v v)) unfolding pole-def
  by (simp add: matrix-scaleR-vector-ac scaleR-matrix-vector-assoc)
- with ⟨k ≠ 0⟩ and proj2-abs-mult
- show pole (proj2-line-abs v) = proj2-abs (M *v v) by simp

qed

**lemma** polar-rep-non-zero: M *v proj2-rep p ≠ 0

**proof**
- have proj2-rep p ≠ 0 by (rule proj2-rep-non-zero)
- with M-invertible
- show M *v proj2-rep p ≠ 0 by (rule invertible-times-non-zero)

qed

**lemma** pole-polar: pole (polar p) = p

**proof**
- from polar-rep-non-zero
- have pole (polar p) = proj2-abs (M *v (M *v proj2-rep p)) unfolding pole-def
  by (rule pole-abs)
- with M-self-inverse
show pole (polar p) = p
by (simp add: matrix-vector-mul-assoc proj2-abs-rep)
qed

lemma pole-rep-non-zero: M *v proj2-line-rep l ≠ 0
proof –
  have proj2-line-rep l ≠ 0 by (rule proj2-line-rep-non-zero)
  with M-invertible
  show M *v proj2-line-rep l ≠ 0 by (rule invertible-times-non-zero)
qed

lemma polar-pole: polar (pole l) = l
proof –
  from pole-rep-non-zero
  have polar (pole l) = proj2-line-abs (M *v (M *v proj2-line-rep l))
  unfolding pole-def
  by (rule polar-abs)
  with M-self-inverse
  show polar (pole l) = l
  by (simp add: matrix-vector-mul-assoc proj2-line-abs-rep
          matrix-vector-mul-lid)
qed

lemma polar-inj:
  assumes polar p = polar q
  shows p = q
proof –
  from ⟨polar p = polar q⟩
  have pole (polar p) = pole (polar q)
  by simp
  thus p = q
  by (simp add: pole-polar)
qed

definition conic-sgn :: proj2 ⇒ real where
  conic-sgn p ≜ sgn (proj2-rep p · (M *v proj2-rep p))

lemma conic-sgn-abs:
  assumes v ≠ 0
  shows conic-sgn (proj2-abs v) = sgn (v · (M *v v))
proof –
  from ⟨v ≠ 0⟩
  obtain j where j ≠ 0
  and proj2-rep-abs2
  obtain j where j ≠ 0
  and proj2-rep (proj2-abs v) = j *R v
  by auto
  have conic-sgn (proj2-abs v) = sgn (j^2 * (v · (M *v v)))
  unfolding conic-sgn-def
  by (simp add:
          matrix-scaleR-vector-ac
          scaleR-matrix-vector-assoc [symmetric]
          dot-scaleR-mult
          power2-eq-square)
also have \( \ldots = \text{sgn} \left( j^2 \right) \ast \text{sgn} \left( v \cdot (M \ast v) \right) \)
by (rule sgn-mult)
also from \( j \neq 0 \), have \( \ldots = \text{sgn} \left( v \cdot (M \ast v) \right) \)
by (simp add: power2-eq-square sgn-mult)
finally show \( \text{conic-sgn} \left( \text{proj2-abs} \ v \right) = \text{sgn} \left( v \cdot (M \ast v) \right) \).

qed

lemma sgn-conic-sgn: \( \text{sgn} \left( \text{conic-sgn} \ p \right) = \text{conic-sgn} \ p \)
by (unfold conic-sgn-def) simp

definition \( S :: \text{proj2 set where} \)
\( S \triangleq \{ p. \text{conic-sgn} \ p = 0 \} \)

definition \( K2 :: \text{proj2 set where} \)
\( K2 \triangleq \{ p. \text{conic-sgn} \ p < 0 \} \)

lemma S-K2-empty: \( S \cap K2 = \{ \} \)
unfolding S-def and K2-def by auto

lemma K2-abs:
assumes \( v \neq 0 \)
shows \( \text{proj2-abs} \ v \in K2 \leftrightarrow v \cdot (M \ast v) < 0 \)
proof –
have \( \text{proj2-abs} \ v \in K2 \leftrightarrow \text{conic-sgn} \left( \text{proj2-abs} \ v \right) < 0 \)
by (simp add: K2-def)
with \( (v \neq 0) \) and conic-sgn-abs
show \( \text{proj2-abs} \ v \in K2 \leftrightarrow v \cdot (M \ast v) < 0 \) by simp
qed

definition \( K2\text{-centre} = \text{proj2-abs} \left( \text{vector} \ [0,0,1] \right) \)

lemma K2-centre-non-zero: \( [0,0,1] \neq (0 :: \text{real}^3) \)
by (unfold vector-def) (simp add: vec-eq-iff forall-3)

lemma K2-centre-in-K2: \( K2\text{-centre} \in K2 \)
proof –
from K2-centre-non-zero and proj2-rep-absolute2
obtain \( k \) where \( k \neq 0 \) and \( \text{proj2-rep} \text{ K2\text{-centre}} = k \ast_R \text{vector} \ [0,0,1] \)
by (unfold K2-centre-def) auto
from \( (k \neq 0) \) have \( 0 < k^2 \) by simp
with \( \text{proj2-rep} \text{ K2\text{-centre}} = k \ast_R \text{vector} \ [0,0,1] \)
show \( K2\text{-centre} \in K2 \)
unfolding K2-def
and conic-sgn-def
and M-def
and matrix-vector-mult-def
and inner-vec-def
and vector-def
by (simp add: vec-eq-iff sum-3 power2-eq-square) qed

lemma K2-imp-M-neg:
  assumes v ≠ 0 and proj2-abs v ∈ K2
  shows v · (M * v v) < 0
  using assms
  by (simp add: K2-abs)

lemma M-neg-imp-z-squared-big:
  assumes v · (M * v v) < 0
  shows (v$3)^2 > (v$1)^2 + (v$2)^2
  using ⟨v · (M * v v) < 0⟩ unfolding matrix-vector-mult-def and M-def and vector-def
  by (simp add: inner-vec-def sum-3 power2-eq-square)

lemma M-neg-imp-z-non-zero:
  assumes v · (M * v v) < 0
  shows v$3 ≠ 0
  proof −
    have (v$1)^2 + (v$2)^2 ≥ 0 by simp
    with M-neg-imp-z-squared-big [of v] and ⟨v · (M * v v) < 0⟩
    have (v$3)^2 > 0 by arith
    thus v$3 ≠ 0 by simp
  qed

lemma M-neg-imp-K2:
  assumes v · (M * v v) < 0
  shows proj2-abs v ∈ K2
  proof −
    from ⟨v · (M * v v) < 0⟩ have v$3 ≠ 0 by (rule M-neg-imp-z-non-zero)
    hence v ≠ 0 by auto
    with ⟨v · (M * v v) < 0⟩ and K2-abs show proj2-abs v ∈ K2 by simp
  qed

lemma M-reverse: a · (M * v b) = b · (M * v a)
  unfolding matrix-vector-mult-def and M-def and vector-def
  by (simp add: inner-vec-def sum-3)

lemma S-abs:
  assumes v ≠ 0
  shows proj2-abs v ∈ S −→ v · (M * v v) = 0
  proof −
    have proj2-abs v ∈ S −→ conic-sgn (proj2-abs v) = 0
      unfolding S-def
      by simp
    also from ⟨v ≠ 0⟩ and conic-sgn-abs
    have … −→ sgn (v · (M * v v)) = 0 by simp
    finally show proj2-abs v ∈ S −→ v · (M * v v) = 0 by (simp add: sgn-0-0)
qed

lemma $S$-alt-def: $p \in S \iff \text{proj}_2\text{-rep } p \cdot (M * v \text{proj}_2\text{-rep } p) = 0$
proof
have $\text{proj}_2\text{-rep } p \neq 0$ by (rule $\text{proj}_2\text{-rep\-non\-zero}$)
hence $\text{proj}_2\text{-abs} (\text{proj}_2\text{-rep } p) \in S \iff \text{proj}_2\text{-rep } p \cdot (M * v \text{proj}_2\text{-rep } p) = 0$
by (rule $S$-abs)
thus $p \in S \iff \text{proj}_2\text{-rep } p \cdot (M * v \text{proj}_2\text{-rep } p) = 0$
by (simp add: $\text{proj}_2\text{-abs\-rep}$)
qed

lemma incident-polar:
$\text{proj}_2\text{-incident } p (\text{polar } q) \iff \text{proj}_2\text{-rep } p \cdot (M * v \text{proj}_2\text{-rep } q) = 0$
using $\text{polar\-rep\-non\-zero}$
unfolding $\text{polar\-def}$
by (rule $\text{proj}_2\text{-incident\-right\-abs}$)

lemma incident-own-polar-in-$S$: $\text{proj}_2\text{-incident } p (\text{polar } p) \iff p \in S$
using incident-polar and $S$-alt-def
by simp

lemma incident-polar-swap:
assumes $\text{proj}_2\text{-incident } p (\text{polar } q)$
shows $\text{proj}_2\text{-incident } q (\text{polar } p)$
proof
from $\langle \text{proj}_2\text{-incident } p (\text{polar } q) \rangle$
have $\text{proj}_2\text{-rep } p \cdot (M * v \text{proj}_2\text{-rep } q) = 0$ by (unfold incident-polar)
hence $\text{proj}_2\text{-rep } q \cdot (M * v \text{proj}_2\text{-rep } p) = 0$ by (simp add: $M$-reverse)
thus $\text{proj}_2\text{-incident } q (\text{polar } p)$ by (unfold incident-polar)
qed

lemma incident-pole-polar:
assumes $\text{proj}_2\text{-incident } p l$
shows $\text{proj}_2\text{-incident } (\text{pole } l) (\text{polar } p)$
proof
from $\langle \text{proj}_2\text{-incident } p l \rangle$
have $\text{proj}_2\text{-incident } p (\text{pole } l) (\text{polar } p)$ by (subst polar-pole)
thus $\text{proj}_2\text{-incident } (\text{pole } l) (\text{polar } p)$ by (rule incident-polar-swap)
qed

definition $z$-zero :: $\text{proj}_2\text{-line}$ where
$z$-zero $\triangleq \text{proj}_2\text{-line\-abs} (\text{vector } [0,0,1])$

lemma $z$-zero:
assumes $\langle \text{proj}_2\text{-rep } p \rangle S3 = 0$
shows $\text{proj}_2\text{-incident } p z$-zero
proof
from $K2$-centre\-non\-zero and $\text{proj}_2\text{-rep\-abs}$
obtain $k$ where $\text{proj}_2\text{-line\-rep } z$-zero $= k * R \text{ vector } [0,0,1]$
by (unfold z-zero-def) auto
with ((proj2-rep p)$3 = 0)
show proj2-incident p z-zero
  unfolding proj2-incident-def and inner-vec-def and vector-def
  by (simp add: sum-3)
qed

lemma z-zero-conic-sgn-1:
  assumes proj2-incident p z-zero
  shows conic-sgn p = 1
proof
  let ?v = proj2-rep p
  have (vector $[0,0,1]$ :: real$^3$) ≠ 0
    unfolding vector-def
    by (simp add: vec-eq-iff)
  with (proj2-incident p z-zero)
  have ?v · vector $[0,0,1]$ = 0
    unfolding z-zero-def
    by (simp add: proj2-incident-right-abs)
  hence ?v$3 = 0
    unfolding inner-vec-def and vector-def
    by (simp add: sum-3)
  hence ?v · (M * ?v ?v) = (?v$1)^2 + (?v$2)^2
    unfolding inner-vec-def
    and power2-eq-square
    and matrix-vector-mult-def
    and M-def
    and vector-def
    and sum-3
    by simp
  have ?v ≠ 0 by (rule proj2-rep-non-zero)
  with (?v$3 = 0) have ?v$1 ≠ 0 ∨ ?v$2 ≠ 0 by (simp add: vec-eq-iff forall-3)
  hence (?v$1)^2 > 0 ∨ (?v$2)^2 > 0 by simp
  with add-sign-intros [of (?v$1)^2 (?v$2)^2]
  have (?v$1)^2 + (?v$2)^2 > 0 by auto
  with (?v · (M * ?v ?v) = (?v$1)^2 + (?v$2)^2)
  have ?v · (M * ?v ?v) > 0 by simp
  thus conic-sgn p = 1
    unfolding conic-sgn-def
    by simp
qed

lemma conic-sgn-not-1-z-non-zero:
  assumes conic-sgn p ≠ 1
  shows z-non-zero p
proof
  from (conic-sgn p ≠ 1)
  have ¬ proj2-incident p z-zero by (auto simp add: z-zero-conic-sgn-1)
thus z-non-zero p by (auto simp add: z-zero)

qed

lemma z-zero-not-in-S:
  assumes proj2-incident p z-zero
  shows p /∈ S
proof
  from ⟨proj2-incident p z-zero⟩ have conic-sgn p = 1
  by (rule z-zero-conic-sgn-1)
  thus p /∈ S
  unfolding S-def
  by simp

qed

lemma line-incident-point-not-in-S:
  ∃ p. p /∈ S ∧ proj2-incident p l
proof
  let ?p = proj2-intersection l z-zero
  have proj2-incident ?p l and proj2-incident ?p z-zero
  by (rule proj2-intersection-incident)+
  from ⟨proj2-incident ?p z-zero⟩ have ?p /∈ S by (rule z-zero-not-in-S)
  with ⟨proj2-incident ?p l⟩ show ∃ p. p /∈ S ∧ proj2-incident p l by auto

qed

lemma apply-cltn2-abs-abs-in-S:
  assumes v ≠ 0 and invertible J
  shows apply-cltn2 (proj2-abs v) (cltn2-abs J) ∈ S
  ⇔ v · (J ** M ** transpose J * v v) = 0
proof
  from ⟨v ≠ 0⟩ and ⟨invertible J⟩
  have v v* J ≠ 0 by (rule non-zero-mult-invertible-non-zero)
  from ⟨v ≠ 0⟩ and ⟨invertible J⟩
  have apply-cltn2 (proj2-abs v) (cltn2-abs J) = proj2-abs (v v* J)
  by (rule apply-cltn2-abs)
  also from ⟨v v* J ≠ 0⟩
  have ... ∈ S ⇐ (v v* J) · (M * v v* J) = 0 by (rule S-abs)
  finally show apply-cltn2 (proj2-abs v) (cltn2-abs J) ∈ S
  ⇐ v · (J ** M ** transpose J * v v) = 0
  by (simp add: dot-lmul-matrix matrix-vector-mul-assoc [symmetric])

qed

lemma apply-cltn2-right-abs-in-S:
  assumes invertible J
  shows apply-cltn2 p (cltn2-abs J) ∈ S
  ⇐ (proj2-rep p) · (J ** M ** transpose J * v (proj2-rep p)) = 0
proof
  have proj2-rep p ≠ 0 by (rule proj2-rep-non-zero)
  with ⟨invertible J⟩
have \( \text{apply-cltn2} \ (\text{proj2-abs} \ (\text{proj2-rep} \ p)) \ (\text{cltn2-abs} \ J) \in S \)
\( \iff \text{proj2-rep} \ p \cdot (J \ * \ M \ * \ \text{transpose} \ J \ * \ \text{proj2-rep} \ p) = 0 \)
by (simp add: apply-cltn2-abs-abs-in-S)

thus \( \text{apply-cltn2} \ p \ (\text{cltn2-abs} \ J) \in S \)
\( \iff \text{proj2-rep} \ p \cdot (J \ * \ M \ * \ \text{transpose} \ J \ * \ \text{proj2-rep} \ p) = 0 \)
by (simp add: proj2-abs-rep)

qed

lemma \( \text{apply-cltn2-abs-in-S} \):
assumes \( v \neq 0 \)
shows \( \text{apply-cltn2} \ (\text{proj2-abs} \ v) \ C \in S \)
\( \iff v \cdot (\text{cltn2-rep} \ C \ * \ M \ * \ \text{transpose} \ (\text{cltn2-rep} \ C) \ * \ v \ \text{proj2-rep} \ p) = 0 \)

proof –

have invertible \( (\text{cltn2-rep} \ C) \)
by (rule cltn2-rep-invertible)

with \( \langle v \neq 0 \rangle \)

have \( \text{apply-cltn2} \ (\text{proj2-abs} \ v) \ (\text{cltn2-abs} \ (\text{cltn2-rep} \ C)) \in S \)
\( \iff v \cdot (\text{cltn2-rep} \ C \ * \ M \ * \ \text{transpose} \ (\text{cltn2-rep} \ C) \ * \ v \ \text{proj2-rep} \ p) = 0 \)
by (rule apply-cltn2-abs-abs-in-S)

thus \( \text{apply-cltn2} \ (\text{proj2-abs} \ v) \ C \in S \)
\( \iff v \cdot (\text{cltn2-rep} \ C \ * \ M \ * \ \text{transpose} \ (\text{cltn2-rep} \ C) \ * \ v \ \text{proj2-rep} \ p) = 0 \)
by (simp add: cltn2-abs-rep)

qed

lemma \( \text{apply-cltn2-in-S} \):
\( \text{apply-cltn2} \ p \ C \in S \)
\( \iff \text{proj2-rep} \ p \cdot (\text{cltn2-rep} \ C \ * \ M \ * \ \text{transpose} \ (\text{cltn2-rep} \ C) \ * \ v \ \text{proj2-rep} \ p) = 0 \)

proof –

have \( \text{proj2-rep} \ p \neq 0 \)
by (rule proj2-rep-non-zero)

hence \( \text{apply-cltn2} \ (\text{proj2-obs} \ (\text{proj2-rep} \ p)) \ C \in S \)
\( \iff \text{proj2-rep} \ p \cdot (\text{cltn2-rep} \ C \ * \ M \ * \ \text{transpose} \ (\text{cltn2-rep} \ C) \ * \ v \ \text{proj2-rep} \ p) = 0 \)
by (rule apply-cltn2-abs-abs-in-S)

thus \( \text{apply-cltn2} \ p \ C \in S \)
\( \iff \text{proj2-rep} \ p \cdot (\text{cltn2-rep} \ C \ * \ M \ * \ \text{transpose} \ (\text{cltn2-rep} \ C) \ * \ v \ \text{proj2-rep} \ p) = 0 \)
by (simp add: proj2-abs-rep)

qed

lemma \( \text{norm-M} \): \( \text{(vector2-append1} \ v) \cdot (M \ * \ \text{vector2-append1} \ v) = (\text{norm} \ v)^2 - 1 \)

proof –

have \( (\text{norm} \ v)^2 = (v \ 1)^2 + (v \ 2)^2 \)
unfolding \( \text{norm-vec-def} \)
and \( \text{L2-set-def} \)
by (simp add: sum-2)

thus \( \text{(vector2-append1} \ v) \cdot (M \ * \ \text{vector2-append1} \ v) = (\text{norm} \ v)^2 - 1 \)
unfolding \( \text{vector2-append1-def} \)
and \( \text{inner-vec-def} \)

113
and matrix-vector-mult-def
and vector-def
and M-def
and power2-norm-eq-inner
by (simp add: sum-3 power2-eq-square)

qed

8.2 Some specific points and lines of the projective plane
definition east = proj2-abs (vector [1,0,1])
definition west = proj2-abs (vector [-1,0,1])
definition north = proj2-abs (vector [0,1,1])
definition south = proj2-abs (vector [0,-1,1])
definition far-north = proj2-abs (vector [0,0,1])

lemmas compass-defs = east-def west-def north-def south-def

lemma compass-non-zero:
  shows vector [1,0,1] ≠ (0 :: real ^ 3)
  and vector [-1,0,1] ≠ (0 :: real ^ 3)
  and vector [0,1,1] ≠ (0 :: real ^ 3)
  and vector [0,-1,1] ≠ (0 :: real ^ 3)
  and vector [0,0,1] ≠ (0 :: real ^ 3)
  unfolding vector-def
  by (simp-all add: vec-eq-iff forall-3)

lemma east-west-distinct: east ≠ west
proof
  assume east = west
  with compass-non-zero
  and proj2-abs-ads-mu [of vector [1,0,1] vector [-1,0,1]]
  obtain k where (vector [1,0,1] :: real ^ 3) = k *R vector [-1,0,1]
  unfolding compass-defs
  by auto
  thus False
  unfolding vector-def
  by (auto simp add: vec-eq-iff forall-3)
qed

lemma north-south-distinct: north ≠ south
proof
  assume north = south
  with compass-non-zero
  and proj2-abs-ads-mu [of vector [0,1,1] vector [0,-1,1]]
  obtain k where (vector [0,1,1] :: real ^ 3) = k *R vector [0,-1,1]
  unfolding compass-defs
  by auto
  thus False
unfolding vector-def
by (auto simp add: vec-eq-iff forall-3)
qed

lemma north-not-east-or-west: north \notin \{east, west\}
proof
assume north \in \{east, west\}
hence east = north \lor west = north by auto
with compass-non-zero
and proj2-args-abs-abs-mult [of - vector [0,1,1]]
obtain k where (vector [1,0,1] :: real^3) = k *R vector [0,1,1]
\lor (vector [-1,0,1] :: real^3) = k *R vector [0,1,1]
unfolding compass-defs
by auto
thus False
unfolding vector-def
by (simp add: vec-eq-iff forall-3)
qed

lemma compass-in-S:
shows east \in S and west \in S and north \in S and south \in S
using compass-non-zero and S-abs
unfolding compass-defs
and M-def
and inner-vec-def
and matrix-vector-mult-def
and vector-def
by (simp-all add: sum-3)

lemma east-west-tangents:
shows polar east = proj2-line-abs (vector [-1,0,1])
and polar west = proj2-line-abs (vector [1,0,1])
proof
have M *v vector [1,0,1] = (1) *R vector [-1,0,1]
and M *v vector [-1,0,1] = (-1) *R vector [1,0,1]
unfolding M-def and matrix-vector-mult-def and vec-def
by (simp-all add: vec-eq-iff sum-3)
with compass-non-zero and polar-abs
have polar east = proj2-line-abs ((1) *R vector [-1,0,1])
and polar west = proj2-line-abs ((-1) *R vector [1,0,1])
unfolding compass-defs
by simp-all
with proj2-line-abs-mult [of -1]
show polar east = proj2-line-abs (vector [-1,0,1])
and polar west = proj2-line-abs (vector [1,0,1])
by simp-all
qed

lemma east-west-tangents-distinct: polar east \neq polar west

115
proof
  assume polar east = polar west
  hence east = west by (rule polar-inj)
  with east-west-distinct show False ..
qed

lemma east-west-tangents-incident-far-north:
  shows proj2-incident far-north (polar east)
  and proj2-incident far-north (polar west)
  using compass-non-zero and proj2-incident-abs
  unfolding far-north-def and east-west-tangents and inner-vec-def
  by (simp-all add: sum-3 vector-3)

lemma east-west-tangents-far-north:
  proj2-intersection (polar east) (polar west) = far-north
  using east-west-tangents-distinct and east-west-tangents-incident-far-north
  by (rule proj2-intersection-unique [symmetric])

instantiation proj2 :: zero
begin
definition proj2-zero-def: 0 = proj2-pt 0
instance ..
end

definition equator ≜ proj2-line-abs (vector [0,1,0])
definition meridian ≜ proj2-line-abs (vector [1,0,0])

lemma equator-meridian-distinct: equator ≠ meridian
proof
  assume equator = meridian
  with compass-non-zero
  and proj2-line-abs-abs-mult [of vector [0,1,0] vector [1,0,0]]
  obtain k where (vector [0,1,0] :: real^3) = k *R vector [1,0,0]
      by (unfold equator-def meridian-def) auto
  thus False by (unfold vector-def) (auto simp add: vec-eq-iff forall-3)
qed

lemma east-west-on-equator:
  shows proj2-incident east equator and proj2-incident west equator
  unfolding east-def and west-def and equator-def
  using compass-non-zero
  by (simp-all add: proj2-incident-abs inner-vec-def vector-def sum-3)

lemma north-far-north-distinct: north ≠ far-north
proof
  assume north = far-north
  with compass-non-zero
  and proj2-abs-abs-mult [of vector [0,1,1] vector [0,1,0]]
  obtain k where (vector [0,1,1] :: real^3) = k *R vector [0,1,0]
by (unfold north-def far-north-def) auto
thus False
  unfolding vector-def
  by (auto simp add: vec-eq-iff forall-3)
qed

lemma north-south-far-north-on-meridian:
  shows proj2-incident north meridian and proj2-incident south meridian
  and proj2-incident far-north meridian
  unfolding compass-defs and far-north-def and meridian-def
  using compass-non-zero
  by (simp-all add: proj2-incident-abs inner-vec-def vector-def sum-3)

lemma K2-centre-on-equator-meridian:
  shows proj2-incident K2-centre equator
  and proj2-incident K2-centre meridian
  unfolding K2-centre-def and equator-def and meridian-def
  using K2-centre-non-zero and compass-non-zero
  by (simp-all add: proj2-incident-abs inner-vec-def vector-def sum-3)

lemma on-equator-meridian-is-K2-centre:
  assumes proj2-incident a equator and proj2-incident a meridian
  shows a = K2-centre
  using assms and K2-centre-on-equator-meridian and equator-meridian-distinct and proj2-incident-unique
  by auto

definition rep-equator-reflect ≜ vector [vector [1, 0, 0],
  vector [0, -1, 0],
  vector [0, 0, 1]] :: real^3^3

definition rep-meridian-reflect ≜ vector [vector [-1, 0, 0],
  vector [0, 1, 0],
  vector [0, 0, 1]] :: real^3^3

definition equator-reflect ≜ cltn2-abs rep-equator-reflect

definition meridian-reflect ≜ cltn2-abs rep-meridian-reflect

lemmas compass-reflect-defs = equator-reflect-def meridian-reflect-def
  rep-equator-reflect-def rep-meridian-reflect-def

lemma compass-reflect-self-inverse:
  shows rep-equator-reflect ** rep-equator-reflect = mat 1
  and rep-meridian-reflect ** rep-meridian-reflect = mat 1
  unfolding compass-reflect-defs matrix-matrix-mult-def mat-def
  by (simp-all add: vec-eq-iff forall-3 sum-3 vector-3)

lemma compass-reflect-invertible:
  shows invertible rep-equator-reflect and invertible rep-meridian-reflect
lemma compass-reflect-compass:
shows apply-cltn2 east meridian-reflect = west
and apply-cltn2 west meridian-reflect = east
and apply-cltn2 north meridian-reflect = north
and apply-cltn2 south meridian-reflect = south
and apply-cltn2 K2-centre meridian-reflect = K2-centre
and apply-cltn2 east equator-reflect = east
and apply-cltn2 west equator-reflect = west
and apply-cltn2 north equator-reflect = south
and apply-cltn2 south equator-reflect = north
and apply-cltn2 K2-centre equator-reflect = K2-centre
proof –
have (vector [1,0,1] :: real^3) ∗ rep-meridian-reflect = vector [-1,0,1]
and (vector [-1,0,1] :: real^3) ∗ rep-meridian-reflect = vector [1,0,1]
and (vector [0,1,1] :: real^3) ∗ rep-meridian-reflect = vector [0,1,1]
and (vector [0,-1,1] :: real^3) ∗ rep-meridian-reflect = vector [0,-1,1]
and (vector [1,0,1] :: real^3) ∗ rep-eqator-reflect = vector [1,0,1]
and (vector [-1,0,1] :: real^3) ∗ rep-eqator-reflect = vector [-1,0,1]
and (vector [0,1,1] :: real^3) ∗ rep-eqator-reflect = vector [0,1,1]
and (vector [0,-1,1] :: real^3) ∗ rep-eqator-reflect = vector [0,-1,1]
and (vector [0,0,1] :: real^3) ∗ rep-eqator-reflect = vector [0,0,1]
unfolding rep-meridian-reflect-def and rep-eqator-reflect-def
and vector-matrix-mult-def
by (simp-all add: vec-eq-iff forall-3 vector-3 sum-3)
with compass-reflect-invertible and compass-non-zero and K2-centre-non-zero
show apply-cltn2 east meridian-reflect = west
and apply-cltn2 west meridian-reflect = east
and apply-cltn2 north meridian-reflect = north
and apply-cltn2 south meridian-reflect = south
and apply-cltn2 K2-centre meridian-reflect = K2-centre
and apply-cltn2 east equator-reflect = east
and apply-cltn2 west equator-reflect = west
and apply-cltn2 north equator-reflect = south
and apply-cltn2 south equator-reflect = north
and apply-cltn2 K2-centre equator-reflect = K2-centre
unfolding compass-defs and K2-centre-def
and meridian-reflect-def and equator-reflect-def
by (simp-all add: apply-cltn2-abs)
qed

lemma on-equator-rep:
assumes z-non-zero a and proj2-incident a equator
shows ∃ x. a = proj2-abs (vector [x,0,1])
proof –
let ?ra = proj2-rep a
let ?ca1 = cart2-append1 a
let ?x = ?ca1$1
from compass-non-zero and (proj2-incident a equator)
  have ?ra · vector [0,1,0] = 0
    by (unfold equator-def) (simp add: proj2-incident-right-abs)
  hence ?ra$2 = 0 by (unfold inner-vec-def vector-def) (simp add: sum-3)
  hence ?ca1$2 = 0 by (unfold cart2-append1-def) simp
moreover
from (z-non-zero a) have ?ca1$3 = 1 by (rule cart2-append1-z)
ultimately
  have ?ca1 = vector [?x,0,1]
    by (unfold vector-def) (simp add: vec-eq-iff forall-3)
with (z-non-zero a):
  have proj2-abs (vector [?x,0,1]) = a by (simp add: proj2-abs-cart2-append1)
thus ∃ x. a = proj2-abs (vector [x,0,1]) by (simp add: exI [of - ?x])
qed

lemma on-meridian-rep:
  assumes z-non-zero a and proj2-incident a meridian
  shows ∃ y. a = proj2-abs (vector [0,y,1])
proof −
let ?ra = proj2-rep a
let ?ca1 = cart2-append1 a
let ?y = ?ca1$2
from compass-non-zero and (proj2-incident a meridian)
  have ?ra · vector [1,0,0] = 0
    by (unfold meridian-def) (simp add: proj2-incident-right-abs)
  hence ?ra$1 = 0 by (unfold inner-vec-def vector-def) (simp add: sum-3)
  hence ?ca1$1 = 0 by (unfold cart2-append1-def) simp
moreover
from (z-non-zero a) have ?ca1$3 = 1 by (rule cart2-append1-z)
ultimately
  have ?ca1 = vector [0,?y,1]
    by (unfold vector-def) (simp add: vec-eq-iff forall-3)
with (z-non-zero a):
  have proj2-abs (vector [0,?y,1]) = a by (simp add: proj2-abs-cart2-append1)
thus ∃ y. a = proj2-abs (vector [0,y,1]) by (simp add: exI [of - ?y])
qed

8.3 Definition of the Klein–Beltrami model of the hyperbolic plane

abbreviation hyp2 == K2
typedef hyp2 = K2
  using K2-centre-in-K2
  by auto
definition hyp2-rep :: hyp2 ⇒ real^2 where
hyp2-rep p ≜ cart2-pt (Rep-hyp2 p)

definition hyp2-abs :: real^2 ⇒ hyp2 where
hyp2-abs v = Abs-hyp2 (proj2-pt v)

lemma norm-lt-1-iff-in-hyp2:
shows norm v < 1 ←→ proj2-pt v ∈ hyp2
proof −
let ?v' = vector2-append1 v
have ?v' ≠ 0 by (rule vector2-append1-non-zero)
from real-less-rsqrt [of norm v 1]
and abs-square-less-1 [of norm v]
have norm v < 1 ←→ (norm v)^2 < 1 by auto
hence norm v < 1 ←→ ?v' ∙ (M ∗ ?v') < 0 by (simp add: norm-M)
with (?v' ≠ 0)
have norm v < 1 ←→ proj2-abs ?v' ∈ K2 by (subst K2-abs)
thus norm v < 1 ←→ proj2-pt v ∈ hyp2 by (unfold proj2-pt-def)
qed

lemma norm-eq-1-iff-in-S:
shows norm v = 1 ←→ proj2-pt v ∈ S
proof −
let ?v' = vector2-append1 v
have ?v' ≠ 0 by (rule vector2-append1-non-zero)
from real-sqrt-unique [of norm v 1]
have norm v = 1 ←→ (norm v)^2 = 1 by auto
hence norm v = 1 ←→ ?v' ∙ (M ∗ ?v') = 0 by (simp add: norm-M)
with (?v' ≠ 0)
have norm v = 1 ←→ proj2-abs ?v' ∈ S by (subst S-abs)
thus norm v = 1 ←→ proj2-pt v ∈ S by (unfold proj2-pt-def)
qed

lemma norm-le-1-iff-in-hyp2-S:
norm v ≤ 1 ←→ proj2-pt v ∈ hyp2 ∪ S
using norm-lt-1-iff-in-hyp2 [of v] and norm-eq-1-iff-in-S [of v]
by auto

proof −
let ?p' = Rep-hyp2 p
let ?v = proj2-rep ?p'
have ?v ≠ 0 by (rule proj2-rep-non-zero)

have proj2-abs ?v = ?p' by (rule proj2-abs-rep)

have ?p' ∈ hyp2 by (rule Rep-hyp2)
with (?v ≠ 0) and (proj2-abs ?v = ?p')
have ?v ∙ (M ∗ ?v) < 0 by (simp add: K2-imp-M-neg)
hence $v \not\equiv 3 \neq 0$ by (rule M-neg-imp-z-non-zero)
hence proj2-pt (cart2-pt $?p') = $?p' by (rule proj2-cart2)
thus proj2-pt (hyp2-rep p) = $?p' by (unfold hyp2-rep-def)
qed

lemma hyp2-rep-abs:
  assumes norm v < 1
  shows hyp2-rep (hyp2-abs v) = v
proof –
  from ⟨norm v < 1⟩ have proj2-pt v ∈ hyp2 by (simp add: norm-lt-1-iff-in-hyp2)
  hence Rep-hyp2 (Abs-hyp2 (proj2-pt v)) = proj2-pt v
    by (simp add: Abs-hyp2-inverse)
  hence hyp2-rep (hyp2-abs v) = cart2-pt (proj2-pt v)
    by (unfold hyp2-rep-def hyp2-abs-def simp)
  thus hyp2-rep (hyp2-abs v) = v by (simp add: cart2-proj2)
qed

lemma hyp2-abs-rep:
  hyp2-abs (hyp2-rep p) = p
by (unfold hyp2-abs-def) (simp add: proj2-pt-hyp2-rep Rep-hyp2-inverse)

lemma norm-hyp2-rep-lt-1:
  norm (hyp2-rep p) < 1
proof –
  have proj2-pt (hyp2-rep p) = Rep-hyp2 p by (rule proj2-pt-hyp2-rep)
  hence proj2-pt (hyp2-rep p) ∈ hyp2 by (simp add: Rep-hyp2)
  thus norm (hyp2-rep p) < 1 by (simp add: norm-lt-1-iff-in-hyp2)
qed

lemma hyp2-S-z-non-zero:
  assumes p ∈ hyp2 ∪ S
  shows z-non-zero p
proof –
  from ⟨p ∈ hyp2 ∪ S⟩ have conic-sgn p ≤ 0 by (unfold K2-def S-def) auto
  hence conic-sgn p ≠ 1 by simp
  thus z-non-zero p by (rule conic-sgn-not-1-z-non-zero)
qed

lemma hyp2-S-not-equal:
  assumes a ∈ hyp2 and p ∈ S
  shows a ≠ p
  using assms and S-K2-empty
  by auto

lemma hyp2-S-cart2-inj:
  assumes p ∈ hyp2 ∪ S and q ∈ hyp2 ∪ S and cart2-pt p = cart2-pt q
  shows p = q
proof –
  from ⟨p ∈ hyp2 ∪ S⟩ and ⟨q ∈ hyp2 ∪ S⟩
have \( z \text{-non-zero } p \) and \( z \text{-non-zero } q \) by \((\text{simp-all add: hyp2-S-z-non-zero})\)

hence \( \text{proj2-pt } (\text{cart2-pt } p) = p \) and \( \text{proj2-pt } (\text{cart2-pt } q) = q \)

by \((\text{simp-all add: proj2-cart2})\)

from \((\text{cart2-pt } p = \text{cart2-pt } q)\)

have \( \text{proj2-pt } (\text{cart2-pt } p) = \text{proj2-pt } (\text{cart2-pt } q) \) by \text{simp}

with \((\text{proj2-pt } (\text{cart2-pt } p) = p) \) \text{[symmetric]} and \((\text{proj2-pt } (\text{cart2-pt } q) = q)\)

show \( p = q \) by \text{simp}

qed

lemma \text{on-equator-in-hyp2-rep}:

assumes \( a \in \text{hyp2} \) and \( \text{proj2-incident } a \text{ equator} \)

shows \( \exists x. |x| < 1 \land a = \text{proj2-abs } (\text{vector } [x,0,1]) \)

proof –

from \((a \in \text{hyp2})\) have \( z \text{-non-zero } a \) by \((\text{simp add: hyp2-S-z-non-zero})\)

with \((\text{proj2-incident } a \text{ equator})\) and \text{on-equator-rep}

obtain \( x \) where \( a = \text{proj2-abs } (\text{vector } [x,0,1]) \) (is \( a = \text{proj2-abs } ?v \))

by \text{auto}

have \(?v \neq 0\) by \((\text{simp add: vec-eq-iff forall-3 vector-3})\)

with \((a \in \text{hyp2})\) and \( (a = \text{proj2-abs } ?v)\)

have \(?v \cdot (M \ast v \ ?v) < 0\) by \((\text{simp add: K2-abs})\)

hence \( x^2 < 1 \)

unfolding \( M \text{-def matrix-vector-mult-def inner-vec-def} \)

by \((\text{simp add: sum-3 vector-3 power2-eq-square})\)

with \( \text{real-sqrt-abs } [\text{of } x] \) and \( \text{real-sqrt-less-iff } [\text{of } x^2 1] \)

have \( |x| < 1 \) by \text{simp}

with \((a = \text{proj2-abs } ?v)\)

show \( \exists x. |x| < 1 \land a = \text{proj2-abs } (\text{vector } [x,0,1]) \)

by \((\text{simp add: exI } [\text{of - } x])\)

qed

lemma \text{on-meridian-in-hyp2-rep}:

assumes \( a \in \text{hyp2} \) and \( \text{proj2-incident } a \text{ meridian} \)

shows \( \exists y. |y| < 1 \land a = \text{proj2-abs } (\text{vector } [0,y,1]) \)

proof –

from \((a \in \text{hyp2})\) have \( z \text{-non-zero } a \) by \((\text{simp add: hyp2-S-z-non-zero})\)

with \((\text{proj2-incident } a \text{ meridian})\) and \text{on-meridian-rep}

obtain \( y \) where \( a = \text{proj2-abs } (\text{vector } [0,y,1]) \) (is \( a = \text{proj2-abs } ?v \))

by \text{auto}

have \(?v \neq 0\) by \((\text{simp add: vec-eq-iff forall-3 vector-3})\)

with \((a \in \text{hyp2})\) and \( (a = \text{proj2-abs } ?v)\)

have \(?v \cdot (M \ast v \ ?v) < 0\) by \((\text{simp add: K2-abs})\)

hence \( y^2 < 1 \)

unfolding \( M \text{-def matrix-vector-mult-def inner-vec-def} \)

by \((\text{simp add: sum-3 vector-3 power2-eq-square})\)

with \( \text{real-sqrt-abs } [\text{of } y] \) and \( \text{real-sqrt-less-iff } [\text{of } y^2 1] \)

have \( |y| < 1 \) by \text{simp}
with \( \langle a = \text{proj2-abs } \vec{v} \rangle \)

show \( \exists y. \ |y| < 1 \land a = \text{proj2-abs } (\text{vector } [0,y,1]) \)

by (simp add: exI \([a \cdot -y]\])

qed

**definition** hyp2-cltn2 :: hyp2 \( \Rightarrow \) cltn2 \( \Rightarrow \) hyp2 where

hyp2-cltn2 p A \( \triangleq \) Abs-hyp2 \((\text{apply-cltn2 } (\text{Rep-hyp2 } p) A)\)

**definition** is-K2-isometry :: cltn2 \( \Rightarrow \) bool where

is-K2-isometry J \( \triangleq \) \( \forall p. \ \text{apply-cltn2 } p J \in S \iff p \in S \)

**lemma** cltn2-id-is-K2-isometry: is-K2-isometry cltn2-id

unfolding is-K2-isometry-def by simp

**lemma** J-M-J-transpose-K2-isometry:

assumes \( k \neq 0 \)

and \( \text{repJ } \ast M \ast \text{transpose } \text{repJ} = k \ast R \ast M \) \( \text{(is } ?N = -) \)

shows is-K2-isometry \((\text{cltn2-abs } \text{repJ}) \) \( \text{(is is-K2-isometry } ?J) \)

proof −

from \( \langle ?N = k \ast R \ast M \rangle \)

have \( ?N \ast \ast ((1/k) \ast R \ast M) = \text{mat } 1 \)

by (simp add: matrix-scalar-ac \([k \neq 0] \text{ M-self-inverse}\))

with right-invertible-iff-invertible \([\text{of repJ}\]

have invertible repJ

by (simp add: matrix-mul-assoc exI \([\text{of } M \ast \text{transpose repJ } \ast \ast ((1/k) \ast R \ast M)])\)

have \( \forall t. \ \text{apply-cltn2 } t \ ?J \in S \iff t \in S \)

proof

fix t :: proj2

have proj2-rep \( t \ast (k \ast R \ast M) \ast \text{v proj2-rep } t \)

\( = k \ast (\text{proj2-rep } t \ast (M \ast \text{v proj2-rep } t)) \)

by (simp add: scaleR-matrix-vector-assoc \([\text{symmetric}]\) dot-scaleR-mult)

with \( \langle ?N = k \ast R \ast M \rangle \)

have proj2-rep \( t \ast (?N \ast \text{v proj2-rep } t) \)

\( = k \ast (\text{proj2-rep } t \ast (M \ast \text{v proj2-rep } t)) \)

by simp

hence proj2-rep \( t \ast (?N \ast \text{v proj2-rep } t) = 0 \)

\( \iff k \ast (\text{proj2-rep } t \ast (M \ast \text{v proj2-rep } t)) = 0 \)

by simp

with \( k \neq 0 \)

have proj2-rep \( t \ast (?N \ast \text{v proj2-rep } t) = 0 \)

\( \iff \text{proj2-rep } t \ast (M \ast \text{v proj2-rep } t) = 0 \)

by simp

with invertible repJ;

have apply-cltn2 \( t \ ?J \in S \iff \text{proj2-rep } t \ast (M \ast \text{v proj2-rep } t) = 0 \)

by (simp add: apply-cltn2-right-abs-in-S)

thus apply-cltn2 \( t \ ?J \in S \iff t \in S \) by (unfold S-alt-def)

123
qed
thus is-K2-isometry J by (unfold is-K2-isometry-def)
qed

lemma equator-reflect-K2-isometry:
shows is-K2-isometry equator-reflect
unfolding compass-reflect-defs
by (rule J-M-J-transpose-K2-isometry [of 1])
(simp-all add: M-def matrix-matrix-mult-def transpose-def
vec-eq-iff forall-3 sum-3 vector-3)

lemma meridian-reflect-K2-isometry:
shows is-K2-isometry meridian-reflect
unfolding compass-reflect-defs
by (rule J-M-J-transpose-K2-isometry [of 1])
(simp-all add: M-def matrix-matrix-mult-def transpose-def
vec-eq-iff forall-3 sum-3 vector-3)

lemma cltn2-compose-is-K2-isometry:
assumes is-K2-isometry H and is-K2-isometry J
shows is-K2-isometry (cltn2-compose H J)
using is-K2-isometry H and is-K2-isometry J;
unfolding is-K2-isometry-def
by (simp add: cltn2.act-act [simplified, symmetric])

lemma cltn2-inverse-is-K2-isometry:
assumes is-K2-isometry J
shows is-K2-isometry (cltn2-inverse J)
proof –
{ fix p
from is-K2-isometry J
have apply-cltn2 p (cltn2-inverse J) ∈ S
←→ apply-cltn2 (apply-cltn2 p (cltn2-inverse J)) J ∈ S
unfolding is-K2-isometry-def
by simp
hence apply-cltn2 p (cltn2-inverse J) ∈ S ←→ p ∈ S
by (simp add: cltn2.act-inv-act [simplified]) }
thus is-K2-isometry (cltn2-inverse J)
unfolding is-K2-isometry-def ..
qed

interpretation K2-isometry-subgroup: subgroup
Collect is-K2-isometry
{|carrier = UNIV, mult = cltn2-compose, one = cltn2-id|}
unfolding subgroup-def
by (simp add:
cltn2-id-is-K2-isometry
cltn2-compose-is-K2-isometry
cltn2-inverse-is-K2-isometry)
interpretation $K2$-isometry: group

$|carrier| = \text{Collect is-K2-isometry, mult = cltn2-compose, one = cltn2-id|}$
using cltn2.is-group and $K2$-isometry-subgroup

by simp

lemma $K2$-isometry-inverse-inv [simp]:

assumes is-K2-isometry $J$

shows $\text{inv(|carrier| = Collect is-K2-isometry, mult = cltn2-compose, one = cltn2-id|)} J$

= cltn2-inverse $J$

using cltn2-left-inverse
and (is-K2-isometry $J$)
and cltn2-inverse-is-K2-isometry
and $K2$-isometry.inv-equality

by simp

definition real-hyp2-C :: $[\text{hyp2, hyp2, hyp2, hyp2}] \Rightarrow \text{bool}$

\[
\text{p q \equiv K_r s \triangleq (\exists A. \text{is-K2-isometry } A \land \text{hyp2-cltn2 } p A = r \land \text{hyp2-cltn2 } q A = s)}
\]

definition real-hyp2-B :: $[\text{hyp2, hyp2, hyp2}] \Rightarrow \text{bool}$

\[
B_K p q r \triangleq B_{K, R} (\text{hyp2-rep } p) (\text{hyp2-rep } q) (\text{hyp2-rep } r)
\]

8.4 $K$-isometries map the interior of the conic to itself

lemma collinear-quadratic:

assumes $t = i \ast R a + r$

shows $t \cdot (M \ast v t) = (a \cdot (M \ast v a)) \ast i^2 + 2 \ast (a \cdot (M \ast v r)) \ast i + r \cdot (M \ast v r)$

proof –

from $M\text{-reverse}$ have $i \ast (a \cdot (M \ast v r)) = i \ast (r \cdot (M \ast v a))$ by simp
with ($t = i \ast R a + r$)

show $t \cdot (M \ast v t) = (a \cdot (M \ast v a)) \ast i^2 + 2 \ast (a \cdot (M \ast v r)) \ast i + r \cdot (M \ast v r)$

by (simp add:
inner-add-left
matrix-vector-right-distrib
inner-add-right
matrix-scaleR-vector-ac
inner-scaleR-right
scaleR-matrix-vector-assoc [symmetric]
$M\text{-reverse}$
power2-eq-square
algebra-simps)

qed
lemma \( S\text{-quadratic'} \):  
assumes \( p \neq 0 \) and \( q \neq 0 \) and \( \text{proj2-abs} \, p \neq \text{proj2-abs} \, q \)  
shows \( \text{proj2-abs} \, (k \ast_R \, p + q) \in S \)  
\( \iff p \cdot (M \ast v \, p) \ast k^2 + q \cdot (M \ast v \, q) = 0 \)  
proof –  
let \( ?r = k \ast_R \, p + q \)  
from \( (p \neq 0) \) and \( (q \neq 0) \) and \( (\text{proj2-abs} \, p \neq \text{proj2-abs} \, q) \)  
and dependent-proj2-abs \, [of \, p \, q \, k \, 1] \)  
have \( ?r \neq 0 \) by auto  
hence \( \text{proj2-abs} \, ?r \in S \iff \) by \( (\text{rule} \, S\text{-abs}) \)  
with \( \text{collinear-quadratic} \, [\text{of} \, ?r \, k \, p \, q] \)  
show \( \text{proj2-abs} \, ?r \in S \iff \) by \( (\text{simp add: \, dot-lmul-matrix [symmetric] \, algebra-simps}) \)  
qed

lemma \( S\text{-quadratic} \):  
assumes \( p \neq q \) and \( r = \text{proj2-abs} \, (k \ast_R \, \text{proj2-rep} \, p + \text{proj2-rep} \, q) \)  
shows \( r \in S \iff \) \( p \cdot (M \ast v \, \text{proj2-rep} \, p) \ast k^2 + p \cdot (M \ast v \, \text{proj2-rep} \, q) \ast 2 \ast k + q \cdot (M \ast v \, q) = 0 \)  
proof –  
let \( ?u = \text{proj2-rep} \, p \)  
let \( ?v = \text{proj2-rep} \, q \)  
let \( ?w = k \ast_R \, ?u + ?v \)  
have \( ?u \neq 0 \) and \( ?v \neq 0 \) by \( (\text{rule} \, \text{proj2-rep-non-zero}) \)  
from \( (p \neq q) \) have \( \text{proj2-abs} \, ?u \neq \text{proj2-abs} \, ?v \) by \( (\text{simp add: \, proj2-abs-rep}) \)  
with \( (?u \neq 0) \) and \( (?v \neq 0) \) and \( (r = \text{proj2-abs} \, ?u) \)  
show \( r \in S \iff \) \( ?u \cdot (M \ast v \, ?u) \ast k^2 + ?u \cdot (M \ast v \, ?v) \ast 2 \ast k + ?v \cdot (M \ast v \, ?v) = 0 \)  
by \( (\text{simp add: \, S\text{-quadratic'}}) \)  
qed

definition \( \text{quarter-discrim} :: \, \text{real}^3 \Rightarrow \text{real}^3 \Rightarrow \text{real} \) where  
\( \text{quarter-discrim} \, p \, q \triangleq (p \cdot (M \ast v \, p))^2 - p \cdot (M \ast v \, p) \ast q \cdot (M \ast v \, q) \)

lemma \( \text{quarter-discrim-invariant} \):  
assumes \( t = i \ast_R \, a + r \)  
shows \( \text{quarter-discrim} \, a \, t = \text{quarter-discrim} \, a \, r \)  
proof –  
from \( (t = i \ast_R \, a + r) \)  
have \( a \cdot (M \ast v \, t) = i \ast (a \cdot (M \ast v \, a)) + a \cdot (M \ast v \, r) \)  
by \( (\text{simp add: \, matrix-vector-right-distrib \, inner-add-right \, matrix-scaleR-vector-ac}) \)  
126
lemma quarter-discrim-positive:
assumes p \neq 0 \text{ and } q \neq 0 \text{ and } \text{proj2-abs } p \neq \text{proj2-abs } q \text{ (is } ?pp \neq ?pq)
and \text{proj2-abs } p \in K2
shows \text{quarter-discrim } p \land q > 0
proof
- let \( \tilde{t} = -q \$3/p \$3 \)
- let \( \tilde{t} = \tilde{t}^R \ p + q \)

from \( p \neq 0 \text{ and } (?pp \in K2) \)
have \( p \cdot (M \ast v \ p) < 0 \text{ by (subst K2-obs [symmetric])} \)
hence \( \$3 \neq 0 \text{ by (rule M-neg-imp-z-non-zero)} \)
hence \( \&t \$3 = 0 \text{ by simp} \)
hence \( \tilde{t} \cdot (M \ast v \ ?t) = (\&t \$1)^2 + (\&t \$2)^2 \)
    unfolding \( \text{matrix-vector-mult-def} \text{ and } M \text{-def and } \text{vector-def} \)
by (simp add: \text{inner-vec-def sum-3 power2-eq-square})

from \( \&t \neq 0 \text{; have } p \neq 0 \text{ by auto} \)
with \( \langle q \neq 0 \rangle \text{ and } (?pp \neq ?pq) \text{ and } \text{dependent-proj2-abs [of } p \ q \ \&i 1 \rangle \)
have \( \tilde{t} \neq 0 \text{ by auto} \)
with \( \langle \&t \$3 = 0 \rangle \text{ have } \&t \$1 \neq 0 \vee \&t \$2 \neq 0 \text{ by (simp add: vec-eq-iff forall-3)} \)
hence \( (\&t \$1)^2 > 0 \vee (\&t \$2)^2 > 0 \text{ by simp} \)
moreover have \( (\&t \$2)^2 \geq 0 \text{ and } (\&t \$1)^2 \geq 0 \text{ by simp-all} \)
ultimately have \( (\&t \$1)^2 + (\&t \$2)^2 > 0 \text{ by arith} \)
with \( \langle \&t \cdot (M \ast v \ ?t) = (\&t \$1)^2 + (\&t \$2)^2 \rangle \text{ have } \&t \cdot (M \ast v \ ?t) > 0 \text{ by simp} \)
with \( \text{mult-neg-pos [of } p \cdot (M \ast v \ p) \text{ and } (p \cdot (M \ast v \ p) < 0) \rangle \)
have \( p \cdot (M \ast v \ ?t) \cdot (\&t \cdot (M \ast v \ ?t)) < 0 \text{ by simp} \)
moreover have \( p \cdot (M \ast v \ ?t))^2 \geq 0 \text{ by simp} \)
ultimately
have \( (p \cdot (M \ast v \ ?t))^2 - p \cdot (M \ast v \ p) \cdot (\&t \cdot (M \ast v \ ?t)) > 0 \text{ by arith} \)
with \( \text{quarter-discrim-invariant [of } \&t \ ?t \ p \ q \rangle \)
show \( \text{quarter-discrim } p \ q > 0 \text{ by (unfold \text{quarter-discrim-def}, simp)} \)
qed
lemma quarter-discrim-self-zero:
assumes proj2-abs a = proj2-abs b
shows quarter-discrim a b = 0
proof cases
  assume b = 0
  thus quarter-discrim a b = 0 by (unfold quarter-discrim-def, simp)
next
  assume b ≠ 0
  with ⟨proj2-abs a = proj2-abs b⟩ and proj2-abs-abs-mult
  obtain k where a = k *R b by auto
  thus quarter-discrim a b = 0
  unfolding quarter-discrim-def
  by (simp add: power2-eq-square
       matrix-scaleR-vector-ac
       scaleR-matrix-vector-assoc [symmetric])
qed

definition S-intersection-coeff1 :: real^3 ⇒ real^3 ⇒ real where
  S-intersection-coeff1 p q ≜ (- p · (M * v q) + sqrt (quarter-discrim p q)) / (p · (M * v p))

definition S-intersection-coeff2 :: real^3 ⇒ real^3 ⇒ real where
  S-intersection-coeff2 p q ≜ (- p · (M * v q) - sqrt (quarter-discrim p q)) / (p · (M * v p))

definition S-intersection1-rep :: real^3 ⇒ real^3 ⇒ real^3 where
  S-intersection1-rep p q ≜ (S-intersection-coeff1 p q) *R p + q

definition S-intersection2-rep :: real^3 ⇒ real^3 ⇒ real^3 where
  S-intersection2-rep p q ≜ (S-intersection-coeff2 p q) *R p + q

definition S-intersection1 :: real^3 ⇒ real^3 ⇒ proj2 where
  S-intersection1 p q ≜ proj2-abs (S-intersection1-rep p q)

definition S-intersection2 :: real^3 ⇒ real^3 ⇒ proj2 where
  S-intersection2 p q ≜ proj2-abs (S-intersection2-rep p q)

lemmas S-intersection-coeffs-defs =
  S-intersection-coeff1-def S-intersection-coeff2-def

lemmas S-intersections-defs =
  S-intersection1-def S-intersection2-def
  S-intersection1-rep-def S-intersection2-rep-def

lemma S-intersection-coeffs-distinct:
  assumes p ≠ 0 and q ≠ 0 and proj2-abs p ≠ proj2-abs q (is ?pp ≠ ?pq)
  and proj2-abs p ∈ K^2
  shows S-intersection-coeff1 p q ≠ S-intersection-coeff2 p q
proof –
from \( p \neq 0 \) and \( \langle \forall pp \in K^2 \rangle \)

have \( p \cdot (M \ast v p) < 0 \) by (subst K2-abs [symmetric])

from assms have quarter-discrim \( p \ q > 0 \) by (rule quarter-discrim-positive)

with \( p \cdot (M \ast v p) < 0 \);

show \( S\text{-}intersection\text{-}coeff1 \) \( p \ q \neq S\text{-}intersection\text{-}coeff2 \) \( p \ q \)

by (unfold S-intersection-coeffs-defs, simp)

qed

lemma \( S\text{-}intersections\text{-}distinct \):

assumes \( p \neq 0 \) and \( q \neq 0 \) and \( proj2\text{-}abs \ p \neq proj2\text{-}abs \ q \) (is \( \langle \forall pp \neq \forall pq \rangle \))

and \( proj2\text{-}abs \ p \in K^2 \)

shows \( S\text{-}intersection1 \) \( p \ q \neq S\text{-}intersection2 \) \( p \ q \)

proof –

from \( \langle p \neq 0 \rangle \) and \( \langle q \neq 0 \rangle \) and \( \langle \forall pp \neq \forall pq \rangle \) and \( \langle \forall pp \in K^2 \rangle \)

have \( S\text{-}intersection\text{-}coeff1 \) \( p \ q \neq S\text{-}intersection\text{-}coeff2 \) \( p \ q \)

by (rule S-intersection-coeffs-distinct)

with \( \langle p \neq 0 \rangle \) and \( \langle q \neq 0 \rangle \) and \( \langle \forall pp \neq \forall pq \rangle \) and \( proj2\text{-}Col\text{-}coeff\text{-}unique \)

show \( S\text{-}intersection1 \) \( p \ q \neq S\text{-}intersection2 \) \( p \ q \)

by (unfold S-intersections-defs, auto)

qed

lemma \( S\text{-}intersections\text{-}in\text{-}S \):

assumes \( p \neq 0 \) and \( q \neq 0 \) and \( proj2\text{-}abs \ p \neq proj2\text{-}abs \ q \) (is \( \langle \forall pp \neq \forall pq \rangle \))

and \( proj2\text{-}abs \ p \in K^2 \)

shows \( S\text{-}intersection1 \) \( p \ q \in S \) and \( S\text{-}intersection2 \) \( p \ q \in S \)

proof –

let \( \langle \forall j \rangle = S\text{-}intersection\text{-}coeff1 \) \( p \ q \)

let \( \langle \forall k \rangle = S\text{-}intersection\text{-}coeff2 \) \( p \ q \)

let \( \langle \forall \ a \rangle = p \cdot (M \ast v p) \)

let \( \langle \forall \ b \rangle = 2 \cdot (p \cdot (M \ast v p)) \)

let \( \langle \forall \ c \rangle = q \cdot (M \ast v q) \)

from \( \langle p \neq 0 \rangle \) and \( \langle \forall pp \in K^2 \rangle \) have \( \langle \forall \ a \rangle < 0 \) by (subst K2-abs [symmetric])

have \( q\text{d}: \text{discrim} \ (p \ b \ c = 4 \ast \text{quarter-discrim} \ p \ q) \)

unfolding \( \text{discrim}\text{-}def \) \( \text{quarter-discrim}\text{-}def \)

by (simp add: power2-eq-square)

with \( \text{times\text{-}divide\text{-}times\text{-}eq} \) of

\( 2 \ 2 \ \text{sqrt} \ (\text{quarter\text{-}discrim} \ p \ q) - p \cdot (M \ast v q) \ \langle \forall a \rangle \)

and \( \text{times\text{-}divide\text{-}times\text{-}eq} \) of

\( 2 \ 2 - p \cdot (M \ast v q) - \text{sqrt} \ (\text{quarter\text{-}discrim} \ p \ q) \ \langle \forall a \rangle \)

and \( \text{real\text{-}sqr\text{-}mult} \) and \( \text{real\text{-}sqr\text{-}abs} \) of 2

have \( \langle \forall j \rangle = (-p \cdot \text{sqrt} \ (\text{discrim} \ (p \ b \ c)) \/ (2 \ast \langle \forall a \rangle) \)

and \( \langle \forall k \rangle = (-p - \text{sqrt} \ (\text{discrim} \ (p \ b \ c)) \/ (2 \ast \langle \forall a \rangle) \)

by (unfold S-intersection-coeffs-defs, simp-all add: algebra-simps)

from assms have quarter-discrim \( p \ q > 0 \) by (rule quarter-discrim-positive)

with \( q\text{d} \)
have \( \text{discrim} \ (p \cdot (M \ast v \ p)) \ (2 \ast (p \cdot (M \ast v \ q))) \ (q \cdot (M \ast v \ q)) > 0 \)

by \( \text{simp} \)

with \( \langle \ ?j = (-\ ?b + \sqrt{\text{discrim} \ ?a \ ?b \ ?c}) \ / \ (2 \ast \ ?a) \rangle \)

and \( \langle \ ?k = (-\ ?b - \sqrt{\text{discrim} \ ?a \ ?b \ ?c}) \ / \ (2 \ast \ ?a) \rangle \)

and \( \langle \ ?a < 0 \rangle \ and \ \text{discriminant-nonneg} \ [\text{of} \ ?a \ ?b \ ?c \ ?j] \)

and \( \text{discriminant-nonneg} \ [\text{of} \ ?a \ ?b \ ?c \ ?k] \)

have \( p \cdot (M \ast v \ p) \ast \ ?j^2 + 2 \ast (p \cdot (M \ast v \ q)) \ast \ ?j + q \cdot (M \ast v \ q) = 0 \)

and \( p \cdot (M \ast v \ p) \ast \ ?k^2 + 2 \ast (p \cdot (M \ast v \ q)) \ast \ ?k + q \cdot (M \ast v \ q) = 0 \)

by \( \text{unfold S-intersection-coeffs-defs, auto} \)

with \( \langle \ ?pp \neq ?pq \rangle \ and \ \langle \ ?pp \neq ?pq \rangle \ and \ ?pp \neq ?pq \ and \ S\text{-quadratic} \)

show \( S\text{-intersection1} \ p \ q \in S \ and \ S\text{-intersection2} \ p \ q \in S \)

by \( \text{unfold S-intersections-defs, simp-all} \)

qed

lemma \( S\text{-intersections-Col} \):

assumes \( p \neq 0 \ and \ q \neq 0 \)

shows \( \text{proj2-Col} \ (\text{proj2-abs} \ p) \ (\text{proj2-abs} \ q) \ (S\text{-intersection1} \ p \ q) \)

(is \( \text{proj2-Col} ? ?pp ? ?pq ? ?pr \))

and \( \text{proj2-Col} \ (\text{proj2-abs} \ p) \ (\text{proj2-abs} \ q) \ (S\text{-intersection2} \ p \ q) \)

(is \( \text{proj2-Col} ? ?pp ? ?pq ? ?ps \))

proof –

\{
  \text{assume} \ ?pp = ?pq


  by \ (\text{simp-all add: proj2-Col-coincide}) \}

moreover

\{
  \text{assume} \ ?pp \neq ?pq

  with \( \langle \ ?p \neq 0 \rangle \ and \ \langle \ ?q \neq 0 \rangle \ and \ \text{dependent-proj2-abs} \ [\text{of} \ ?p \ ?q \ - \ 1] \)

  have \( S\text{-intersection1}-\text{rep} \ p \ q \neq 0 \) \ (\text{is} \ ?r \neq 0)

  and \( S\text{-intersection2}-\text{rep} \ p \ q \neq 0 \) \ (\text{is} \ ?s \neq 0)

  by \ (\text{unfold S-intersection1-rep-def S-intersection2-rep-def, auto})

  with \( \langle \ ?p \neq 0 \rangle \ and \ \langle \ ?q \neq 0 \rangle \)

  and \( \text{proj2-Col-\text{abs} [\text{of} \ ?p \ ?q \ ?r \ S\text{-intersection-coeff}1 \ ?p \ ?q \ - \ 1]} \)

  and \( \text{proj2-Col-\text{abs} [\text{of} \ ?p \ ?q \ ?s \ S\text{-intersection-coeff}2 \ ?p \ ?q \ - \ 1]} \)


  by \ (\text{unfold S-intersections-defs, simp-all}) \}


qed

lemma \( S\text{-intersections-incident} \):

assumes \( p \neq 0 \ and \ q \neq 0 \ and \ \text{proj2-abs} \ p \neq \ \text{proj2-abs} \ q \) \ (\text{is} \ ?pp \neq ?pq)

and \( \text{proj2-incident} \ (\text{proj2-abs} \ p) \ l \ and \ \text{proj2-incident} \ (\text{proj2-abs} \ q) \ l \)

shows \( \text{proj2-incident} \ (S\text{-intersection1} \ p \ q) \ l \) \ (\text{is} \ \text{proj2-incident} \ ?pr \ l)

and \( \text{proj2-incident} \ (S\text{-intersection2} \ p \ q) \ l \) \ (\text{is} \ \text{proj2-incident} \ ?ps \ l) \)

proof –

from \( \langle \ ?p \neq 0 \rangle \ and \ \langle \ ?q \neq 0 \rangle \)


by \ (\text{rule S-intersections-Col}+)

with \( \langle ?pp \neq ?pq \rangle \ and \ \text{proj2-incident} \ ?pp \ l \) \ and \( \text{proj2-incident} \ ?pq \ l \)

and \( \text{proj2-incident-iff-Col} \)

130
show proj2-incident ?pr l and proj2-incident ?ps l by fast+

qed

lemma K2-line-intersect-twice:
  assumes a ∈ K2 and a ≠ r
  shows ∃ s u. s ≠ a ∧ s ∈ S ∧ u ∈ S ∧ proj2-Col a r s ∧ proj2-Col a r u
proof –
  let ?a' = proj2-rep a
  let ?r' = proj2-rep r
  from proj2-rep-non-zero have ?a' ≠ 0 and ?r' ≠ 0 by simp-all
  from ⟨?a' ≠ 0⟩ and K2-imp-M-neg and proj2-abs-rep and ⟨a ∈ K2⟩
  have ?a' · (M ∗ v ?a') < 0 by simp
  from ⟨a ≠ r⟩ have proj2-abs ?a' ≠ proj2-abs ?r' by (simp add: proj2-abs-rep)
  from ⟨a ∈ K2⟩ have proj2-abs ?a' ∈ K2 by (simp add: proj2-abs-rep)
  with ⟨?a' ≠ 0⟩ and ⟨?r' ≠ 0⟩ and ⟨proj2-abs ?a' ≠ proj2-abs ?r'⟩
  have S-intersection1 ?a' ?r' ≠ S-intersection2 ?a' ?r' (is ?s ≠ ?u)
    by (rule S-intersections-distinct)
  from ⟨?a' ≠ 0⟩ and ⟨?r' ≠ 0⟩ and ⟨proj2-abs ?a' ≠ proj2-abs ?r'⟩
    and ⟨proj2-abs ?a' ∈ K2⟩
  have ?s ∈ S and ?u ∈ S by (rule S-intersections-in-S)+
  from ⟨?a' ≠ 0⟩ and ⟨?r' ≠ 0⟩
  have proj2-Col (proj2-abs ?a') (proj2-abs ?r') ?s
    and proj2-Col (proj2-abs ?a') (proj2-abs ?r') ?u
    by (rule S-intersections-Col)+
  hence proj2-Col a r ?s and proj2-Col a r ?u
    by (simp-all add: proj2-abs-rep)
  with ⟨?s ≠ ?u⟩ and ⟨?s ∈ S⟩ and ⟨?u ∈ S⟩
  show ∃ s u. s ≠ u ∧ s ∈ S ∧ u ∈ S ∧ proj2-Cola r s ∧ proj2-Col a r u
    by auto
qed

lemma point-in-S-polar-is-tangent:
  assumes p ∈ S and q ∈ S and proj2-incident q (polar p)
  shows q = p
proof –
  from ⟨p ∈ S⟩ have proj2-incident p (polar p)
    by (subst incident-own-polar-in-S)
  from line-incident-point-not-in-S
  obtain r where r ∉ S and proj2-incident r (polar p) by auto
  let ?u = proj2-rep r
  let ?v = proj2-rep p
  from ⟨r ∉ S⟩ and ⟨p ∈ S⟩ and ⟨q ∈ S⟩ have r ≠ p and q ≠ r by auto
  with ⟨proj2-incident p (polar p)⟩

131
and (proj2-incident q (polar p))
and (proj2-incident r (polar p))
and proj2-incident-iff [of r p polar p q]

obtain k where q = proj2-abs (k *r ?u + ?v) by auto
with (r ≠ p) and (q ∈ S) and S-quadratic
have ?u · (M *v ?u) * k^2 + ?u · (M *v ?v) * 2 * k + ?v · (M *v ?v) = 0
  by simp
moreover from (p ∈ S) have ?v · (M *v ?v) = 0 by (unfold S-alt-def)
moreover from (proj2-incident r (polar p))
have ?u · (M *v ?v) = 0 by (unfold incident-polar)
moreover from (r ∉ S) have ?u · (M *v ?u) ≠ 0 by (unfold S-alt-def)
ultimately have k = 0 by simp
with (q = proj2-abs (k *r ?u + ?v))
show q = p by (simp add: proj2-abs-rep)

qed

lemma line-through-K2-intersect-S-twice:
assumes p ∈ K2 and proj2-incident p l
shows ∃ r q. q ≠ r ∧ q ∈ S ∧ r ∈ S ∧ proj2-incident q l ∧ proj2-incident r l

proof –
from proj2-another-point-on-line
obtain s where s ≠ p and proj2-incident s l by auto
from (p ∈ K2) and (s ≠ p) and K2-line-intersect-twice [of p s]
obtain q and r where q ≠ r and q ∈ S and r ∈ S
  and proj2-Col p s q and proj2-Col p s r
  by auto
with (s ≠ p) and (proj2-incident p l) and (proj2-incident s l)
  and proj2-incident-iff-Col [of p s]
have proj2-incident q l and proj2-incident r l by fast+
with (q ≠ r) and (q ∈ S) and (r ∈ S)
show ∃ r q. q ≠ r ∧ q ∈ S ∧ r ∈ S ∧ proj2-incident q l ∧ proj2-incident r l
  by auto
qed

lemma line-through-K2-intersect-S-again:
assumes p ∈ K2 and proj2-incident p l
shows ∃ r q r. q ≠ r ∧ q ∈ S ∧ proj2-incident r l

proof –
from (p ∈ K2) and (proj2-incident p l)
  and line-through-K2-intersect-S-twice [of p l]
obtain s and t where s ≠ t and s ∈ S and t ∈ S
  and proj2-incident s l and proj2-incident t l
  by auto
show ∃ r q r. q ≠ r ∧ q ∈ S ∧ proj2-incident r l
  by cases
  assume t = q
  with (s ≠ t) and (s ∈ S) and (proj2-incident s l)
  have s ≠ q ∧ s ∈ S ∧ proj2-incident s l by simp
  thus ∃ r q r. q ≠ r ∧ q ∈ S ∧ proj2-incident r l ..
next
assume \( t \neq q \)
with \( t \in S \) and \( \text{proj2-incident } t \ l \)
have \( t \neq q \land t \in S \land \text{proj2-incident } t \ l \) by simp
thus \( \exists r. r \neq q \land r \in S \land \text{proj2-incident } r \ l \). ..
qed
qed

lemma line-through-K2-intersect-S:
assumes \( p \in K2 \) and \( \text{proj2-incident } p \ l \)
shows \( \exists r. r \neq p \land r \in S \land \text{proj2-incident } r \ l \)
proof –
from asms
have \( \exists r. r \neq p \land r \in S \land \text{proj2-incident } r \ l \)
by \( \text{(rule line-through-K2-intersect-S-again)} \)
thus \( \exists r. r \in S \land \text{proj2-incident } r \ l \) by auto
qed

lemma line-intersect-S-at-most-twice:
\( \exists p \ q. \forall r \in S. \text{proj2-incident } r \ l \rightarrow r = p \lor r = q \)
proof –
from line-incident-point-not-in-S
obtain \( s \) where \( s \notin S \) and \( \text{proj2-incident } s \ l \) by auto
let \( ?v = \text{proj2-rep } s \)
from proj2-another-point-on-line
obtain \( t \) where \( t \neq s \) and \( \text{proj2-incident } t \ l \) by auto
let \( ?w = \text{proj2-rep } t \)
have \( ?v \neq 0 \) and \( ?w \neq 0 \) by \( \text{(rule proj2-rep-non-zero)} \)
let \( ?a = ?v \cdot (M \ast v ?v) \)
let \( ?b = 2 \ast (?v \cdot (M \ast v ?w)) \)
let \( ?c = ?w \cdot (M \ast v ?w) \)
from \( s \notin S \) have \( ?a \neq 0 \)
unfolding \( S\text{-def} \) and \( \text{conic-sgn-def} \)
by auto
let \( ?j = (\neg ?b + \text{sqrt} (\text{discrim } ?a \ ?b \ ?c)) / (2 \ast ?a) \)
let \( ?k = (\neg ?b - \text{sqrt} (\text{discrim } ?a \ ?b \ ?c)) / (2 \ast ?a) \)
let \( ?p = \text{proj2-obs } (?j \ast_R ?v + ?w) \)
let \( ?q = \text{proj2-obs } (?k \ast_R ?v + ?w) \)
have \( \forall r \in S. \text{proj2-incident } r \ l \rightarrow r = ?p \lor r = ?q \)
proof
fix \( r \)
assume \( r \in S \)
with \( s \notin S \) have \( r \neq s \) by auto
{ assume \( \text{proj2-incident } r \ l \)
with \( t \neq s \) and \( r \neq s \) and \( \text{proj2-incident } s \ l \) and \( \text{proj2-incident } t \ l \)
and \( \text{proj2-incident-iff} \ [\text{of } s \ t \ l \ r] \)
obtain \( i \) where \( r = \text{proj2-obs } (i \ast_R ?v + ?w) \) by auto
with \( r \in S \) and \( t \neq s \) and \( \text{S-quadratic} \)
}
have \( a \cdot i^2 + b \cdot i + c = 0 \) by simp
with \( (a \neq 0) \) and discriminant-iff have \( i = ?j \lor i = ?k \) by simp
with \( (r = \text{proj2-abs } (i \ast_R ?v + ?w)) \) have \( r = ?p \lor r = ?q \) by auto 
thus \( \text{proj2-incident } r \ l \rightarrow r = ?p \lor r = ?q \) ..
qed
thus \( \exists \ p \ q. \forall \ r \in S. \text{proj2-incident } r \ l \rightarrow r = p \lor r = q \) by auto
qed

lemma \text{card-line-intersect-S}:  
assumes \( T \subseteq S \) and \( \text{proj2-set-Col } T \)
shows \( \text{card } T \leq 2 \)
proof –
from \( \langle \text{proj2-set-Col } T \rangle \)
obtain \( l \) where \( \forall \ \ p \in T. \text{proj2-incident } p \ l \) unfolding \( \text{proj2-set-Col-def} \) ..
from \( \text{line-intersect-S-at-most-twice } \) \( \langle \text{of } l \rangle \)
obtain \( b \) and \( c \) where \( \forall \ \ a \in S. \text{proj2-incident } a \ l \rightarrow a = b \lor a = c \) by auto
with \( \forall \ \ p \in T. \text{proj2-incident } p \ b \) and \( \langle T \subseteq S \rangle \)
have \( T \subseteq \{b,c\} \) by auto
hence \( \text{card } T \leq \text{card } \{b,c\} \) by \( \langle \text{simp add: card-mono} \rangle \)
also from \( \text{card-suc-ge-insert } \) \( \langle \text{of } b \ \{c\} \rangle \) have \( \ldots \leq 2 \) by simp
finally show \( \text{card } T \leq 2 \).
qed

lemma \text{line-S-two-intersections-only}:  
assumes \( p \neq q \) and \( p \in S \) and \( q \in S \) and \( r \in S \)
and \( \text{proj2-incident } p \ l \) and \( \text{proj2-incident } q \ l \) and \( \text{proj2-incident } r \ l \)
shows \( r = p \lor r = q \)
proof –
from \( \langle p \neq q \rangle \) have \( \text{card } \{p,q\} = 2 \) by simp
from \( \langle p \in S \rangle \) and \( \langle q \in S \rangle \) and \( \langle r \in S \rangle \) have \( \{r,p,q\} \subseteq S \) by simp-all
from \( \langle \text{proj2-incident } p \ b \rangle \) and \( \langle \text{proj2-incident } q \ b \rangle \) and \( \langle \text{proj2-incident } r \ b \rangle \)
have \( \text{proj2-set-Col } \{r,p,q\} \)
by \( \langle \text{unfold } \text{proj2-set-Col-def} \ \langle \text{simp add: exI } \langle \text{of } l \rangle \rangle \rangle \)
with \( \langle \{r,p,q\} \subseteq S \rangle \) have \( \text{card } \{r,p,q\} \leq 2 \) by \( \langle \text{rule card-line-intersect-S} \rangle \)
show \( r = p \lor r = q \)
proof \( \langle \text{rule ccontr} \rangle \)
assume \( \neg \ (r = p \lor r = q) \)
hence \( r \notin \{p,q\} \) by simp
with \( \langle \text{card } \{p,q\} = 2 \rangle \) and \( \text{card-insert-disjoint } \langle \{p,q\} \ r \rangle \)
have \( \text{card } \{r,p,q\} = 3 \) by simp
with \( \langle \text{card } \{r,p,q\} \leq 2 \rangle \) show \( \text{False} \) by simp
qed
qed

lemma \text{line-through-K2-intersect-S-exactly-twice}:  
assumes \( p \in K2 \) and \( \text{proj2-incident } p \ l \)
shows \( \exists \ q \ r. \ q \neq r \land q \in S \land r \in S \land \text{proj2-incident } l \land (\forall s \in S. \text{proj2-incident } s \ l \implies s = q \lor s = r) \)

proof –

from \( \langle p \in K2 \rangle \) and \( \langle \text{proj2-incident } p \ l \rangle \)
and \( \text{line-through-K2-intersect-S-twice } [\text{of } p \ l] \)
obtain \( q \) and \( r \) where \( q \neq r \) and \( q \in S \) and \( r \in S \)
and \( \text{proj2-incident } q \ l \) and \( \text{proj2-incident } r \ l \)
by auto

with \( \text{line-S-two-intersections-only} \)

show \( \exists \ q \ r. \ q \neq r \land q \in S \land r \in S \land \text{proj2-incident } q \ l \land \text{proj2-incident } r \ l \land (\forall s \in S. \text{proj2-incident } s \ l \implies s = q \lor s = r) \)
by blast

qed

lemma tangent-not-through-K2:

assumes \( p \in S \) and \( q \in K2 \)

shows \( \neg \text{proj2-incident } q \ (\text{polar } p) \)

proof

assume \( \text{proj2-incident } q \ (\text{polar } p) \)

with \( \langle q \in K2 \rangle \) and \( \text{line-through-K2-intersect-S-again } [\text{of } q \ \text{polar } p \ p] \)
obtain \( r \) where \( r \neq p \) and \( r \in S \) and \( \text{proj2-incident } r \ (\text{polar } p) \) by auto

from \( \langle p \in S \rangle \) and \( \langle r \in S \rangle \) and \( \langle \text{proj2-incident } r \ (\text{polar } p) \rangle \)

have \( r = p \) by \( \text{(rule point-in-S-polar-is-tangent)} \)

with \( \langle r \neq p \rangle \) show False ..

qed

lemma outside-exists-line-not-intersect-S:

assumes \( \text{conic-sgn } p = 1 \)

shows \( \exists l. \text{proj2-incident } p \ l \land (\forall q. \text{proj2-incident } q \ l \implies q \notin S) \)

proof –

let \( ?r = \text{proj2-intersection } (\text{polar } p) \ z-zero \)

have \( \text{proj2-incident } ?r \ (\text{polar } p) \) and \( \text{proj2-incident } ?r \ z-zero \)

by \( \text{(rule proj2-intersection-incident+)} \)

from \( \langle \text{proj2-incident } ?r \ z-zero \rangle \)

have \( \text{conic-sgn } ?r = 1 \) by \( \text{(rule z-zero-conic-sgn-1)} \)

with \( \langle \text{conic-sgn } p = 1 \rangle \)

have \( \text{proj2-rep } p \cdot (M \ast v \ \text{proj2-rep } p) > 0 \)

and \( \text{proj2-rep } ?r \cdot (M \ast v \ \text{proj2-rep } ?r) > 0 \)

by \( \text{(unfold conic-sgn-def) \ (simp-all add: sgn-1-pos)} \)

from \( \langle \text{proj2-incident } ?r \ (\text{polar } p) \rangle \)

have \( \text{proj2-incident } p \ (\text{polar } ?r) \) by \( \text{(rule incident-polar-swap)} \)

hence \( \text{proj2-rep } p \cdot (M \ast v \ \text{proj2-rep } ?r) = 0 \) by \( \text{(simp add: incident-polar)} \)

have \( p \neq ?r \)

proof

assume \( p = ?r \)

with \( \langle \text{proj2-incident } ?r \ (\text{polar } p) \rangle \) have \( \text{proj2-incident } p \ (\text{polar } p) \) by simp

hence \( \text{proj2-rep } p \cdot (M \ast v \ \text{proj2-rep } p) = 0 \) by \( \text{(simp add: incident-polar)} \)
with \( \langle \text{proj2-rep } p \cdot (M \ast v \text{proj2-rep } p) > 0 \rangle \); show False by simp

qed

let \( \mathcal{I} = \text{proj2-line-through } p \ ?r \)

have proj2-incident p \( \mathcal{I} \) and proj2-incident \( ?r \ ?l \)
  by (rule proj2-line-through-incident)+

have \( \forall q. \text{proj2-incident } q \ ?l \rightarrow q \notin S \)

proof
  fix q
  show proj2-incident q \( ?l \rightarrow q \notin S \)
  proof
    assume proj2-incident q \( ?l \)
    with \( p \neq ?r \) and \( (\exists k. \ q = \text{proj2-abs } (k \ast_R \text{proj2-rep } p + \text{proj2-rep } ?r) ) \)
    by (simp add: proj2-incident-iff [of p ?r ?l q])
  
  show q \( \notin S \)
  proof
    cases
    assume q = p
    with \( \text{conic-sgn } p = 1 \) show q \( \notin S \) by (unfold S-def simp)
    next
    assume q \( \neq p \)
    with \( q = \text{proj2-abs } (k \ast_R \text{proj2-rep } p + \text{proj2-rep } ?r) \)
    obtain k where \( q = \text{proj2-abs } (k \ast_R \text{proj2-rep } p + \text{proj2-rep } ?r) \)
      by auto
      from \( \langle \text{proj2-rep } p \cdot (M \ast v \text{proj2-rep } p) > 0 \rangle \)
    have \( \langle \text{proj2-rep } p \cdot (M \ast v \text{proj2-rep } p) \ast k^2 > 0 \; \rangle \)
      by simp
    with \( \langle \text{proj2-rep } p \cdot (M \ast v \text{proj2-rep } ?r) = 0 \; \rangle \)
      and \( \langle \text{proj2-rep } ?r \cdot (M \ast v \text{proj2-rep } ?r) > 0 \; \rangle \)
    have \( \langle \text{proj2-rep } p \cdot (M \ast v \text{proj2-rep } ?r) \ast k^2 \)
      + \text{proj2-rep } ?r \cdot (M \ast v \text{proj2-rep } ?r) \ast 2 \ast k
    + \text{proj2-rep } ?r \cdot (M \ast v \text{proj2-rep } ?r) \ast 0 \)
      > 0
      by simp
    with \( \langle p \neq ?r \rangle \) and \( \langle q = \text{proj2-abs } (k \ast_R \text{proj2-rep } p + \text{proj2-rep } ?r) \rangle \)
    show q \( \notin S \) by (simp add: S-quadratic)
  qed
  qed
  qed

with \( \langle \text{proj2-incident } p \ ?l \rangle \)

show \( \exists l. \text{proj2-incident } p \ l \land (\forall q. \text{proj2-incident } q \ l \rightarrow q \notin S) \)
  by (simp add: exI [of - ?l])

qed

lemma lines-through-intersect-S-twice-in-K2:
  assumes \( \forall l. \text{proj2-incident } p \ l \)
  \( \rightarrow (\exists \ q \ r. \ q \neq r \land q \in S \land r \in S \land \text{proj2-incident } q \ l \land \text{proj2-incident } r \ l) \)
shows $p \in K_2$

proof (rule ccontr)

assume $p \notin K_2$

hence $\text{conic-sgn } p \geq 0$ by (unfold $K_2$-def) simp

have $\neg (\forall \ l. \ \text{proj2-incident } p \ l \longrightarrow (\exists \ q \ r. \ q \neq r \land q \in S \land r \in S \land \text{proj2-incident } q \ l \land \text{proj2-incident } r \ l))$

proof cases

assume $\text{conic-sgn } p = 0$

hence $p \in S$ unfolding S-def ..

hence $\text{proj2-incident } p$ (polar $p$) by (simp add: incident-own-polar-in-S)

let $\l = \text{polar } p$

have $\neg (\exists \ q \ r. \ q \neq r \land q \in S \land r \in S \land \text{proj2-incident } q \ \l \land \text{proj2-incident } r \ \l)$

proof

assume $\exists \ q \ r. \ q \neq r \land q \in S \land r \in S \land \text{proj2-incident } q \ \l \land \text{proj2-incident } r \ \l$

then obtain $q$ and $r$ where $q \neq r$ and $q \in S$ and $r \in S$

and $\text{proj2-incident } q \ \l$ and $\text{proj2-incident } r \ \l$

by auto

from $(p \in S)$ and $(q \in S)$ and $(\text{proj2-incident } q \ \l)$

and $(r \in S)$ and $(\text{proj2-incident } r \ \l)$

have $q = p$ and $r = p$ by (simp add: point-in-S-polar-is-tangent)+

with $(q \neq r)$ show False by simp

qed

with $(\text{proj2-incident } p \ \l)$

show $\neg (\forall \ l. \ \text{proj2-incident } p \ l \longrightarrow (\exists \ q \ r. \ q \neq r \land q \in S \land r \in S \land \text{proj2-incident } q \ l \land \text{proj2-incident } r \ l))$

by auto

next

assume $\text{conic-sgn } p \neq 0$

with $(\text{conic-sgn } p \geq 0)$ have $\text{conic-sgn } p > 0$ by simp

hence $\text{sgn } (\text{conic-sgn } p) = 1$ by simp

hence $\text{conic-sgn } p = 1$ by (simp add: sgn-conic-sgn)

with outside-exists-line-not-intersect-S

obtain $\l$ where $\text{proj2-incident } p \ \l$ and $\forall \ q. \ \text{proj2-incident } q \ \l \longrightarrow q \notin S$

by auto

have $\neg (\exists \ q \ r. \ q \neq r \land q \in S \land r \in S \land \text{proj2-incident } q \ l \land \text{proj2-incident } r \ l)$

proof

assume $\exists \ q \ r. \ q \neq r \land q \in S \land r \in S \land \text{proj2-incident } q \ l \land \text{proj2-incident } r \ l$

then obtain $q$ where $q \in S$ and $\text{proj2-incident } q \ l$ by auto

from $(\text{proj2-incident } q \ \l)$ and $(\forall \ q. \ \text{proj2-incident } q \ l \longrightarrow q \notin S)$

have $q \notin S$ by simp

with $(q \in S)$ show False by simp

qed

with $(\text{proj2-incident } p \ \l)$

show $\neg (\forall \ l. \ \text{proj2-incident } p \ l \longrightarrow (\exists \ q \ r. \ q \neq r \land q \in S \land r \in S \land \text{proj2-incident } q \ l \land \text{proj2-incident } r \ l))$
\[ q \neq r \land q \in S \land r \in S \land \text{proj2-incident } q l \land \text{proj2-incident } r l \) 
by auto 
qed
with \( \forall l. \text{proj2-incident } p l \rightarrow (\exists q r. q \neq r \land q \in S \land r \in S \land \text{proj2-incident } q l \land \text{proj2-incident } r l) \)
show False by simp 
qed 

lemma line-through-hyp2-pole-not-in-hyp2:
assumes a \( \in \) hyp2 and proj2-incident a l 
shows pole l \( \notin \) hyp2 
proof 
from assms and line-through-K2-intersect-S 
obtain p where p \( \in \) S and proj2-incident p l by auto 
from \( \langle \text{proj2-incident } p l \rangle \) have proj2-incident (pole l) (polar p) by (rule incident-pole-polar) 
with \( \langle p \in S \rangle \) 
show pole l \( \notin \) hyp2 
by (auto simp add: tangent-not-through-K2) 
qed 

lemma statement60-one-way:
assumes is-K2-isometry J and p \( \in \) K2 
shows apply-cltn2 p J \( \in \) K2 (is ?p' \( \in \) K2) 
proof 
let ?J' = cltn2-inverse J 
have \( \forall l'. \text{proj2-incident } ?p' l' \rightarrow (\exists q' r'. q' \neq r' \land q' \in S \land r' \in S \land \text{proj2-incident } q' l' \land \text{proj2-incident } r' l') \) 
proof 
fix l' 
let \( l' = \) apply-cltn2-line l' ?J' 
show proj2-incident ?p' l' \( \rightarrow (\exists q' r'. q' \neq r' \land q' \in S \land r' \in S \land \text{proj2-incident } q' l' \land \text{proj2-incident } r' l') \) 
proof 
assume proj2-incident ?p' l' 
 hence proj2-incident p ?l 
 by (simp add: apply-cltn2-incident [of p l' ?J'] 
 cltn2.inv-inv [simplified]) 
with \( \langle p \in K2 \rangle \) and line-through-K2-intersect-S-twice [of p ?l] 
obtain q and r where q \( \neq r \) and q \( \in \) S and r \( \in \) S 
 and proj2-incident q ?l and proj2-incident r ?l 
 by auto 
let \( ?q' = \) apply-cltn2 q J 
let \( ?r' = \) apply-cltn2 r J 
from \( q \neq r \) and apply-cltn2-injective [of q J r] have ?q' \( \neq \) ?r' by auto 
from \( \langle q \in S \rangle \) and \( \langle r \in S \rangle \) and (is-K2-isometry J)
have \( ?q' \in S \) and \( ?r' \in S \) by (unfold is-K2-isometry-def) simp-all

from ⟨proj2-incident q ?l⟩ and ⟨proj2-incident r ?l⟩ have proj2-incident ?q' l' and proj2-incident ?r' l'
  by (simp-all add: apply-cltn2-incident [of - l' ?J]
cltn2.inv-inv [simplified])
with ⟨\( ?q' \neq ?r' \) and \( ?q' \in S \) and \( ?r' \in S \)
show \( \exists q', r'. \) \( q' \neq r' \land q' \in S \land r' \in S \land proj2-incident q' l' \land proj2-incident r' l' \)
  by auto

qed

thus \( ?p' \in K2 \) by (rule lines-through-intersect-S-twice-in-K2)

qed

lemma is-K2-isometry-hyp2-S:
  assumes \( p \in hyp2 \cup S \) and is-K2-isometry J
  shows apply-cltn2 p J \in hyp2 \cup S
proof cases
  assume \( p \in hyp2 \)
  with (is-K2-isometry J)
  have apply-cltn2 p J \in hyp2 by (rule statement60-one-way)
  thus apply-cltn2 p J \in hyp2 \cup S ..
next
  assume \( p \notin hyp2 \)
  with \( p \in hyp2 \cup S \)
  have \( p \in S \) by simp
  with (is-K2-isometry J)
  have apply-cltn2 p J \in hyp2 \cup S by (unfold is-K2-isometry-def) simp
  thus apply-cltn2 p J \in hyp2 \cup S ..

qed

lemma is-K2-isometry-z-non-zero:
  assumes \( p \in hyp2 \cup S \) and is-K2-isometry J
  shows z-non-zero (apply-cltn2 p J)
proof –
  from \( p \in hyp2 \cup S \) and (is-K2-isometry J)
  have apply-cltn2 p J \in hyp2 \cup S by (rule is-K2-isometry-hyp2-S)
  thus z-non-zero (apply-cltn2 p J) by (rule hyp2-S-z-non-zero)

qed

lemma cart2-append1-apply-cltn2:
  assumes \( p \in hyp2 \cup S \) and is-K2-isometry J
  shows \( \exists k. \ k \neq 0 \land cart2-append1 p v* cltn2-rep J = k \ast_R cart2-append1 (apply-cltn2 p J) \)
proof –
  have cart2-append1 p v* cltn2-rep J
    = \((1 / (proj2-rep p)\$3) \ast_R (proj2-rep p v* cltn2-rep J)\)
    by (unfold cart2-append1-def) (simp add: scaleR-vector-matrix-assoc)

139
from ⟨p ∈ hyp2 ∪ S⟩ have (proj2-rep p)$3 ≠ 0 by (rule hyp2-S-z-non-zero)

from apply-cltn2-imp-mult [of p J]
obtain j where j ≠ 0
  and proj2-rep p v* cltn2-rep J = j *R proj2-rep (apply-cltn2 p J)
  by auto

from ⟨p ∈ hyp2 ∪ S⟩ and ⟨is-K2-isometry J⟩ have z-non-zero (apply-cltn2 p J)
by (rule is-K2-isometry-z-non-zero)

by (rule proj2-rep-cart2-append1)

let ?k = 1 / (proj2-rep p)$3 * j * (proj2-rep (apply-cltn2 p J))$3
from ⟨proj2-rep p)$3 ≠ 0; and (j ≠ 0)⟩ and ⟨(proj2-rep (apply-cltn2 p J))$3 ≠ 0⟩
have ?k ≠ 0 by simp

from ⟨cart2-append1 p v* cltn2-rep J
= (1 / (proj2-rep p)$3) *R (proj2-rep p v* cltn2-rep J),
and (proj2-rep p v* cltn2-rep J) = j *R proj2-rep (apply-cltn2 p J)⟩
have cart2-append1 p v* cltn2-rep J
= (1 / (proj2-rep p)$3 * j) *R proj2-rep (apply-cltn2 p J)
by simp

from ⟨proj2-rep (apply-cltn2 p J)
= (proj2-rep (apply-cltn2 p J))$3 *R cart2-append1 (apply-cltn2 p J)),
have (1 / (proj2-rep p)$3 * j) *R proj2-rep (apply-cltn2 p J)
= (1 / (proj2-rep p)$3 * j) *R ((proj2-rep (apply-cltn2 p J))$3 *R cart2-append1 (apply-cltn2 p J))
by simp

with ⟨cart2-append1 p v* cltn2-rep J
= (1 / (proj2-rep p)$3 * j) *R proj2-rep (apply-cltn2 p J)⟩,
have cart2-append1 p v* cltn2-rep J = ?k *R cart2-append1 (apply-cltn2 p J)
by simp

with ⟨?k ≠ 0⟩
show ∃ k, k ≠ 0
  ∧ cart2-append1 p v* cltn2-rep J = k *R cart2-append1 (apply-cltn2 p J)
by (simp add: ex1 [of - ?k])
qed

8.5 The K-isometries form a group action

lemma hyp2-cltn2-id [simp]: hyp2-cltn2 p cltn2-id = p
  by (unfold hyp2-cltn2-def) (simp add: Rep-hyp2-inverse)

lemma apply-cltn2-Rep-hyp2:
  assumes is-K2-isometry J
  shows apply-cltn2 (Rep-hyp2 p) J ∈ hyp2
proof –
from (is-K2-isometry J) and Rep-hyp2 [of p]
show apply-cltn2 (Rep-hyp2 p) J ∈ K2 by (rule statement60-one-way)
qed

lemma Rep-hyp2-cltn2:
assumes is-K2-isometry J
shows Rep-hyp2 (hyp2-cltn2 p J) = apply-cltn2 (Rep-hyp2 p) J
proof –
from (is-K2-isometry J)
have apply-cltn2 (Rep-hyp2 p) J ∈ hyp2 by (rule apply-cltn2-Rep-hyp2)
thus Rep-hyp2 (hyp2-cltn2 p J) = apply-cltn2 (Rep-hyp2 p) J
  by (unfold hyp2-cltn2-def) (rule Abs-hyp2-inverse)
qed

lemma hyp2-cltn2-compose:
assumes is-K2-isometry H
shows hyp2-cltn2 (hyp2-cltn2 p H) J = hyp2-cltn2 p (cltn2-compose H J)
proof –
from (is-K2-isometry H)
have apply-cltn2 (Rep-hyp2 p) H ∈ hyp2 by (rule apply-cltn2-Rep-hyp2)
thus hyp2-cltn2 (hyp2-cltn2 p H) J = hyp2-cltn2 p (cltn2-compose H J)
  by (unfold hyp2-cltn2-def) (simp add: Abs-hyp2-inverse apply-cltn2-compose)
qed

interpretation K2-isometry: action
|carrier = Collect is-K2-isometry, mult = cltn2-compose, one = cltn2-id|
hyp2-cltn2
proof
let ?G =
  (|carrier = Collect is-K2-isometry, mult = cltn2-compose, one = cltn2-id|)
fix p
show hyp2-cltn2 p 1?G = p
  by (unfold hyp2-cltn2-def) (simp add: Rep-hyp2-inverse)
fix H J
show H ∈ carrier ?G ∧ J ∈ carrier ?G
  → hyp2-cltn2 (hyp2-cltn2 p H) J = hyp2-cltn2 p (H ⊗ ?G J)
  by (simp add: hyp2-cltn2-compose)
qed

8.6 The Klein–Beltrami model satisfies Tarski’s first three axioms

lemma three-in-S-tangent-intersection-no-3-Col:
assumes p ∈ S and q ∈ S and r ∈ S
and p ≠ q and r ∉ {p,q}
shows proj2-no-3-Col {proj2-intersection (polar p) (polar q),r,p,q}
(is proj2-no-3-Col {?s,r,p,q})
let \(?T = \{s,r,p,q\}\)

from \(p \neq q\) have card \(\{p,q\} = 2\) by simp
with \(r \notin \{p,q\}\) have card \(\{r,p,q\} = 3\) by simp

from \(p \in S\) and \(q \in S\) and \(r \in S\) have \(\{r,p,q\} \subseteq S\) by simp

have proj2-incident \(?s\) (polar \(p\)) and proj2-incident \(?s\) (polar \(q\))
by (rule proj2-intersection-incident+)

have \(?s \notin S\)
proof
assume \(?s \in S\)
with \(p \in S\) and proj2-incident \(?s\) (polar \(p\)):
and \(q \in S\) and proj2-incident \(?s\) (polar \(q\)):
have \(?s = p\) and \(?s = q\) by (simp-all add: point-in-S-polar-is-tangent)
then obtain \(i\) where \(\forall a \in (\{p,q\} = 3\). proj2-incident a i\)
unfolding proj2-set-Col-def ..

from proj2-set-Col (\(p,q\} \subseteq S\) have \(?s \notin \{p,q\}\) by auto
with card \(\{r,p,q\} = 3\) have card \(\{s,r,p,q\} = 4\) by simp

have \(\forall t \in \?T. \neg proj2-set-Col (\?T - \{t\})\)
proof standard+
fix \(t\)
assume \(t \in \?T\)
assume proj2-set-Col (\(\?T - \{t\}\))
then obtain \(i\) where \(\forall a \in (\?T - \{t\}). proj2-incident a i\)
unfolding proj2-set-Col-def ..

from proj2-set-Col (\(\?T - \{t\}\))
have proj2-set-Col \((S \cap (\?T - \{t\}))\)
by (simp add: proj2-subset-Col [of \((S \cap (\?T - \{t\}))\) \(\?T - \{t\})])
hence card \((S \cap (\?T - \{t\}))\) \(\leq 2\) by (simp add: card-line-intersect-S)

show False
proof cases
assume \(t = \?s\)
with \(?s \notin \{p,q\}\) have \(\?T - \{t\} = \{r,p,q\}\) by simp
with \(\{r,p,q\} \subseteq S\) have \(S \cap (\?T - \{t\}) = \{r,p,q\}\) by simp
with card \(\{r,p,q\} = 3\) and card \((S \cap (\?T - \{t\}))\) \(\leq 2\) show False by simp

next
assume \(t \neq \?s\)
hence \(?s \in \?T - \{t\}\) by simp
with \(\forall a \in (\?T - \{t\}). proj2-incident a i\) have proj2-incident \(?s\) \(?s\) ..

from \(p \neq q\) have \(\{p,q\} \cap (\?T - \{t\}) \neq \{\}\) by auto
then obtain \(d\) where \(d \in \{p,q\}\) and \(d \in (\?T - \{t\})\) by auto
from \(d \in ?T - \{t\}\) and \(\forall a \in (?T - \{t\})\), proj2-incident a l

have proj2-incident d l by simp

from \(\langle d \in \{p.q\}\rangle\);
and (proj2-incident ?s (polar p));
and (proj2-incident ?s (polar q));

have proj2-incident ?s (polar d) by auto

from \(\langle d \in \{p.q\}\rangle\) and \(\langle r,p.q \rangle \subseteq S\) have \(d \in S\) by auto

hence proj2-incident d (polar d) by (unfold incident-own-polar-in-S)

from \(\langle d \in S\rangle\) and \(\langle ?s \notin S\rangle\) have \(d \neq ?s\) by auto

with (proj2-incident ?s l);
and (proj2-incident d l);
and (proj2-incident ?s (polar d));
and (proj2-incident d (polar d));

and proj2-incident-unique

have \(l = \text{polar} d\) by auto

with \(d \in S\) and point-in-S-polar-is-tangent

have \(\forall a \in S\). proj2-incident a l \(\rightarrow a = d\) by simp

with \(\forall a \in (?T - \{t\})\). proj2-incident a l

have \(S \cap (?T - \{t\}) \subseteq \{d\}\) by auto

with card-mono [of \(\{d\}\)] have \(\text{card} (S \cap (?T - \{t\})) \leq 1\) by simp

hence \(\text{card} ((S \cap ?T) - \{t\}) \leq 1\) by (simp add: Int-Diff)

have \(S \cap ?T \subseteq \text{insert} t ((S \cap ?T) - \{t\})\) by auto

with card-suc-ge-insert [of \(t ((S \cap ?T) - \{t\})\)]
and card-mono [of \(\text{insert} t ((S \cap ?T) - \{t\}) S \cap ?T\)]

have \(\text{card} (S \cap ?T) \leq \text{card} ((S \cap ?T) - \{t\}) + 1\) by simp

with \(\langle \text{card} ((S \cap ?T) - \{t\}) \leq 1\rangle\) have \(\text{card} (S \cap ?T) \leq 2\) by simp

from \(\langle r,p,q \rangle \subseteq S\) have \(\langle r,p,q \rangle \subseteq S \cap ?T\) by simp

with \(\langle \text{card} (r,p,q) = 3\rangle\) and card-mono [of \(S \cap ?T\) \(\langle r,p,q \rangle\)]

have \(\text{card} (S \cap ?T) \geq 3\) by simp

with \(\langle \text{card} (S \cap ?T) \leq 2\rangle\) show False by simp

qed

with \(\langle \text{card} ?T = 4\rangle\) show proj2-no-3-Col ?T unfolding proj2-no-3-Col-def ..

qed

lemma statement65-special-case:
assumes \(p \in S\) and \(q \in S\) and \(r \in S\) and \(p \neq q\) and \(r \notin \{p,q\}\)
shows \(\exists J. \text{is-K2-isometry} J\)
\& apply-cltn2 east \(J = p\)
\& apply-cltn2 west \(J = q\)
\& apply-cltn2 north \(J = r\)
\& apply-cltn2 far-north \(J = \text{proj2-intersection} (\text{polar} p) (\text{polar} q)\)

proof –
let \(?s = \text{proj2-intersection} (\text{polar} p) (\text{polar} q)\)
have range \((\forall \ t \ (\forall \ s \ t \in S))\) = \{\ s, r, p, q\}

unfolding `image_def`
by (auto simp add: `UNIV-4 vector-4`)

with \( p \in S \) and \( q \in S \) and \( r \in S \) and \( p \neq q \) and \( r \notin \{p, q\} \)
have \(\text{proj2-no-3-Col} \ (\text{range} \ ((\forall \ t \ (\forall \ s \ t \in S)))\) by simp

ultimately have \(\forall \ i. \ \text{proj2-no-3-Col} \ (\text{range} \ ((\forall \ t \ (\forall \ s \ t \in S)))\) by simp
\(\exists \ J. \forall \ j. \ \text{apply-cltn2} \ (\forall \ t \ (\forall \ s \ t \in S) \ J = \forall \ t \ (\forall \ s \ t \in S)\) by (rule `statement53-existence`

moreover have \( 0 = (2::2) \) by simp
ultimately obtain \( J \) where \(\forall \ j. \ \text{apply-cltn2} \ (\forall \ t \ (\forall \ s \ t \in S) \ J = \forall \ t \ (\forall \ s \ t \in S)\) by auto

hence \(\text{apply-cltn2} \ (\forall \ t \ (\forall \ s \ t \in S) \ J = \forall \ t \ (\forall \ s \ t \in S)\)

and \(\text{apply-cltn2} \ (\forall \ t \ (\forall \ s \ t \in S) \ J = \forall \ t \ (\forall \ s \ t \in S)\)

and \(\text{apply-cltn2} \ (\forall \ t \ (\forall \ s \ t \in S) \ J = \forall \ t \ (\forall \ s \ t \in S)\)

by simp-all

hence \(\text{apply-cltn2} \ east \ J = p\)

and \(\text{apply-cltn2} \ west \ J = q\)

and \(\text{apply-cltn2} \ north \ J = r\)

by (simp-all add: `vector-2 vector-4`)

with compass-non-zero
have \( p = \text{proj2-abs} \ (\text{vector} \ [1,0,1] \ v = \text{cltn2-rep} \ J)\)

and \( q = \text{proj2-abs} \ (\text{vector} \ [-1,0,1] \ v = \text{cltn2-rep} \ J)\)

and \( r = \text{proj2-abs} \ (\text{vector} \ [0,1,1] \ v = \text{cltn2-rep} \ J)\)

and \( s = \text{proj2-abs} \ (\text{vector} \ [0,1,0] \ v = \text{cltn2-rep} \ J)\)

unfolding compass-defs and far-north-def
by (simp-all add: `apply-cltn2-left-abs`)

let \( ?N = \text{cltn2-rep} \ J \)** \( M \)** transpose \( (\text{cltn2-rep} \ J)\)

from `M-symmatrix` have `symmatrix ?N` by (rule `symmatrix-preserve`

hence \(?N\$\$\$1 = ?N\$\$\$2 \ and \ ?N\$\$\$3 = ?N\$\$\$1 \ and \ ?N\$\$\$2 = ?N\$\$\$3`

unfolding `symmatrix-def` and transpose-def
by (simp-all add: `vec-eq-iff`

from compass-non-zero and \(\text{apply-cltn2} \ east \ J = p\) and \( p \in S\)

and \(\text{apply-cltn2-abs-in-S} \ (\text{of vector} \ [1,0,1] \ J)\)

have \(\text{vector} \ [1,0,1] :: \text{real} \cdot (\forall \ N \ v \ (\text{vector} \ [1,0,1]) = 0\)

unfolding `east-def`

144
by simp
hence ?N$1$ + ?N$3$ + ?N$3$ = 0
unfolding inner-vec-def and matrix-vector-mult-def
by (simp add: sum-3 vector-3)
with (?N$3$ = ?N$1$) have ?N$1$ + 2 * (?N$1$) + ?N$3$ = 0 by simp

from compass-non-zero and (apply-cltn west J) and (q ∈ S)
and apply-cltn-abs-in-S [of vector [-1,0,1] J]
have (vector [-1,0,1] :: real^3) · (?N *v vector [-1,0,1]) = 0
unfolding west-def
by simp
hence ?N$1$ - ?N$3$ - ?N$3$ = 0
unfolding inner-vec-def and matrix-vector-mult-def
by (simp add: sum-3 vector-3)
with (?N$3$ = ?N$1$) have ?N$1$ - 2 * (?N$1$) + ?N$3$ = 0 by simp

with (?N$1$ + 2 * (?N$1$) + ?N$3$ = 0)
have ?N$1$ - 2 * (?N$1$) + ?N$3$ = 0
by simp

from compass-non-zero and (apply-cltn north J) and (r ∈ S)
and apply-cltn-abs-in-S [of vector [0,1,1] J]
have (vector [0,1,1] :: real^3) · (?N *v vector [0,1,1]) = 0
unfolding north-def
by simp
hence ?N$2$ + ?N$3$ + ?N$3$ = 0
unfolding inner-vec-def and matrix-vector-mult-def
by (simp add: sum-3 vector-3)
with (?N$3$ = ?N$2$) have ?N$2$ + 2 * (?N$2$) + ?N$3$ = 0 by simp

have proj2-incident ?s (polar p) and proj2-incident ?s (polar q)
by (rule proj2-intersection-incident+)

from compass-non-zero
have vector [1,0,1] * v* cltn2-rep J ≠ 0
and vector [-1,0,1] * v* cltn2-rep J ≠ 0
and vector [0,1,0] * v* cltn2-rep J ≠ 0
by (simp-all add: non-zero-mult-rep-non-zero)
from (vector [1,0,1] * v* cltn2-rep J ≠ 0)
and (vector [-1,0,1] * v* cltn2-rep J ≠ 0)
and p = proj2-abs (vector [1,0,1] * v* cltn2-rep J)
and q = proj2-abs (vector [-1,0,1] * v* cltn2-rep J)
have polar p = proj2-bine-abs (M *v (vector [1,0,1] * v* cltn2-rep J))
and $\text{polar } q = \text{proj2-line-abs} \begin{pmatrix} M \ast v \begin{pmatrix} -1,0,1 \end{pmatrix} \ast \text{cltn2-rep } J \end{pmatrix}$
by $(\text{simp-all add: polar-abs})$

from $(\text{vector } [1,0,1] \ast v \ast \text{cltn2-rep } J \neq 0)$
and $(\text{vector } [-1,0,1] \ast v \ast \text{cltn2-rep } J \neq 0)$
and $M$-invertible
have $M \ast v \begin{pmatrix} [1,0,1] \ast v \ast \text{cltn2-rep } J \neq 0$
and $M \ast v \begin{pmatrix} [-1,0,1] \ast v \ast \text{cltn2-rep } J \neq 0$
by $(\text{simp-all add: invertible-times-non-zero})$

with $(\text{polar } [0,1,0] \ast v \ast \text{cltn2-rep } J \neq 0)$
and $(\text{polar } p = \text{proj2-line-abs} \begin{pmatrix} M \ast v \begin{pmatrix} [1,0,1] \ast v \ast \text{cltn2-rep } J \end{pmatrix} \end{pmatrix})$
and $(\text{polar } q = \text{proj2-line-abs} \begin{pmatrix} M \ast v \begin{pmatrix} [-1,0,1] \ast v \ast \text{cltn2-rep } J \end{pmatrix} \end{pmatrix})$
and $(?s = \text{proj2-abs} \begin{pmatrix} \text{vector } [0,1,0] \ast v \ast \text{cltn2-rep } J \end{pmatrix})$

have $\text{proj2-incident } ?s (\text{polar } p)$
\begin{itemize}
  \item $(M \ast v \begin{pmatrix} [1,0,1] \ast v \ast \text{cltn2-rep } J \end{pmatrix}) = 0$
  \item $(\text{proj2-incident } ?s (\text{polar } q))$
\end{itemize}
by $(\text{simp-all add: proj2-incident-abs})$

with $(\text{proj2-incident } ?s (\text{polar } p))$ and $(\text{proj2-incident } ?s (\text{polar } q))$

have $(\text{vector } [0,1,0] \ast v \ast \text{cltn2-rep } J)$
\begin{itemize}
  \item $(M \ast v \begin{pmatrix} [1,0,1] \ast v \ast \text{cltn2-rep } J \end{pmatrix}) = 0$
  \item $(\text{vector } [0,1,0] \ast v \ast \text{cltn2-rep } J)$
\end{itemize}
by $(\text{simp-all add: dot-bmul-matrix matrix-vector-mult-assoc [symmetric]})$

hence $(?N \text{\$2\$1} + (?N \text{\$2\$2}) + ?N \text{\$2\$3} = 0)$ and $-(?N \text{\$2\$1}) + ?N \text{\$2\$2} = 0$

unfolding inner-vec-def and matrix-vector-mult-def
by $(\text{simp-all add: sum-3 vector-3})$

hence $(?N \text{\$2\$1} + ?N \text{\$2\$2} = -(?N \text{\$2\$1}) + ?N \text{\$2\$3} \text{ by simp})$

hence $(?N \text{\$2\$1} = 0 \text{ by simp})$
with $(?N \text{\$2\$1} + ?N \text{\$2\$2} = 0), \text{ have } ?N \text{\$2\$3} = 0 \text{ by simp})$
with $(?N \text{\$2\$2} + 2 \ast (?N \text{\$2\$3}) + ?N \text{\$3\$3} = 0)$ and $(?N \text{\$3\$3} = -(?N \text{\$1\$1}))$

have $(?N \text{\$2\$2} = ?N \text{\$1\$1} \text{ by simp})$
with $(?N \text{\$1\$3} = 0)$ and $(?N \text{\$2\$1} = ?N \text{\$1\$2})$ and $(?N \text{\$1\$3} = 0)$
and $(?N \text{\$2\$1} = 0)$ and $(?N \text{\$2\$2} = ?N \text{\$1\$1})$ and $(?N \text{\$2\$3} = 0)$
and $(?N \text{\$3\$1} = ?N \text{\$1\$3})$ and $(?N \text{\$3\$2} = ?N \text{\$2\$3})$ and $(?N \text{\$3\$3} = -(?N \text{\$1\$1}))$

have $?N = (?N \text{\$1\$1} \ast R M$

unfolding $M$-def
by $(\text{simp add: vec-eq-iff vector-3 forall-3})$

have $\text{invertible } (\text{cltn2-rep } J)$ by $(\text{rule cltn2-rep-invertible})$
with $M$-invertible
have $\text{invertible } ?N$ by $(\text{simp add: invertible-mult transpose-invertible})$

hence $?N \neq 0 \text{ by (auto simp add: zero-not-invertible)}$
with \( \langle ?N = (\langle \langle ?N \$1 \rangle \rangle) \star R M \rangle \) have \( \langle \langle ?N \$1 \rangle \rangle \neq 0 \) by auto
with \( \langle ?N = (\langle \langle ?N \$1 \rangle \rangle) \star R M \rangle \) have is-K2-isometry (cltn2-abs (cltn2-rep J))
  by (simp add: J-M-J-transpose-K2-isometry)
hence is-K2-isometry J by (simp add: cltn2-abs-rep)
with (apply-cltn2 east J = p)
  and (apply-cltn2 west J = q)
  and (apply-cltn2 north J = r)
  and (apply-cltn2 far-north J = \(?s\))
show \( \exists J. \) is-K2-isometry J \\
  \& apply-cltn2 east J = p \\
  \& apply-cltn2 west J = q \\
  \& apply-cltn2 north J = r \\
  \& apply-cltn2 far-north J = \(?s\)
  by auto
qed

lemma statement66-existence:
  assumes a1 \( \in \) K2 and a2 \( \in \) K2 and p1 \( \in \) S and p2 \( \in \) S
  shows \( \exists J. \) is-K2-isometry J \\
  \& apply-cltn2 a1 J = a2 \& apply-cltn2 p1 J = p2
proof –
  let \(?a = vector [a1,a2] :: proj2\^2\)
  from \(\langle a1 \in K2 \rangle \) and \(\langle a2 \in K2 \rangle \) have \(\forall i. \) a\$i \( \in \) K2 by (simp add: forall-2)
  let \(?p = vector [p1,p2] :: proj2\^2\)
  from \(\langle p1 \in S \rangle \) and \(\langle p2 \in S \rangle \) have \(\forall i. \) p\$i \( \in \) S by (simp add: forall-2)
  let \(?l = \chi i. \) proj2-line-through (?a\$i) (?p\$i)
  have \(\forall i. \) proj2-incident (?a\$i) (?l\$i)
    by (simp add: proj2-line-through-incident)
  hence proj2-incident (?a\$1) (?l\$1) and proj2-incident (?a\$2) (?l\$2)
    by fast+
  have \(\forall i. \) proj2-incident (?p\$i) (?l\$i)
    by (simp add: proj2-line-through-incident)
  hence proj2-incident (?p\$1) (?l\$1) and proj2-incident (?p\$2) (?l\$2)
    by fast+
  let \(?q = \chi i. \) \& proj2-incident (?q\$i) (?l\$i)
  have \(\forall i. \) \& proj2-incident (?q\$i) (?l\$i)
    by (rule line-through-K2-intersect-S-again)
with someI-ex [of \( \lambda \ q\!\!. \ q\!\! \neq \ ?p\!\!\! i \wedge q\!\!i \in S \wedge proj2-incident q\!\!i \)]
show \( \forall \) \( ?q\!\!i \neq \ ?p\!\!\! i \wedge \ ?q\!\!i \in S \wedge proj2-incident \ (?)q\!\!i \) \( (?!i) \) by simp qed

hence \( ?q\!\!1 \neq \ ?p\!\!1 \) \textit{and} proj2-incident \( (?)q\!\!1 \) \( (?!1) \)
and proj2-incident \( (?)q\!\!2 \) \( (?!2) \)
by fast+

let \( ?r = \chi i \). proj2-intersection \( (\text{polar} \ (?)q\!\!i) \) \( (\text{polar} \ (?)p\!\!i) \)
let \( ?m = \chi i \). proj2-line-through \( (\text{?a}!i) \) \( (?r\!i) \)
have \( \forall \) \( i \). proj2-incident \( (\text{?a}!i) \) \( (?)m\!i \)
by \( \text{(simp add: proj2-line-through-incident)} \)
hence proj2-incident \( (\text{?a}!1) \) \( (?)m\!1 \) \textit{and} proj2-incident \( (\text{?a}!2) \) \( (?)m\!2 \)
by fast+

have \( \forall \) \( i \). proj2-incident \( (?)r\!i \) \( (?)m\!i \)
by \( \text{(simp add: proj2-line-through-incident)} \)
hence proj2-incident \( (?)r\!1 \) \( (?)m\!1 \) \textit{and} proj2-incident \( (?)r\!2 \) \( (?)m\!2 \)
by fast+

let \( ?s = \chi i \). \( \epsilon \) \( si \). \( si \neq \ ?r\!i \wedge \ ?s\!i \in S \wedge proj2-incident \ (?)s\!i \)
have \( \forall \) \( i \). \( ?s\!i \neq \ ?r\!i \wedge \ ?s\!i \in S \wedge proj2-incident \ (?)s\!i \) \( (?)m\!i \)
proof
fix \( i \)
from \( \forall \) \( i \). \( ?a\!i \in K2 \). have \( ?a\!i \in K2 \) ..

from \( \forall \) \( i \). proj2-incident \( (\text{?a}!i) \) \( (?)m\!i \)
have proj2-incident \( (\text{?a}!i) \) \( (?)m\!i \) ..
with \( (\text{?a}!i \in K2) \)
have \( \exists \) \( si \). \( si \neq \ ?r\!i \wedge \ ?s\!i \in S \wedge proj2-incident \ (?)s\!i \)
by \( \text{(rule line-through-K2-intersect-S-again)} \)
with someI-ex [of \( \lambda \ si \). \( si \neq \ ?r\!i \wedge \ ?s\!i \in S \wedge proj2-incident \ (?)s\!i \)]
show \( ?s\!i \neq \ ?r\!i \wedge ?s\!i \in S \wedge proj2-incident \ (?)s\!i \) \( (?)m\!i \) by simp qed

hence \( ?s\!1 \neq \ ?r\!1 \) \textit{and} proj2-incident \( (?)s\!1 \) \( (?)m\!1 \)
and proj2-incident \( (?)s\!2 \) \( (?)m\!2 \)
by fast+

have \( \forall \) \( i \) . \( \forall \) \( u \). proj2-incident \( u \) \( (?)m\!i \) \( \longrightarrow \) \( \neg (u = ?p\!i \vee u = ?q\!i) \)
proof standard+
fix \( i :: 2 \)
fix \( u :: \text{proj2} \)
assume proj2-incident \( u \) \( (?)m\!i \)
assume \( u = ?p\!i \vee u = ?q\!i \)
from \( \forall \) \( i \). \( ?p\!i \in S \). have \( ?p\!i \in S \) ..

from \( \forall \) \( i \). \( ?q\!i \neq ?p\!i \wedge ?q\!i \in S \wedge proj2-incident \ (?)q\!i \) \( (?)l\!i \)
have \( ?q\!i \neq ?p\!i \) \textit{and} \( ?q\!i \in S \)
by simp-all

148
from \( \{p_i \in S\} \) and \( \{q_i \in S\} \) and \( u = p_i \lor u = q_i \)
have \( u \in S \) by auto
hence proj2-incident \( u \) (polar \( u \))
by (simp add: incident-own-polar-in-S)

have proj2-incident \( (r_i) \) (polar \( (p_i) \))
and proj2-incident \( (r_i) \) (polar \( (q_i) \))
by (simp-all add: proj2-intersection-incident)
with \( u = p_i \lor u = q_i \)
have proj2-incident \( (r_i) \) (polar \( u \)) by auto

from \( \forall \ i. \ \text{proj2-incident} \ (r_i) \ (m_i) \)
have proj2-incident \( (r_i) \) \( (m_i) \) ..

from \( \forall \ i. \ \text{proj2-incident} \ (a_i) \ (m_i) \)
have proj2-incident \( (a_i) \) \( (m_i) \) ..

from \( \forall \ i. \ a_i \in K2 \) have \( a_i \in K2 \) ..

have \( u \neq r_i \)
proof
assume \( u = r_i \)
with (proj2-incident \( (r_i) \) (polar \( (p_i) \))):
and (proj2-incident \( (r_i) \) (polar \( (q_i) \))):
have proj2-incident \( u \) (polar \( (p_i) \))
and proj2-incident \( u \) (polar \( (q_i) \))
by simp-all
with \( u \in S \) and \( p_i \in S \) and \( q_i \in S \)
have \( u = p_i \) and \( u = q_i \)
by (simp-all add: point-in-S-polar-is-tangent)
with \( q_i \neq p_i \) show False by simp
qed

with proj2-incident \( (u) \) (polar \( u \)):
and (proj2-incident \( (r_i) \) (polar \( u \))):
and (proj2-incident \( (m_i) \) (polar \( u \))):
and proj2-incident-unique
have \( m_i \) = polar \( u \) by auto
with (proj2-incident \( (a_i) \) \( (m_i) \)):
have proj2-incident \( (a_i) \) (polar \( u \)) by simp
with \( u \in S \) and \( a_i \in K2 \) and tangent-not-through-K2
show False by simp
qed

let \( \chi \ i. \in Hi \) is-K2-isometry \( Hi \)
\& apply-cltn2 east \( Hi \) = \( q_i \)
\& apply-cltn2 west \( Hi \) = \( p_i \)
\& apply-cltn2 north \( Hi \) = \( s_i \)
\[\forall \ i. \ is-K2-isometry \ (\forall \ H \ S \ i. \ ?q S i)\]
\[\forall \ i. \ is-K2-isometry \ (\forall \ H \ S \ i. \ ?p S i)\]
\[\forall \ i. \ is-K2-isometry \ (\forall \ H \ S \ i. \ ?s S i)\]
\[\forall \ i. \ is-K2-isometry \ (\forall \ H \ S \ i. \ ?r S i)\]

**Proof**

fix \(i :: 2\)

from \(\forall \ i. \ ?p S i \in S \land \exists \ ?r S i \in S \land \text{proj2-incident} \ (\exists \ ?q S i) \ (\exists \ ?l S i)\)
have \(?q S i \neq \ ?p S i \land ?q S i \in S \land \text{proj2-incident} \ (?q S i) \ (\exists \ ?l S i)\)
by simp-all

from \(\forall \ i. \ ?s S i \neq \ ?r S i \land \exists \ ?s S i \in S \land \text{proj2-incident} \ (?s S i) \ (\exists \ ?m S i)\)
have \(?s S i \in S \land \exists \ ?m S i \land \text{proj2-incident} \ (?s S i) \ (?m S i)\)
by simp-all

from \(?s S i \in \{ ?q S i, \ ?p S i \}\) by fast
with \(?q S i \in S \land \exists \ ?p S i \in S \land ?s S i \in S \land \exists \ ?q S i \neq \ ?p S i\)

have \(?i. \ is-K2-isometry \ Hi\)
\[\forall \ i. \ apply-cltn2 \ far-north \ Hi = ?r S i\]
\[\forall \ i. \ apply-cltn2 \ west \ Hi = ?p S i\]
\[\forall \ i. \ apply-cltn2 \ north \ Hi = ?s S i\]
\[\forall \ i. \ apply-cltn2 \ far-north \ Hi = ?r S i\]
by (simp add: statement65-special-case)

with someI-ex \(\exists \ ?i. \ is-K2-isometry \ Hi\)
\[\forall \ i. \ apply-cltn2 \ far-north \ Hi = ?r S i\]
\[\forall \ i. \ apply-cltn2 \ west \ Hi = ?p S i\]
\[\forall \ i. \ apply-cltn2 \ north \ Hi = ?s S i\]

show \(?i. \ is-K2-isometry \ (\exists \ ?H \ S \ i. \ ?q S i)\)
\[\forall \ i. \ apply-cltn2 \ far-north \ (\exists \ ?H \ S \ i. \ ?q S i) = ?r S i\]
\[\forall \ i. \ apply-cltn2 \ west \ (\exists \ ?H \ S \ i. \ ?q S i) = ?p S i\]
\[\forall \ i. \ apply-cltn2 \ north \ (\exists \ ?H \ S \ i. \ ?q S i) = ?s S i\]

by simp

qed

hence \(?i. \ is-K2-isometry \ (\forall \ ?H \ S \ i. \ ?q S i)\)
\[\forall \ i. \ apply-cltn2 \ far-north \ (?H \ S \ i) = ?r S 1\]
\[\forall \ i. \ apply-cltn2 \ west \ (?H \ S \ i) = ?p S 1\]
\[\forall \ i. \ apply-cltn2 \ north \ (?H \ S \ i) = ?s S 1\]
\[\forall \ i. \ apply-cltn2 \ far-north \ (?H \ S \ i) = ?r S 1\]

and \(?i. \ is-K2-isometry \ (\forall \ ?H \ S \ i. \ ?q S 2)\)
\[\forall \ i. \ apply-cltn2 \ far-north \ (?H \ S \ i) = ?r S 2\]
\[\forall \ i. \ apply-cltn2 \ west \ (?H \ S \ i) = ?p S 2\]
\[\forall \ i. \ apply-cltn2 \ north \ (?H \ S \ i) = ?s S 2\]

by fast+
let \( ?J = \text{cltn2-compose (cltn2-inverse (?H$1)) (?H$2)} \)
from (is-K2-isometry (?H$1)) and (is-K2-isometry (?H$2))
have is-K2-isometry ?J
  by (simp only: cltn2-inverse-is-K2-isometry cltn2-compose-is-K2-isometry)

from (apply-cltn2 west (?H$1) = ?p$1)
have apply-cltn2 p1 (cltn2-inverse (?H$1)) = west
  by (simp add: cltn2.act-inv-iff [simplified])
with (apply-cltn2 west (?H$2) = ?p$2)
have apply-cltn2 p1 ?J = p2
  by (simp add: cltn2.act [simplified, symmetric])

from (apply-cltn2 east (?H$1) = ?q$1)
have apply-cltn2 (?q$1) (cltn2-inverse (?H$1)) = east
  by (simp add: cltn2.act-inv-iff [simplified])
with (apply-cltn2 east (?H$2) = ?q$2)
have apply-cltn2 (?q$1) ?J = ?q$2
  by (simp add: cltn2.act [simplified, symmetric])
with (?q$1 \neq ?p$1) and (apply-cltn2 p1 ?J = p2; and (proj2-incident (?p$1) (?q$1));
  and (proj2-incident (?q$1) (?p$1));
  and (proj2-incident (?q$2) (?p$1));
  and (proj2-incident (?q$2) (?p$2));
have apply-cltn2-line (?q$1) ?J = (?q$2)
  by (simp add: apply-cltn2-line-unique)
moreover from (proj2-incident (?a$1) (?i$1))
have proj2-incident (apply-cltn2 (?a$1) ?J) (apply-cltn2-line (?i$1) ?J)
  by simp
ultimately have proj2-incident (apply-cltn2 (?a$1) ?J) (?i$2) by simp

from (apply-cltn2 north (?H$1) = ?s$1)
have apply-cltn2 (?s$1) (cltn2-inverse (?H$1)) = north
  by (simp add: cltn2.act-inv-iff [simplified])
with (apply-cltn2 north (?H$2) = ?s$2)
have apply-cltn2 (?s$1) ?J = ?s$2
  by (simp add: cltn2.act [simplified, symmetric])

from (apply-cltn2 far-north (?H$1) = ?r$1)
have apply-cltn2 (?r$1) (cltn2-inverse (?H$1)) = far-north
  by (simp add: cltn2.act-inv-iff [simplified])
with (apply-cltn2 far-north (?H$2) = ?r$2)
have apply-cltn2 (?r$1) ?J = ?r$2
  by (simp add: cltn2.act [simplified, symmetric])
with (?s$1 \neq ?r$1) and (apply-cltn2 (?s$1) ?J = (?s$2));
  and (proj2-incident (?r$1) (?m$1));
  and (proj2-incident (?r$1) (?m$1));
  and (proj2-incident (?r$2) (?m$2));
  and (proj2-incident (?r$2) (?m$2));

151
have $\text{apply-cltn2-line} \ ((?m\$1) \ ?J = (?m\$2))$
by (simp add: apply-cltn2-line-unique)

moreover from $\text{proj2-incident} \ ((?a\$1) \ (?m\$1))$
have $\text{proj2-incident} \ (\text{apply-cltn2} \ (?a\$1) \ ?J) \ (\text{apply-cltn2-line} \ (?m\$1) \ ?J)$
by simp

ultimately have $\text{proj2-incident} \ (\text{apply-cltn2} \ (?a\$1) \ ?J) \ (?m\$2)$ by simp

from $\forall \ i. \ \forall \ u. \ \text{proj2-incident} \ u \ (?m\$i) \implies \neg \ (u = ?p\$i \lor u = ?q\$i)$
have $\neg \ \text{proj2-incident} \ (?p\$2) \ (?m\$2)$ by fast

with $\text{proj2-incident} \ (?p\$2) \ (?m\$2)$ have $?m\$2 \neq ?m\$2$ by auto

with $\text{proj2-incident} \ (?a\$2) \ (?p\$2)$
and $\text{proj2-incident} \ (?a\$2) \ (?m\$2)$
and $\text{proj2-incident} \ (\text{apply-cltn2} \ (?a\$1) \ ?J) \ (?p\$2)$
and $\text{proj2-incident} \ (\text{apply-cltn2} \ (?a\$1) \ ?J) \ (?m\$2)$
and $\text{proj2-incident-unique}$

have $\text{apply-cltn2} \ a1 \ ?J = a2$ by auto

with $\text{is-K2-isometry} \ ?J$ and $\text{apply-cltn2} \ p1 \ ?J = p2$

show $\exists \ J. \ \text{is-K2-isometry} \ J \land \text{apply-cltn2} \ a1 \ J = a2 \land \text{apply-cltn2} \ p1 \ J = p2$
by auto

qed

lemma K2-isometry-swap:

assumes $a \in \text{hyp2}$ and $b \in \text{hyp2}$

shows $\exists \ J. \ \text{is-K2-isometry} \ J \land \text{apply-cltn2} \ a \ J = b \land \text{apply-cltn2} \ b \ J = a$

proof

from $a \in \text{hyp2}$ and $b \in \text{hyp2}$

have $a \in K2$ and $b \in K2$ by simp-all

let $?l = \text{proj2-line-through} \ a \ b$

have $\text{proj2-incident} \ a \ ?l$ and $\text{proj2-incident} \ b \ ?l$
by (rule proj2-line-through-incident)+

from $a \in K2$ and $\text{proj2-incident} \ a \ ?l$
and line-through-K2-intersect-S-exactly-twice [of $a \ ?l$]

obtain $p$ and $q$ where $p \neq q$
and $p \in S$ and $q \in S$
and $\text{proj2-incident} \ p \ ?l$ and $\text{proj2-incident} \ q \ ?l$
and $\forall \ r \in S. \ \text{proj2-incident} \ r \ ?l \implies r = p \lor r = q$
by auto

from $a \in K2$ and $b \in K2$ and $p \in S$ and $q \in S$
and statement66-existence [of $a \ b \ p \ q$]

obtain $J$ where $\text{is-K2-isometry} \ J$ and $\text{apply-cltn2} \ a \ J = b$
and $\text{apply-cltn2} \ p \ J = q$
by auto

from $\text{apply-cltn2} \ a \ J = b$ and $\text{apply-cltn2} \ p \ J = q$
and $\text{proj2-incident} \ b \ ?l$ and $\text{proj2-incident} \ q \ ?l$

have $\text{proj2-incident} \ (\text{apply-cltn2} \ a \ J) \ ?l$
and $\text{proj2-incident} \ (\text{apply-cltn2} \ p \ J) \ ?l$
by simp-all
from ⟨a ∈ K2⟩ and ⟨p ∈ S⟩ have a ≠ p 

unfolding S-def and K2-def

by auto

with ⟨proj2-incident a ?l⟩
and ⟨proj2-incident p ?l⟩
and ⟨proj2-incident (apply-cltn2 a J) ?l⟩
and ⟨proj2-incident (apply-cltn2 p J) ?l⟩
have apply-cltn2-line ?l J = ?l by (simp add: apply-cltn2-line-unique)
with ⟨proj2-incident q ?l⟩ and apply-cltn2-preserve-incident [of q J ?l]
have proj2-incident (apply-cltn2 q J) ?l by simp

from ⟨q ∈ S⟩ and ⟨is-K2-isometry J⟩
have apply-cltn2 q J ∈ S by (unfold is-K2-isometry-def) simp
with ⟨proj2-incident q J⟩
and (∀ r ∈ S. proj2-incident r ?l ⟷ r = p ∨ r = q)
have apply-cltn2 q J = p ∨ apply-cltn2 q J = q by simp

have apply-cltn2 q J ≠ q

proof

assume apply-cltn2 q J = q

with ⟨apply-cltn2 p J = q⟩
have apply-cltn2 p J = apply-cltn2 q J by simp

hence p = q by (rule apply-cltn2-injective [of p J q])

with ⟨p ≠ q⟩ show False.. qed

with ⟨apply-cltn2 q J = p ∨ apply-cltn2 q J = q⟩
have apply-cltn2 q J = p by simp

with ⟨p ≠ q⟩
and ⟨apply-cltn2 p J = q⟩
and ⟨proj2-incident p ?l⟩
and ⟨proj2-incident q ?l⟩
and ⟨proj2-incident a ?l⟩
and statement55
have apply-cltn2 (apply-cltn2 a J) J = a by simp
with ⟨apply-cltn2 a J = b⟩ have apply-cltn2 b J = a by simp
with ⟨is-K2-isometry J⟩ and ⟨apply-cltn2 a J = b⟩
show ∃ J. is-K2-isometry J ∧ apply-cltn2 a J = b ∧ apply-cltn2 b J = a
by (simp add: exI [of - J]) qed

theorem hyp2-axiom1: ∀ a b. a b ≡ K b a

proof standard+

fix a b
let ?a' = Rep-hyp2 a
let ?b' = Rep-hyp2 b
from Rep-hyp2 and K2-isometry-swap [of ?a' ?b']
obtain J where is-K2-isometry J and apply-cltn2 ?a' J = ?b'
and apply-cltn2 ?b' J = ?a'
by auto

153
from (apply-cltn2 ?a' J = ?b' \text{ and } (apply-cltn2 ?b' J = ?a'))
have hyp2-cltn2 a J = b \text{ and } hyp2-cltn2 b J = a
  unfolding hyp2-cltn2-def by (simp-all add: Rep-hyp2-inverse)
with (is-K2-isometry J)
show a b \equiv_K b a
  by (unfold real-hyp2-C-def) (simp add: exI [of - J])
qed

theorem hyp2-axiom2: \(\forall a b p q r s. a b \equiv K p q \land a b \equiv_K r s \longrightarrow p q \equiv_K r s\)
proof standard+
fix a b p q r s
assume a b \equiv_K p q \land a b \equiv_K r s
then obtain G and H where is-K2-isometry G and is-K2-isometry H
  and hyp2-cltn2 a G = p and hyp2-cltn2 b G = q
  and hyp2-cltn2 a H = r and hyp2-cltn2 b H = s
  by (unfold real-hyp2-C-def) auto
let ?J = cltn2-compose (cltn2-inverse G) H
from (is-K2-isometry G) have is-K2-isometry (cltn2-inverse G)
  by (rule cltn2-inverse-is-K2-isometry)
with (is-K2-isometry H)
have is-K2-isometry ?J by (simp only: cltn2-compose-is-K2-isometry)

from (is-K2-isometry G) and (hyp2-cltn2 a G = p) and (hyp2-cltn2 b G = q)
  and K2-isometry.act-inv-iff
have hyp2-cltn2 p (cltn2-inverse G) = a
  and hyp2-cltn2 q (cltn2-inverse G) = b
  by simp-all
with (hyp2-cltn2 a H = r) and (hyp2-cltn2 b H = s)
  and (is-K2-isometry (cltn2-inverse G)) and (is-K2-isometry H)
  and K2-isometry.act-act [symmetric]
have hyp2-cltn2 p ?J = r and hyp2-cltn2 q ?J = s by simp-all
with (is-K2-isometry ?J)
show p q \equiv_K r s
  by (unfold real-hyp2-C-def) (simp add: exI [of - ?J])
qed

theorem hyp2-axiom3: \(\forall a b c. a b \equiv_K c c \longrightarrow a = b\)
proof standard+
fix a b c
assume a b \equiv_K c c
then obtain J where is-K2-isometry J
  and hyp2-cltn2 a J = c and hyp2-cltn2 b J = c
  by (unfold real-hyp2-C-def) auto
from (hyp2-cltn2 a J = c) and (hyp2-cltn2 b J = c)
have hyp2-cltn2 a J = hyp2-cltn2 b J by simp

from (is-K2-isometry J)
have apply-cltn2 (Rep-hyp2 a) J \in hyp2
and apply-cltn2 (Rep-hyp2 b) J ∈ hyp2
by (rule apply-cltn2-Rep-hyp2)+
with {hyp2-cltn2 a J = hyp2-cltn2 b J;
have apply-cltn2 (Rep-hyp2 a) J = apply-cltn2 (Rep-hyp2 b) J
  by (unfold hyp2-cltn2-def) (simp add: Abs-hyp2-inject)
hence Rep-hyp2 a = Rep-hyp2 b by (rule apply-cltn2-injective)
thus a = b by (simp add: Rep-hyp2-inject)
qed

interpretation hyp2: tarski-first3 real-hyp2-C
  using hyp2-axiom1 and hyp2-axiom2 and hyp2-axiom3
by unfold-locales

8.7 Some lemmas about betweenness

lemma S-at-edge:
  assumes p ∈ S and q ∈ hyp2 ∪ S and r ∈ hyp2 ∪ S and proj2-Col p q r
  shows B_R (cart2-pt p) (cart2-pt q) (cart2-pt r)
  ∨ B_R (cart2-pt p) (cart2-pt r) (cart2-pt q)
  (is B_R (?cp ?cq ?cr ∨ -)
proof −
  from ⟨p ∈ S ⟩ and ⟨q ∈ hyp2 ∪ S ⟩ and ⟨r ∈ hyp2 ∪ S ⟩
  have z-non-zero p and z-non-zero q and z-non-zero r
  by (simp-all add: hyp2-S-z-non-zero)
  with ⟨proj2-Col p q r⟩
  have real-euclid.Col ?cp ?cq ?cr by (simp add: proj2-Col-iff-euclid-cart2)
  with ⟨z-non-zero p⟩ and ⟨z-non-zero q⟩ and ⟨z-non-zero r⟩
  have proj2-pt ?cp = p and proj2-pt ?cq = q and proj2-pt ?cr = r
  by (simp-all add: proj2-cart2)
  from ⟨proj2-pt ?cp = p ⟩ and ⟨p ∈ S ⟩
  have norm ?cp = 1 by (simp add: norm-eq-1-iff-in-S)
  from ⟨proj2-pt ?cq = q ⟩ and ⟨proj2-pt ?cr = r ⟩
  and ⟨q ∈ hyp2 ∪ S ⟩ and ⟨r ∈ hyp2 ∪ S ⟩
  have norm ?cq ≤ 1 and norm ?cr ≤ 1
  by (simp-all add: norm-le-1-iff-in-hyp2-S)
proof cases
  assume B_R ?cr ?cp ?cq
  then obtain k where k ≥ 0 and k ≤ 1
  and ?cp − ?cr = k ∗_R (?cq − ?cr)
  by (unfold real-euclid-B-def) auto
  from ⟨?cp − ?cr = k ∗_R (?cq − ?cr)⟩
  have ?cp = k ∗_R ?cq + (1 − k) ∗_R ?cr by (simp add: algebra-simps)
  with ⟨norm ?cp = 1 ⟩ have norm (k ∗_R ?cq + (1 − k) ∗_R ?cr) = 1 by simp
  with norm-triangle-ineq [of k ∗_R ?cq (1 − k) ∗_R ?cr]
  have norm (k ∗_R ?cq) + norm ((1 − k) ∗_R ?cr) ≥ 1 by simp

155
from \( k \geq 0 \) and \( k \leq 1 \)

have \( \text{norm} (k \ast_R \text{cq}) + \text{norm} ((1 - k) \ast_R \text{cr}) = k \ast \text{norm} \text{cq} + (1 - k) \ast \text{norm} \text{cr} \)
by simp

with \( \text{norm} (k \ast_R \text{cq}) + \text{norm} ((1 - k) \ast_R \text{cr}) \geq 1 \)

have \( k \ast \text{norm} \text{cq} + (1 - k) \ast \text{norm} \text{cr} \geq 1 \) by simp

from \( \text{norm} \text{cq} \leq 1 \) and \( k \geq 0 \) and \mult mono \( \text{of} k \ast \text{norm} \text{cq} \)

have \( k \ast \text{norm} \text{cq} \leq k \) by simp

from \( \text{norm} \text{cr} \leq 1 \) and \( k \leq 1 \)

and \mult mono \( \text{of} 1 - k \ast 1 - k \ast \text{norm} \text{cr} \)

have \( (1 - k) \ast \text{norm} \text{cr} \leq 1 - k \) by simp

with \( k \ast \text{norm} \text{cr} \leq k \)

have \( k \ast \text{norm} \text{cq} + (1 - k) \ast \text{norm} \text{cr} \leq 1 \) by simp

with \( k \ast \text{norm} \text{cq} + (1 - k) \ast \text{norm} \text{cr} \geq 1 \)

have \( k \ast \text{norm} \text{cq} + (1 - k) \ast \text{norm} \text{cr} = 1 \) by simp

with \( k \ast \text{norm} \text{cq} \leq k \) have \( (1 - k) \ast \text{norm} \text{cr} \leq 1 - k \) have \( (1 - k) \ast \text{norm} \text{cr} = 1 - k \) by simp

with \( k \ast \text{norm} \text{cq} + (1 - k) \ast \text{norm} \text{cr} = 1 \) have \( k \ast \text{norm} \text{cq} = k \) by simp

have \( \text{cp} = \text{cq} \lor \text{cq} = \text{cr} \lor \text{cr} = \text{cp} \)

proof cases

assume \( k = 0 \lor k = 1 \)

with \( \text{cp} = k \ast \text{cq} + (1 - k) \ast \text{cr} \)

show \( \text{cp} = \text{cq} \lor \text{cq} = \text{cr} \lor \text{cr} = \text{cp} \) by auto

next

assume \( \neg (k = 0 \lor k = 1) \)

hence \( k \neq 0 \) and \( k \neq 1 \) by simp-all

with \( k \ast \text{norm} \text{cq} = k \) and \( (1 - k) \ast \text{norm} \text{cr} = 1 - k \)

have \( \text{norm} \text{cq} = 1 \) and \( \text{norm} \text{cr} = 1 \) by simp-all

with \( \text{proj2-pt} \text{cq} = q \) and \( \text{proj2-pt} \text{cr} = r \)

have \( q \in S \) and \( r \in S \) by \( \text{simp-all add:} \text{norm-eq-1-iff-in-S} \)

with \( \langle p, q, r \rangle \subseteq S \) have \( \{ p, q, r \} \subseteq S \) by simp

from \( \langle \text{proj2-Col} p q r \rangle \)

have \( \text{proj2-set-Col} \{ p, q, r \} \) by \( \text{simp add:} \text{proj2-Col-iff-set-Col} \)

with \( \{ p, q, r \} \subseteq S \) have \( \text{card} \{ p, q, r \} \leq 2 \) by \( \text{rule card-line-intersect-S} \)

have \( p = q \lor q = r \lor r = p \)

proof \( \text{rule ccontr} \)

assume \( \neg (p = q \lor q = r \lor r = p) \)

hence \( p \neq q \) and \( q \neq r \) and \( r \neq p \) by simp-all

from \( q \neq r \) have \( \text{card} \{ q, r \} = 2 \) by simp

with \( p \neq q \) and \( r \neq p \) have \( \text{card} \{ p, q, r \} = 3 \) by simp

with \( \text{card} \{ p, q, r \} \leq 2 \) show \( \text{False} \) by simp
qed

thus \( \exists p \exists q \exists r \exists \text{cp} \) \( \exists \text{cq} \) \( \exists \text{cr} \) \( \text{cp} \lor \text{cq} \lor \text{cr} \) by auto

qed

thus \( B \exists p \exists q \exists r \exists \text{cp} \) \( \exists \text{cq} \) \( \exists \text{cr} \) \( \text{cp} \lor \text{cq} \lor \text{cr} \) \( \text{by} \) \( \text{auto simp add: real-euclid.th3-1 real-euclid.th3-2} \)

next

assume \( \neg B \exists p \exists q \exists r \exists \text{cp} \) \( \exists \text{cq} \) \( \exists \text{cr} \)

with \( \langle \text{real-euclid.Col } \exists \text{cp} \rangle \)

show \( B \exists p \exists q \exists r \exists \text{cp} \) \( \exists \text{cq} \) \( \exists \text{cr} \) \( \text{by} \) \( \langle \text{auto simp add: real-euclid.th3-1 real-euclid.th3-2} \rangle \)

qed

lemma hyp2-in-middle:

assumes \( p \in S \) \( q \in S \) \( r \in \text{hyp2} \cup S \) \( \text{proj2-Col } p \exists q \exists r \)

and \( p \neq q \)

shows \( B \exists p \exists q \exists r \exists \text{cp} \) \( \exists \text{cq} \) \( \exists \text{cr} \) \( \text{by} \) \( \langle \text{simp add: S-at-edge} \rangle \)

proof \( \langle \text{rule ccontr} \rangle \)

assume \( \neg B \exists p \exists q \exists r \exists \text{cp} \)

hence \( \neg B \exists p \exists q \exists r \exists \text{cq} \)

by \( \langle \text{auto simp add: real-euclid.th3-2 [of } \exists \text{cq} \rangle \rangle \)

from \( \langle p \in S \rangle \) \( \langle q \in S \rangle \) \( \langle r \in \text{hyp2} \cup S \rangle \) \( \langle \text{proj2-Col } p \exists q \exists r \rangle \)

have \( B \exists p \exists q \exists r \exists \text{cp} \) \( \exists \text{cq} \) \( \exists \text{cr} \) \( \text{by} \) \( \langle \text{simp add: S-at-edge} \rangle \)

with \( \langle \neg B \exists p \exists q \exists r \exists \text{cp} \rangle \) have \( B \exists p \exists q \exists r \exists \text{cp} \) \( \exists \text{cq} \) \( \exists \text{cr} \) \( \text{by} \) \( \langle \text{simp} \rangle \)

from \( \langle p \in S \rangle \) \( \langle q \in S \rangle \)

have \( \text{z-non-zero } p \) \( \text{and} \) \( \text{z-non-zero } q \) \( \text{by} \) \( \langle \text{simp-all add: hyp2-S-z-non-zero} \rangle \)

hence \( \text{proj2-pt } \exists \text{cp} \) \( \exists \text{cq} \) \( \exists \text{cr} \) \( \text{by} \) \( \langle \text{rule real-euclid.th3-4} \rangle \)

hence \( \text{proj2-pt } \exists \text{cp} = \text{proj2-pt } \exists \text{cq} \)

with \( \langle p \neq q \rangle \) show \( \text{False} \).

qed

lemma hyp2-incident-in-middle:

assumes \( p \neq q \) \( \langle p \in S \rangle \) \( \langle q \in S \rangle \) \( \langle r \in \text{hyp2} \cup S \rangle \) \( \langle \text{proj2-incident } p \exists q \exists r \rangle \)

and \( \text{proj2-incident } p \exists r \) \( \text{proj2-incident } q \exists r \) \( \text{proj2-incident } a \exists l \)

shows \( B \exists p \exists q \exists r \exists \text{cp} \) \( \exists \text{cq} \) \( \exists \text{cr} \) \( \text{by} \) \( \langle \text{simp add: proj2-cart2} \rangle \)

proof \( \langle \text{rule proj2-incident-Col} \rangle \)

from \( \langle \text{proj2-incident } p \exists r \rangle \) \( \langle \text{proj2-incident } q \exists r \rangle \) \( \langle \text{proj2-incident } a \exists l \rangle \)

have \( \text{proj2-Col } p \exists q \exists a \) \( \langle \text{rule proj2-incident-Col} \rangle \)

from \( \langle p \in S \rangle \) \( \langle q \in S \rangle \) \( \langle a \in \text{hyp2} \cup S \rangle \) \( \langle a \text{ and } p \neq q \rangle \)
lemma extend-to-S:
assumes \( p \in \text{hyp2} \cup S \) and \( q \in \text{hyp2} \cup S \)
shows \( \exists r \in S. \ B_R \ (\text{cart2-pt} \ p) \ (\text{cart2-pt} \ q) \ (\text{cart2-pt} \ r) \)
(prove cases)
assume \( q \in S \)

have \( B_R \ ?cp \ ?cq \ ?cq \) by (rule real-euclid.th3-1)
with \( (q \in S) \) show \( \exists r \in S. \ B_R \ ?cp \ ?cq \ (\text{cart2-pt} \ r) \) by auto
next
assume \( q \not \in S \)
with \( (q \in \text{hyp2} \cup S) \) have \( q \in K2 \) by simp
let \( ?l = \text{proj2-line-through} \ p \ q \)
have \( \text{proj2-incident} \ p \ ?l \) and \( \text{proj2-incident} \ q \ ?l \)
by (rule proj2-line-through-incident)+
from \( (q \in K2) \) and \( (\text{proj2-incident} \ q \ ?l) \)
and \( \text{line-through-K2-intersect-S-twice} \ [\text{of q ?l}] \)
obtain \( s \) and \( t \) where \( s \neq t \) and \( s \in S \) and \( t \in S \)
and \( \text{proj2-incident} \ s \ ?l \) and \( \text{proj2-incident} \ t \ ?l \)
by auto
let \( ?cs = \text{cart2-pt} \ s \)
let \( ?ct = \text{cart2-pt} \ t \)
from \( \text{proj2-incident} \ s \ ?l \)
and \( \text{proj2-incident} \ t \ ?l \)
and \( \text{proj2-incident} \ p \ ?l \)
and \( \text{proj2-incident} \ q \ ?l \)
have \( \text{proj2-Col} \ s \ p \ q \) and \( \text{proj2-Col} \ t \ p \ q \) and \( \text{proj2-Col} \ s \ t \ q \)
by (simp-all add: proj2-incident-Col)
from \( \text{proj2-Col} \ s \ p \ q \) and \( \text{proj2-Col} \ t \ p \ q \)
and \( (s \in S) \) and \( (t \in S) \) and \( (p \in \text{hyp2} \cup S) \) and \( (q \in \text{hyp2} \cup S) \)
have \( B_R \ ?cs \ ?cp \ ?cq \ ?cp \ ?cs \ ?cq \ ?cp \ ?ct \ ?eq \ ?ct \)
by (simp-all add: S-at-edge)
with real-euclid.th3-2
have \( B_R \ ?cq \ ?cp \ ?cs \ ?V \ B_R \ ?cp \ ?cq \ ?cs \ ?and \ B_R \ ?eq \ ?cp \ ?ct \ ?V \ B_R \ ?cp \ ?eq \ ?ct \)
by fast+
from \( (s \in S) \) and \( (t \in S) \) and \( (q \in \text{hyp2} \cup S) \) and \( (\text{proj2-Col} \ s \ t \ q) \) and \( (s \neq t) \)
have \( B_R \ ?cs \ ?cq \ ?ct \) by (rule hyp2-in-middle)
hence \( B_R \ ?ct \ ?cq \ ?cs \ ) by (rule real-euclid.th3-2)

have \( B_R \ ?cp \ ?cq \ ?cs \ ?V \ B_R \ ?cp \ ?eq \ ?ct \)
(prove (rule ccontr))
assume \( \neg (B_R \ ?cp \ ?cq \ ?cs \ ?V \ B_R \ ?cp \ ?eq \ ?ct) \)
hence \( \neg B_I R \ ?ep \ ?cq \ ?cs \) and \( \neg B_I R \ ?cp \ ?cq \ ?ct \) by simp-all

with \( B_I R \ ?cq \ ?cp \ ?cs \lor B_I R \ ?cp \ ?cq \ ?cs \)

and \( B_I R \ ?cq \ ?cp \ ?ct \lor B_I R \ ?cp \ ?cq \ ?ct \)

have \( B_I R \ ?cq \ ?cp \ ?cs \) and \( B_I R \ ?cq \ ?cp \ ?ct \)

by simp-all

from \( \neg B_I R \ ?cp \ ?cq \ ?cs \) and \( B_I R \ ?cq \ ?cp \ ?cs \) have \( ?cp \neq ?cq \) by auto

with \( B_I R \ ?cq \ ?cp \ ?cs \lor B_I R \ ?cq \ ?cp \ ?ct \)

have \( B_I R \ ?cq \ ?cs \ ?ct \lor B_I R \ ?cq \ ?ct \ ?cs \)

by \((\text{simp add: real-euclid-th5-1} \ [\text{of } ?cq \ ?cp \ ?cs \ ?ct])\)

with \( B_I R \ ?cs \ ?cq \ ?ct \) and \( B_I R \ ?ct \ ?cq \ ?cs \)

have \( ?cq = ?cs \lor ?cq = ?ct \) by \((\text{auto simp add: real-euclid.\text{th3-4}})\)

with \( q \in \text{hyp2} \cup S \) and \( s \in S \) and \( t \in S \)

have \( q = s \lor q = t \) by \((\text{auto simp add: hyp2-S-cart2-inj})\)

with \( \{s \in S\} \) and \( \{t \in S\} \) have \( q \in S \) by auto

with \( q \notin S \) show False ..

qed

with \( \{s \in S\} \) and \( \{t \in S\} \) show \( \exists r \in S. B_I R \ ?cp \ ?cq \ (\text{cart2-pt } r) \) by auto

qed

definition endpoint-in-S :: \( \text{proj2} \Rightarrow \text{proj2} \Rightarrow \text{proj2} \) where

endpoint-in-S \( a \ b \triangleq \\epsilon \ p \. p \in S \land B_I R \ (\text{cart2-pt } a) \ (\text{cart2-pt } b) \ (\text{cart2-pt } p) \)

lemma endpoint-in-S: assumes \( a \in \text{hyp2} \cup S \) and \( b \in \text{hyp2} \cup S \)

shows \( \text{endpoint-in-S} \ a \ b \) (is \( ?p \in S \))

and \( B_I R \ (\text{cart2-pt } a) \ (\text{cart2-pt } b) \ (\text{cart2-pt } (\text{endpoint-in-S} \ a \ b)) \)

(is \( B_I R \ ?ca \ ?cb \ ?cp \))

proof –

from \( a \in \text{hyp2} \cup S \) and \( b \in \text{hyp2} \cup S \) and \( \text{extend-to-S} \)

have \( \exists p. \ p \in S \land B_I R \ ?ca \ ?cb \ ?cp \) by auto

hence \( ?p \in S \land B_I R \ ?ca \ ?cb \ ?cp \)

by \((\text{unfold endpoint-in-S-def}) \ (\text{rule someI-ex})\)

thus \( ?p \in S \) and \( B_I R \ ?ca \ ?cb \ ?cp \) by simp-all

qed

lemma endpoint-in-S-swap:

assumes \( a \neq b \) and \( a \in \text{hyp2} \cup S \) and \( b \in \text{hyp2} \cup S \)

shows \( \text{endpoint-in-S} \ a \ b \neq \text{endpoint-in-S} \ b \ a \) (is \( ?p \neq ?q \))

proof

let \( ?ca = \text{cart2-pt} \ a \)

let \( ?cb = \text{cart2-pt} \ b \)

let \( ?cp = \text{cart2-pt} \ ?p \)

let \( ?cq = \text{cart2-pt} \ ?q \)

from \( a \neq b \) and \( a \in \text{hyp2} \cup S \) and \( b \in \text{hyp2} \cup S \)

have \( B_I R \ ?ca \ ?cb \ ?cp \ ?cq \)

by \((\text{simp-all add: endpoint-in-S})\)

assume \( ?p = ?q \)

with \( B_I R \ ?cb \ ?ca \ ?cq \) have \( B_I R \ ?cb \ ?ca \ ?cp \) by simp
lemma endpoints-in-S-incident:
  assumes a ≠ b and a ∈ hyp2 ∪ S and b ∈ hyp2 ∪ S
  and proj2-incident a l and proj2-incident b l
  shows proj2-incident (endpoint-in-S a b) l (is proj2-incident ?p l)
proof –
from (a ∈ hyp2 ∪ S; and (b ∈ hyp2 ∪ S)
have ?p ∈ S and B_R (cart2-pt a) (cart2-pt b) (cart2-pt ?p)
  (is B_R ?ca ?cb ?cp)
  by (rule endpoint-in-S)+
  from (a ∈ hyp2 ∪ S; and (b ∈ hyp2 ∪ S) and (?p ∈ S)
  have z-non-zero a and z-non-zero b and z-non-zero ?p
  by (simp-all add: hyp2-S-z-non-zero)
  from (B_R (?ca ?cb ?cp)
  have real-euclid.Col (?ca ?cb ?cp unfolding real-euclid.Col-def ..
  with (?z-non-zero a and (?z-non-zero b and (?z-non-zero ?p) and (a ≠ b)
  and (proj2-incident a l and (proj2-incident b l)
  show proj2-incident ?p l by (rule euclid-Col-cart2-incident)
  qed

lemma endpoints-in-S-incident-unique:
  assumes a ≠ b and a ∈ hyp2 ∪ S and b ∈ hyp2 ∪ S and p ∈ S
  and proj2-incident a l and proj2-incident b l and proj2-incident p l
  shows p = endpoint-in-S a b ∨ p = endpoint-in-S b a
  (is p = (?q ∨ p = ?r)
proof –
from (a ≠ b) and (a ∈ hyp2 ∪ S; and (b ∈ hyp2 ∪ S)
have ?q ≠ ?r by (rule endpoint-in-S-swap)
  from (a ∈ hyp2 ∪ S; and (b ∈ hyp2 ∪ S)
  have ?q ∈ S and ?r ∈ S by (simp-all add: endpoint-in-S)
  from (a ≠ b) and (a ∈ hyp2 ∪ S; and (b ∈ hyp2 ∪ S)
  and (proj2-incident a l and (proj2-incident b l)
  have proj2-incident ?q l and proj2-incident ?r l
    by (simp-all add: endpoint-in-S-incident)
    with (?q ≠ ?r) and (?q ∈ S; and (?r ∈ S) and (p ∈ S) and (proj2-incident p l
    show p = ?q ∨ p = ?r by (simp add: line-S-two-intersections-only)
  qed

lemma endpoint-in-S-unique:
  assumes a ≠ b and a ∈ hyp2 ∪ S and b ∈ hyp2 ∪ S and p ∈ S
  and B_R (cart2-pt a) (cart2-pt b) (cart2-pt p) (is B_R ?ca ?cb ?cp)
shows \( p = \text{endpoint-in-S} \ a \ b \) (is \( p = \ ?q \))

proof (rule contr)
from \( \langle a \in \text{hyp2} \cup S, \quad \text{and} \quad \langle b \in \text{hyp2} \cup S, \quad \text{and} \quad \langle p \in S \rangle \)
have \( z\text{-non-zero} \ a \quad \text{and} \quad z\text{-non-zero} \ b \quad \text{and} \quad z\text{-non-zero} \ p \)
by (simp-all add: hyp2-S-z-non-zero)
with \( \langle B \quad \text{?ca} \quad \text{?cb} \quad \text{?cp} \quad \text{and} \quad \text{euclid-B-cart2-common-line} \ [\text{of} \quad a \ b \ p \] \)
obtain \( \langle \text{proj2-incident} \ a \ l, \quad \text{proj2-incident} \ b \ l, \quad \text{proj2-incident} \ p \ l \quad \text{by} \quad \text{auto} \)
with \( \langle a \neq b, \quad \text{and} \quad \langle a \in \text{hyp2} \cup S, \quad \text{and} \quad \langle b \in \text{hyp2} \cup S, \quad \text{and} \quad \langle p \in S \rangle \)
have \( p = \ ?q \lor p = \text{endpoint-in-S} \ b \ a \) (is \( p = \ ?q \lor p = \ ?r \))
by (rule endpoints-in-S-incident-unique)

assume \( p \neq \ ?q \)
with \( \langle \quad \text{?q} \lor p = \ ?r \rangle \quad \text{have} \quad p = \ ?r \quad \text{by} \quad \text{simp} \)
with \( \langle \quad \text{?q} \lor p = \ ?r \rangle \quad \text{have} \quad \text{?ca} = \ ?cb \quad \text{by} \quad \text{(simp add: endpoint-in-S)} \)
with \( \langle \quad \text{?ca} = \ ?cb \quad \text{by} \quad \text{(rule real-euclid.th3-4)} \)
with \( \langle \quad \text{?ca} = \ ?cb \quad \text{by} \quad \text{(rule hyp2-S-cart2-inj)} \)
with \( \langle a \neq b \rangle \quad \text{show} \quad \text{False} \quad .. \)

qed

lemma between-hyp2-S:
assumes \( p \in \text{hyp2} \cup S \quad \text{and} \quad r \in \text{hyp2} \cup S \quad \text{and} \quad k \geq 0 \quad \text{and} \quad k \leq 1 \)
shows \( \text{proj2-pt} \ (k * R \ (\text{cart2-pt} \ r) + (1 - k) * R \ (\text{cart2-pt} \ p)) \in \text{hyp2} \cup S \)
(is \( \text{proj2-pt} \ ?cq \in -) \)

proof –
let \( ?cp = \text{cart2-pt} \ p \)
let \( ?cr = \text{cart2-pt} \ r \)
let \( ?q = \text{proj2-pt} \ ?cq \)
from \( \langle p \in \text{hyp2} \cup S, \quad \text{and} \quad \langle r \in \text{hyp2} \cup S \rangle \)
have \( z\text{-non-zero} \ p \quad \text{and} \quad z\text{-non-zero} \ r \quad \text{by} \quad \text{(simp-all add: hyp2-S-z-non-zero)} \)
hence \( \text{proj2-pt} \ ?cp = p \quad \text{and} \quad \text{proj2-pt} \ ?cr = r \quad \text{by} \quad \text{(simp-all add: proj2-cart2)} \)
with \( \langle \quad \text{?cq} \leq 1 \quad \text{and} \quad \text{?cr} \leq 1 \quad \text{by} \quad \text{(simp-all add: norm-le-1-iff-in-hyp2-S)} \)
from \( \langle k \geq 0 \rangle \quad \text{and} \quad \langle k \leq 1 \rangle \quad \text{and} \quad \text{norm-triangle-ineq} \ [\text{of} \quad k * R \ ?cr \ (1 - k) * R \ ?cp] \)
have \( \text{norm} \ ?cq \leq k * \text{norm} \ ?cr \ + \ (1 - k) * \text{norm} \ ?cp \quad \text{by} \quad \text{simp} \)
from \( \langle k \geq 0 \rangle \quad \text{and} \quad \langle \text{norm} \ ?cq \leq 1 \rangle \quad \text{and} \quad \text{mult-mono} \ [\text{of} \quad k \ k \ \text{norm} \ ?cr \ 1] \)
have \( k * \text{norm} \ ?cr \leq 1 \quad \text{by} \quad \text{simp} \)
from \( \langle k \leq 1 \rangle \quad \text{and} \quad \langle \text{norm} \ ?cp \leq 1 \rangle \quad \text{and} \quad \text{mult-mono} \ [\text{of} \quad 1 - k \ 1 - k \ \text{norm} \ ?cp \ 1] \)
have \( (1 - k) * \text{norm} \ ?cp \leq 1 - k \quad \text{by} \quad \text{simp} \)
with \( \langle \text{norm} \ ?cq \leq k * \text{norm} \ ?cr \ + \ (1 - k) * \text{norm} \ ?cp \rangle \quad \text{and} \quad \langle k * \text{norm} \ ?cr \leq k \rangle \)
have \( \|q\| \leq 1 \) by simp
thus \( q \in \text{hyp2} \cup S \) by (simp add: norm-le-1-iff-hyp2-S)
qed

8.8 The Klein–Beltrami model satisfies axiom 4

definition expansion-factor :: \( \text{proj2} \Rightarrow \text{cltn2} \Rightarrow \text{real} \)
where

\[
\text{expansion-factor} \ p \ J \triangleq (\text{cart2-append1} \ p \ v \ \text{cltn2-rep} \ J)
\]

lemma expansion-factor:
assumes \( p \in \text{hyp2} \cup S \) and is-K2-isometry \( J \)
shows expansion-factor \( p \ J \neq 0 \)
and cart2-append1 \( p \ v \ \text{cltn2-rep} \ J \)
= expansion-factor \( p \ J \ *_R \text{cart2-append1} \ (\text{apply-cltn2} \ p \ J) \)
proof -
from \( p \in \text{hyp2} \cup S \) and is-K2-isometry \( J \)
have z-non-zero \( (\text{apply-cltn2} \ p \ J) \)
by (rule is-K2-isometry-z-non-zero)
from \( p \in \text{hyp2} \cup S \) and is-K2-isometry \( J \)
and cart2-append1-apply-cltn2
obtain \( k \) where \( k \neq 0 \)
and cart2-append1 \( p \ v \ \text{cltn2-rep} \ J = k *_R \text{cart2-append1} \ (\text{apply-cltn2} \ p \ J) \)
by auto
from cart2-append1 \( p \ v \ \text{cltn2-rep} \ J = k *_R \text{cart2-append1} \ (\text{apply-cltn2} \ p \ J) \)
and z-non-zero \( (\text{apply-cltn2} \ p \ J) \)
have expansion-factor \( p \ J = k \)
by (unfold expansion-factor-def) (simp add: cart2-append1-z)
with \( k \neq 0 \)
and cart2-append1 \( p \ v \ \text{cltn2-rep} \ J = k *_R \text{cart2-append1} \ (\text{apply-cltn2} \ p \ J) \)
show expansion-factor \( p \ J \neq 0 \)
and cart2-append1 \( p \ v \ \text{cltn2-rep} \ J \)
= expansion-factor \( p \ J \ *_R \text{cart2-append1} \ (\text{apply-cltn2} \ p \ J) \)
by simp-all
qed

lemma expansion-factor-linear-apply-cltn2:
assumes \( p \in \text{hyp2} \cup S \) and \( q \in \text{hyp2} \cup S \) and \( r \in \text{hyp2} \cup S \)
and is-K2-isometry \( J \)
and cart2-pt \( r = k *_R \text{cart2-pt} \ p + (1 - k) *_R \text{cart2-pt} \ q \)
shows expansion-factor \( r \ J *_R \text{cart2-append1} \ (\text{apply-cltn2} \ r \ J) \)
= \( k * \text{expansion-factor} \ p \ J \ *_R \text{cart2-append1} \ (\text{apply-cltn2} \ p \ J) \)
+ \((1 - k) * \text{expansion-factor} \ q \ J \ *_R \text{cart2-append1} \ (\text{apply-cltn2} \ q \ J) \)
(is \( ?er \ * _R - = (k * \ ?ep) * _R - + ((1 - k) * \ ?eq) * _R - \) )
proof -
let \( ?cp = \text{cart2-pt} \ p \)
let \( ?cq = \text{cart2-pt} \ q \)
let \( ?cr = \text{cart2-pt} \ r \)
let \( ?ep1 = \text{cart2-append1} \ p \)
let \( ?eq1 = \text{cart2-append1} \ q \)
qed
let \(?cr1 = \text{cart2-append1} \, r\) 
let \(?repJ = \text{cltn2-rep} \, J\) 
from \(?p \in \text{hyp2} \cup \mathcal{S}\) and \(?q \in \text{hyp2} \cup \mathcal{S}\) and \(?r \in \text{hyp2} \cup \mathcal{S}\) 
have \(\text{z-non-zero} \, p\) and \(\text{z-non-zero} \, q\) and \(\text{z-non-zero} \, r\) 
  by (simp-all add: \text{hyp2} \cup \mathcal{S} \, \text{z-non-zero})

from \(?cr = k * \text{cart2-append1} \, cp + (1 - k) * R \, ?eq\) 
have \(\text{vector2-append1} \, ?cr\) 
  \(= k * R \, \text{vector2-append1} \, ?cp + (1 - k) * R \, \text{vector2-append1} \, ?eq\) 
  by (unfold \text{vector2-append1-def} \, \text{vector-def}) (simp add: vec-eq-iff)

with \((\text{z-non-zero} \, p)\) and \((\text{z-non-zero} \, q)\) and \((\text{z-non-zero} \, r)\) 
have \(?cr1 = k * \text{cart2-append1} \, \text{cart2-pt} \, r \, J\) by (simp add: \text{cart2-append1})

hence \(?cr1 * ?repJ = k * R \, \{\text{cart2-pt} \, q \, J\} + (1 - k) * R \, \{\text{cart2-append1} \, Q \, J\}\) 
  by (simp add: \text{expansion-factor})

qed

lemma \text{expansion-factor-linear}: 
  assumes \(p \in \text{hyp2} \cup \mathcal{S}\) and \(q \in \text{hyp2} \cup \mathcal{S}\) and \(r \in \text{hyp2} \cup \mathcal{S}\) 
  and \((\text{is-K2-isometry} \, J)\) 
  and \(\text{cart2-pt} \, r = k * R \, \text{cart2-pt} \, p + (1 - k) * R \, \text{cart2-pt} \, q\) 
  shows \((\text{expansion-factor} \, J)\) 
  \(= k * \text{expansion-factor} \, J + (1 - k) * \text{expansion-factor} \, q \, J\) 
  \((\text{is} \, ?cr = k * ?eq + (1 - k) * ?eq)\)

proof –

from \(?p \in \text{hyp2} \cup \mathcal{S}\) and \(?q \in \text{hyp2} \cup \mathcal{S}\) and \(?r \in \text{hyp2} \cup \mathcal{S}\) 
  and \((\text{is-K2-isometry} \, J)\) 
  have \(\text{z-non-zero} \, \text{apply-cltn2} \, p \, J\) 
  and \(\text{z-non-zero} \, \text{apply-cltn2} \, q \, J\) 
  and \(\text{z-non-zero} \, \text{apply-cltn2} \, r \, J\) 
  by (simp-all add: \text{is-K2-isometry-z-non-zero})

from \(?p \in \text{hyp2} \cup \mathcal{S}\) and \(?q \in \text{hyp2} \cup \mathcal{S}\) and \(?r \in \text{hyp2} \cup \mathcal{S}\) 
  and \((\text{is-K2-isometry} \, J)\) 
  and \(\text{cart2-pt} \, r = k * R \, \text{cart2-pt} \, p + (1 - k) * R \, \text{cart2-pt} \, q\) 
  have \(?cr * R \, \text{cart2-append1} \, \text{apply-cltn2} \, r \, J\) 
  \(= (k * ?eq) * R \, \text{cart2-append1} \, \text{apply-cltn2} \, p \, J\) 
  + ((1 - k) * ?eq) * R \, \text{cart2-append1} \, \text{apply-cltn2} \, q \, J\) 
  by (rule \text{expansion-factor-linear-apply-cltn2})

hence \((?cr * R \, \text{cart2-append1} \, \text{apply-cltn2} \, r \, J)\) \((\text{simp add: vec-eq-iff})\) 
  \(= ((k * ?eq) * R \, \text{cart2-append1} \, \text{apply-cltn2} \, p \, J)\) 
  + ((1 - k) * ?eq) * R \, \text{cart2-append1} \, \text{apply-cltn2} \, q \, J\) 
  by \text{simp}

with \((\text{z-non-zero} \, \text{apply-cltn2} \, p \, J)\)
and (\textit{z-non-zero} (apply-cltn2 q J))
and (\textit{z-non-zero} (apply-cltn2 r J))
show \(\omega r = k * \omega p + (1 - k) * \omega q\) by (simp add: cart2-append1-z)
qed

\textbf{lemma} \textit{expansion-factor-\textit{sgn}-invariant}:
\textbf{assumes} \(p \in \text{hyp2} \cup S\) \textbf{and} \(q \in \text{hyp2} \cup S\) \textbf{and} \(\text{is-K2-isometry} J\)
\textbf{shows} \(\text{sgn} (\text{expansion-factor} p J) = \text{sgn} (\text{expansion-factor} q J)\)
(is \(\text{sgn} \ \omega p = \text{sgn} \ \omega q\))
\textbf{proof} (rule ccontr)
\textbf{assume} \(\text{sgn} \ \omega p \neq \text{sgn} \ \omega q\)
\begin{itemize}
  \item from \((p \in \text{hyp2} \cup S)\) \textbf{and} \((q \in \text{hyp2} \cup S)\) \textbf{and} \(\text{(is-K2-isometry} J)\)
  \item have \(\omega p \neq 0\) \textbf{and} \(\omega q \neq 0\) by (simp-all add: expansion-factor)
  \item hence \(\text{sgn} \ \omega p \in \{-1,1\}\) \textbf{and} \(\text{sgn} \ \omega q \in \{-1,1\}\)
  \item by \(\text{(simp-all add: sgn-real-def)}\)
  \item with \((\text{sgn} \ \omega p \neq \text{sgn} \ \omega q)\) have \(\text{sgn} \ \omega p = - \text{sgn} \ \omega q\) by auto
  \item hence \(\text{sgn} \ \omega p = \text{sgn} (\ - \text{sgn} \ \omega q)\) by (subst sgn-minus)
  \item with \(\text{sgn-plus} [\text{of} \ \omega p - \omega q]\)
  \item have \(\text{sgn} (\ \omega p - \omega q) = \text{sgn} \ \omega p\) by (simp add: algebra-simps)
  \item with \((\text{sgn} \ \omega p \in \{-1,1\})\) have \(\omega p - \omega q \neq 0\) by (auto simp add: sgn-real-def)
\end{itemize}

let \(\cdots = \cdots / (\omega p - \omega q)\)
\begin{itemize}
  \item from \((\text{sgn} \ (\omega p - \omega q) = \text{sgn} \ \omega p)\) \textbf{and} \((\text{sgn} \ \omega p = \text{sgn} (\ - \omega q))\)
  \item have \(\text{sgn} (\omega p - \omega q) = \text{sgn} (\ - \omega q)\) by simp
  \item with \((\omega p - \omega q \neq 0)\) and \(\text{sgn-div} [\text{of} \omega p - \omega q - \omega q]\)
  \item have \(\omega k > 0\) by simp
\end{itemize}

from \((\omega p - \omega q \neq 0)\)
\begin{itemize}
  \item have \(1 - \omega k = \omega p / (\omega p - \omega q)\) by (simp add: field-simps)
  \item with \((\text{sgn} \ (\omega p - \omega q) = \text{sgn} \ \omega p)\) \textbf{and} \((\omega p - \omega q \neq 0)\)
  \item have \(1 - \omega k > 0\) by (simp add: sgn-div)
  \item hence \(\omega k < 1\) by simp
\end{itemize}

let \(\cdots = \cdots\)
\begin{itemize}
  \item from \((p \in \text{hyp2} \cup S)\) \textbf{and} \((q \in \text{hyp2} \cup S)\) \textbf{and} \(\omega k > 0\) \textbf{and} \(\omega k < 1\)
  \item and \(\text{between-hyp2-S} [\text{of} \ q \ \omega k]\)
  \item have \(\omega r \in \text{hyp2} \cup S\) by simp
  \item with \((p \in \text{hyp2} \cup S)\) \textbf{and} \((q \in \text{hyp2} \cup S)\) \textbf{and} \(\text{(is-K2-isometry} J)\)
  \item and \(\text{cart2-pt} \ \omega r = \omega cr\)
  \item and \(\text{expansion-factor-linear} [\text{of} \ p \ q \ \omega r \ \omega k]\)
  \item have \(\omega er = \omega k * \omega ep + (1 - \omega k) * \omega eq\) by simp
  \item with \((\omega p - \omega q \neq 0)\) have \(\omega er = 0\) by (simp add: field-simps)
\end{itemize}
with \( \forall r \in \text{hyp2} \cup S \) and \((\text{is-K2-isometry} J)\)

show False by (simp add: expansion-factor)

qed

lemma statement-63:
assumes \( p \in \text{hyp2} \cup S \) and \( q \in \text{hyp2} \cup S \) and \( r \in \text{hyp2} \cup S \)
and \((\text{is-K2-isometry} J) \) and \( B_{R} \) (\text{cart2-pt} \( p \)) (\text{cart2-pt} \( q \)) (\text{cart2-pt} \( r \))

shows \( B_{R} \)
(\text{cart2-pt} \( \text{apply-cltn2} \( p \) \))
(\text{cart2-pt} \( \text{apply-cltn2} \( q \) \))
(\text{cart2-pt} \( \text{apply-cltn2} \( r \) \))

proof –
let \( \equiv_{p} = \text{cart2-pt} \( p \) \)
let \( \equiv_{q} = \text{cart2-pt} \( q \) \)
let \( \equiv_{r} = \text{cart2-pt} \( r \) \)
let \( \equiv_{p} = \text{expansion-factor} \( p \) \)
let \( \equiv_{q} = \text{expansion-factor} \( q \) \)
let \( \equiv_{r} = \text{expansion-factor} \( r \) \)
from \( \forall q \in \text{hyp2} \cup S \) and \((\text{is-K2-isometry} J) \)
have \( \equiv_{q} \neq 0 \) by (rule expansion-factor)

from \( \forall p \in \text{hyp2} \cup S \) and \( \forall q \in \text{hyp2} \cup S \) and \( \forall r \in \text{hyp2} \cup S \) and \((\text{is-K2-isometry} J) \) and \( \text{expansion-factor-\text{sgn}-invariant} \)
have \( \text{sgn} \equiv_{p} = \text{sgn} \equiv_{q} \) and \( \text{sgn} \equiv_{r} = \text{sgn} \equiv_{q} \) by fast+
with \( \forall \equiv_{q} \neq 0 \)
have \( \equiv_{p} / \equiv_{q} > 0 \) and \( \equiv_{r} / \equiv_{q} > 0 \) by (simp-all add: \text{sgn-div})

from \( \forall p \in B_{R} \) \( \equiv_{p} \equiv_{q} \equiv_{r} \)
obtain \( k \) where \( \forall k \geq 0 \) and \( k \leq 1 \) and \( \equiv_{q} = k \ast \equiv_{r} + (1 - k) \ast \equiv_{p} \)
by (unfold real-euclid-B-def) (auto simp add: algebra-simps)

let \( \equiv_{c} = k \ast \equiv_{r} / \equiv_{q} \)
from \( \forall k \geq 0 \) and \( \forall \equiv_{r} / \equiv_{q} > 0 \) and \( \text{mult-nonneg-nonneg} \) of \( k \equiv_{r} / \equiv_{q} \)
have \( \forall k \geq 0 \) by simp

from \( \forall r \in \text{hyp2} \cup S \) and \( \forall p \in \text{hyp2} \cup S \) and \( \forall q \in \text{hyp2} \cup S \) and \((\text{is-K2-isometry} J) \) and \( \forall \equiv_{q} = k \ast \equiv_{r} + (1 - k) \ast \equiv_{p} \)
have \( \equiv_{q} = k \ast \equiv_{r} + (1 - k) \ast \equiv_{p} \) by (rule expansion-factor-linear)
with \( \forall \equiv_{q} \neq 0 \)
have \( \forall 1 - \equiv_{c} = (1 - k) \ast \equiv_{r} / \equiv_{q} \) by (simp add: \text{field-simps})
with \( \forall k \leq 1 \) and \( \forall \equiv_{r} / \equiv_{q} > 0 \)
and \( \text{mult-nonneg-nonneg} \) of \( 1 - k \equiv_{r} / \equiv_{q} \)
have \( \forall k \leq 1 \) by simp

let \( \equiv_{a} = \text{apply-cltn2} \( p \) \)
let \( \equiv_{b} = \text{apply-cltn2} \( q \) \)
let \( \equiv_{c} = \text{apply-cltn2} \( r \) \)
let \( \equiv_{d} = \text{cart2-pt} \( \equiv_{a} \) \)
let \( \equiv_{e} = \text{cart2-pt} \( \equiv_{b} \) \)
let \( \equiv_{f} = \text{cart2-pt} \( \equiv_{c} \) \)

165
let ?cpJ1 = cart2-append1 ?pJ
let ?cqJ1 = cart2-append1 ?qJ
let ?crJ1 = cart2-append1 ?rJ
from ⟨p ∈ hyp2 ∪ S⟩ and ⟨q ∈ hyp2 ∪ S⟩ and ⟨r ∈ hyp2 ∪ S⟩ and ⟨is-K2-isometry J⟩
  by (simp-all add: is-K2-isometry-z-non-zero)
from ⟨r ∈ hyp2 ∪ S⟩ and ⟨p ∈ hyp2 ∪ S⟩ and ⟨q ∈ hyp2 ∪ S⟩ and ⟨is-K2-isometry J⟩ and ⟨?eq = k *R ?crJ + (1 - k) *R ?cp⟩
  by (rule expansion-factor-linear-apply-cltn2)
  with ⟨1 - ?c = (1 - k) * ?ep / ?eq and (?eq ≠ 0)⟩
  by (simp add: scaleR-right-distrib)
with ⟨z-non-zero ?pJ⟩ and ⟨z-non-zero ?qJ⟩ and ⟨z-non-zero ?rJ⟩
have vector2-append1 ?eqJ
  by (simp add: cart2-append1)
  unfolding vector2-append1-def and vector-def
  by (simp add: vec-eq-iff forall-2 forall-3)
with ⟨?c ≥ 0⟩ and ⟨?c ≤ 1⟩
show BR ?cpJ1 ?cqJ1 ?crJ1
  by (unfold real-euclid-B-def) (simp add: algebra-simps cz1 [of ?c])
qed

theorem hyp2-axiom4: ∀ q a b c. ∃ x. BR q a x ∧ a x ≡K b c
proof (rule allI)+
  fix q a b c :: hyp2
  let ?pq = Rep-hyp2 q
  let ?pa = Rep-hyp2 a
  let ?pb = Rep-hyp2 b
  let ?pc = Rep-hyp2 c
  have ⟨?pq ∈ hyp2⟩ and ⟨?pa ∈ hyp2⟩ and ⟨?pb ∈ hyp2⟩ and ⟨?pc ∈ hyp2⟩
    by (rule Rep-hyp2)+
  let ?eq = cart2-pt ?pq
  let ?ca = cart2-pt ?pa
  let ?cb = cart2-pt ?pb
  let ?cc = cart2-pt ?pc
  let ?cp = cart2-pt ?pp
from ⟨?pb ∈ hyp2⟩ and ⟨?pc ∈ hyp2⟩ and extend-to-S [of ?pb ?pc]
  and somel-ex [of λ p. p ∈ S ∧ BR ?cb ?cc (cart2-pt p)]
have ⟨?pp ∈ S and BR ?cb ?cc ?cp⟩ by auto
  let ?pr = x. r. r ∈ S ∧ BR ?eq ?ca (cart2-pt r)
let \( ?cr = \text{cart2-pt} ?pr \)
from \( (?pq \in \text{hyp2}) \text{ and } (?pa \in \text{hyp2}) \text{ and } \text{extend-to-S} \text{ [of } ?pq \text{ ?pa]} \)
and \( \text{somel-ex} \text{ [of } \lambda r. \, r \in S \land B_R ?cq ?ca (\text{cart2-pt } r) \] \) have \( ?pr \in S \text{ and } B_R ?cq ?ca \) by auto

from \( (?pb \in \text{hyp2}) \text{ and } (?pa \in \text{hyp2}) \text{ and } (?pp \in S) \text{ and } (?pr \in S) \)
and \( \text{statement66-existence} \text{ [of } ?pb \text{ ?pa ?pp ?pr] \) obtain \( J \text{ where } \text{is-K2-isometry } J \)
and \( \text{apply-cltn2 } ?pb J = ?pa \) and \( \text{apply-cltn2 } ?pp J = ?pr \)
by auto
let \( ?px = \text{apply-cltn2 } ?pc J \)
let \( ?cx = \text{cart2-pt} ?px \)
let \( ?x = \text{Abs-hyp2 } ?px \)
from \( \text{is-K2-isometry } J \) and \( (?pc \in \text{hyp2}) \) have \( ?px \in \text{hyp2} \) by (rule statement60-one-way)
hence \( \text{Rep-hyp2 } ?x = ?px \) by (rule Abs-hyp2-inverse)

from \( (?pb \in \text{hyp2}) \text{ and } (?pc \in \text{hyp2}) \text{ and } (?pp \in S) \text{ and } \text{is-K2-isometry } J \)
and \( (B_R \text{ cart2-pt } (\text{apply-cltn2 } ?pb J)) \text{ ?x (cart2-pt } (\text{apply-cltn2 } ?pp J)) \)
by simp
with \( \text{apply-cltn2 } ?pb J = ?pa \) and \( \text{apply-cltn2 } ?pp J = ?pr \)
have \( B_R ?ca ?cx ?cr \) by simp
with \( (B_R ?cq ?ca ?cr) \) have \( B_R ?cq ?ca ?cx \) by (rule real-euclid.th3-5-1)
with \( \text{Rep-hyp2 } ?x = ?px \)
have \( B_K q a ?x \)
unfolding real-hyp2-B-def and hyp2-rep-def
by simp

have \( \text{Abs-hyp2 } ?pa = a \) by (rule Rep-hyp2-inverse)
with \( \text{apply-cltn2 } ?pb J = ?pa \)
have \( \text{hyp2-cltn2 } b J = a \) by (unfold hyp2-cltn2-def) simp

have \( \text{hyp2-cltn2 } c J = ?x \) unfolding hyp2-cltn2-def ..
with \( \text{is-K2-isometry } J \) and \( \text{hyp2-cltn2 } b J = a \)
have \( b c \equiv_K a ?x \)
by (unfold real-hyp2-C-def) (simp add: exI [of - J])
hence \( ?x \equiv_K b c \) by (rule hyp2.th2-2)
with \( (B_K q a ?x) \)
show \( \exists x. \, B_K q a x \land a x \equiv_K b c \) by (simp add: exI [of - ?x])
qed

8.9 More betweenness theorems

lemma hyp2-S-points-fix-line:
assumes \( a \in \text{hyp2} \) and \( p \in S \) and \( \text{is-K2-isometry } J \)
and \( \text{apply-cltn2 } a J = a \) (is \( ?aJ = a \))
and \( \text{apply-cltn2 } p J = p \) (is \( ?pJ = p \))
and \( \text{proj2-incident } a l \) and \( \text{proj2-incident } p l \) and \( \text{proj2-incident } b l \)
shows \( \text{apply-cltn2} \ b \ J = b \) (is \( \forall b J = b \))

proof –
let \( \forall l J = \text{apply-cltn2-line} \ l J \)
from \( \langle \text{proj2-incident} \ a \ l \rangle \) and \( \langle \text{proj2-incident} \ p \ l \rangle \)
have \( \text{proj2-incident} \ ?a J \ ?l J \) and \( \text{proj2-incident} \ ?p J \ ?l J \) by simp-all
with \( \langle ?a J = a \rangle \) and \( \langle ?p J = p \rangle \)
have \( \text{proj2-incident} \ a \ ?l J \) and \( \text{proj2-incident} \ p \ ?l J \) by simp-all
from \( \langle a \in \text{hyp2} \rangle \) and \( \langle \text{line-through-K2-intersect-S-again} \ [of \ a l] \rangle \)
obtain \( q \) where \( q \neq p \) and \( q \in S \) and \( \text{proj2-incident} \ q \ l \) by auto
let \( \forall q J = \text{apply-cltn2} \ q J \)
from \( \langle a \in \text{hyp2} \rangle \) and \( \langle p \in S \rangle \) and \( \langle q \in S \rangle \)
have \( a \neq p \) and \( a \neq q \) by (simp-all add: hyp2-S-not-equal)
from \( \langle a \neq p \rangle \) and \( \langle \text{proj2-incident} \ a \ b \rangle \) and \( \langle \text{proj2-incident} \ p \ b \rangle \)
and \( \langle \text{proj2-incident} \ a \ ?l J \rangle \) and \( \langle \text{proj2-incident} \ p \ ?l J \rangle \)
and \( \text{proj2-incident-unique} \)
have \( ?l J = l \) by auto
from \( \langle \text{proj2-incident} \ q \ l \rangle \)
have \( \text{proj2-incident} \ ?q J \ ?l J \) by simp
with \( \langle ?l J = l \rangle \)
have \( \text{proj2-incident} \ ?q J l \) by simp
from \( \langle q \in S \rangle \) and \( \langle \text{is-K2-isometry} \ J \rangle \)
have \( ?q J \in S \) by (unfold is-K2-isometry-def simp)
with \( \langle q \neq p \rangle \) and \( \langle p \in S \rangle \) and \( \langle q \in S \rangle \) and \( \langle \text{proj2-incident} \ p \ b \rangle \)
and \( \langle \text{proj2-incident} \ q \ b \rangle \) and \( \langle \text{proj2-incident} \ ?q J \ b \rangle \)
and \( \text{line-S-two-intersections-only} \)
have \( ?q J = p \lor ?q J = q \) by simp
have \( ?q J = q \)
proof (rule ccontr)
assume \( ?q J \neq q \)
with \( \langle ?q J = p \lor ?q J = q \rangle \) have \( ?q J = p \) by simp
with \( \langle ?p J = p \rangle \) have \( ?q J = ?p J \) by simp
with apply-cltn2-injective have \( q = p \) by fast
with \( q \neq p \) show False ..
qed
with \( \langle q \neq p \rangle \) and \( \langle a \neq p \rangle \) and \( \langle a \neq q \rangle \) and \( \langle \text{proj2-incident} \ p \ b \rangle \)
and \( \langle \text{proj2-incident} \ q \ b \rangle \) and \( \langle \text{proj2-incident} \ a \ b \rangle \)
and \( \langle ?p J = p \rangle \) and \( \langle ?a J = a \rangle \) and \( \langle \text{proj2-incident} \ b \ b \rangle \)
and \( \text{cltn2-three-point-line} \ [of \ p \ q \ a \ l \ b] \)
show \( \forall b J = b \) by simp
qed

lemma K2-isometry-endpoint-in-S:
assumes \( a \neq b \) and \( a \in \text{hyp2} \cup S \) and \( b \in \text{hyp2} \cup S \) and \( \text{is-K2-isometry} \ J \)
shows \( \text{apply-cltn2} \ (\text{endpoint-in-S} \ a \ b) \ J \)
= \textit{endpoint-in-S} (\textit{apply-cltn2} a J) (\textit{apply-cltn2} b J)


\textbf{proof} –

\begin{itemize}
\item \textbf{let} ?p = \textit{endpoint-in-S} a b
\item \textbf{from} (a \neq b) and \textit{apply-cltn2-injective} \textbf{have} ?aJ \neq ?bJ by fast
\item \textbf{from} (a \in \textit{hyp2} \cup S) and (b \in \textit{hyp2} \cup S) and \textit{(is-K2-isometry J)} and \textit{is-K2-isometry-hyp2-S} \textbf{have} \ ?aJ \in \textit{hyp2} \cup S and \ ?bJ \in \textit{hyp2} \cup S by simp-all
\item \textbf{let} ?ca = \textit{cart2-pt} a
\item \textbf{let} ?cb = \textit{cart2-pt} b
\item \textbf{let} ?cp = \textit{cart2-pt} ?p
\item \textbf{from} (a \in \textit{hyp2} \cup S) and (b \in \textit{hyp2} \cup S) have \ ?p \in S and \textit{B R} ?ca ?cb ?cp by (\textit{rule endpoint-in-S})+
\item \textbf{from} (?p \in S) and \textit{(is-K2-isometry J)} \textbf{have} \ ?pJ \in S by (unfold \textit{is-K2-isometry-def}) simp
\item \textbf{let} ?caJ = \textit{cart2-pt} ?aJ
\item \textbf{let} ?cbJ = \textit{cart2-pt} ?bJ
\item \textbf{let} ?cpJ = \textit{cart2-pt} ?pJ
\item \textbf{from} (a \in \textit{hyp2} \cup S) and (b \in \textit{hyp2} \cup S) and (?p \in S) and \textit{(is-K2-isometry J)} and \textit{B R} ?caJ ?cbJ ?cpJ by simp
\item \textbf{with} (?aJ \neq ?bJ) and (?aJ \in \textit{hyp2} \cup S) and (?bJ \in \textit{hyp2} \cup S) and (?pJ \in S) \textbf{show} \ ?pJ = \textit{endpoint-in-S} ?aJ ?bJ by \textbf{(rule endpoint-in-S-unique)}
\end{itemize}

\textit{qed}

\textbf{lemma} \textit{between-endpoint-in-S}:

\textbf{assumes} a \neq b and b \neq c and a \in \textit{hyp2} \cup S and b \in \textit{hyp2} \cup S and c \in \textit{hyp2} \cup S and \textit{B R} (\textit{cart2-pt} a) (\textit{cart2-pt} b) (\textit{cart2-pt} c) (\textit{is} \textit{B R} \ ?ca \ ?cb \ ?cc)

\textbf{shows} \textit{endpoint-in-S} a b = \textit{endpoint-in-S} b c (\textit{is} \ ?p = ?q)

\textbf{proof} –

\begin{itemize}
\item \textbf{from} (?b \neq c) and (?b \in \textit{hyp2} \cup S) and (?c \in \textit{hyp2} \cup S) and \textit{hyp2-S-cart2-inj} \textbf{have} \ ?cb \neq ?cc by auto
\item \textbf{let} ?cq = \textit{cart2-pt} ?q
\item \textbf{from} (?b \in \textit{hyp2} \cup S) and (?c \in \textit{hyp2} \cup S) have \ ?q \in S and \textit{B R} ?cb ?cc ?cq by \textbf{(rule endpoint-in-S)}+
\item \textbf{from} (?cb \neq ?cc) and \textit{B R} ?ca ?cb ?cc and \textit{B R} ?cb ?cc ?cq, \textbf{have} \textit{B R} ?ca ?cb ?cq by \textbf{(rule \textit{real-euclid.th3-7-2})}
\item \textbf{with} (?a \neq b) and (?a \in \textit{hyp2} \cup S) and (?b \in \textit{hyp2} \cup S) and (?q \in S) \textbf{have} \ ?q = ?p by \textbf{(rule endpoint-in-S-unique)}
\item \textbf{thus} \ ?p = ?q ..
\end{itemize}

\textit{qed}
lemma hyp2-extend-segment-unique:
  assumes \( a \neq b \) and \( B_K \ a b c \) and \( B_K \ a b d \) and \( b c \equiv_K b d \)
  shows \( c = d \)
proof cases
  assume \( b = c \)
  with \( b c \equiv_K b d \) show \( c = d \) by (simp add: hyp2.A3-reversed)
next
  assume \( b \neq c \)

have \( b \neq d \)
proof (rule ccontr)
  assume \( \neg b \neq d \)
  hence \( b = d \) by simp
  with \( b c \equiv_K b d \) have \( b c \equiv_K b b \) by simp
  hence \( b = c \) by (rule hyp2.A3′)
  with \( b \neq c \) show False ..
qed

with \( a \neq b \) and \( b \neq c \)
have Rep-hyp2 \( a \neq \) Rep-hyp2 \( b \) (is \(?pa \neq ?pb\))
  and Rep-hyp2 \( b \neq \) Rep-hyp2 \( c \) (is \(?pb \neq ?pc\))
  and Rep-hyp2 \( b \neq \) Rep-hyp2 \( d \) (is \(?pb \neq ?pd\))
  by (simp-all add: Rep-hyp2-inject)

have \(?pa \in \) hyp2 and \(?pb \in \) hyp2 and \(?pc \in \) hyp2 and \(?pd \in \) hyp2
  by (rule Rep-hyp2)+

let \(?pp = endpoint-in-S \) ?pb \( ?pc\)
let \(?ca = cart2-pt \) ?pa
let \(?cb = cart2-pt \) ?pb
let \(?cc = cart2-pt \) ?pc
let \(?cd = cart2-pt \) ?pd
let \(?cp = cart2-pt \) ?pp
from \(?pb \in \) hyp2\) and \(?pc \in \) hyp2\)
  have \(?pp \in \) S and \( B_R \) ?cb \( ?cc \) ?cp by (simp-all add: endpoint-in-S)
from \( b c \equiv_K b d \)
obtain \( J \) where is-K2-isometry \( J \)
  and hyp2-cltn2 \( b J = b \) and hyp2-cltn2 \( c J = d \)
  by (unfold real-hyp2-C-def) auto

from \( hyp2-cltn2 \) \( b J = b \) and \( hyp2-cltn2 \) \( c J = d \)
have Rep-hyp2 \( hyp2-cltn2 \) \( b J \) = \(?pb\)
  and Rep-hyp2 \( hyp2-cltn2 \) \( c J \) = \(?pd\)
  by simp-all
with \( is-K2-isometry \) \( J \),
have apply-cltn2 \(?pb \) \( J \) = \(?pb\) and apply-cltn2 \(?pc \) \( J \) = \(?pd\)
  by (simp-all add: Rep-hyp2-cltn2)
from \( \langle B_K \ a \ b \ c \rangle \) and \( \langle B_K \ a \ b \ d \rangle \)

have \( B_R \ ?ca \ ?cb \ ?cc \) and \( B_R \ ?ca \ ?cb \ ?cd \)

unfolding real-hyp2-B-def and hyp2-rep-def.

from \( \langle ?pb \neq ?pc \rangle \) and \( \langle ?pb \in \text{hyp2} \rangle \) and \( \langle ?pc \in \text{hyp2} \rangle \) and \( \langle \text{is-K2-isometry} \ J \rangle \)

have apply-cltn2 \( ?pp \ J \)

= endpoint-in-S \( \langle \text{apply-cltn2} \ ?pb \ J \rangle \) (apply-cltn2 \( ?pc \ J \))

by (simp add: K2-isometry-endpoint-in-S)

also from \( \langle \text{apply-cltn2} \ ?pb \ J = ?pb \rangle \) and \( \langle \text{apply-cltn2} \ ?pc \ J = ?pd \rangle \)

have \( \cdots \) = endpoint-in-S \( ?pa \ ?pb \) by simp

also from \( \langle ?pa \neq ?pb \rangle \) and \( \langle ?pb \neq ?pd \rangle \)

and \( \langle ?pa \in \text{hyp2} \rangle \) and \( \langle ?pb \in \text{hyp2} \rangle \) and \( \langle ?pd \in \text{hyp2} \rangle \) and \( \langle B_R \ ?ca \ ?cb \ ?cd \rangle \)

have \( \cdots \) = endpoint-in-S \( ?pa \ ?pb \) by (simp add: between-endpoint-in-S)

also from \( \langle ?pa \neq ?pb \rangle \) and \( \langle ?pb \neq ?pc \rangle \)

and \( \langle ?pa \in \text{hyp2} \rangle \) and \( \langle ?pb \in \text{hyp2} \rangle \) and \( \langle ?pc \in \text{hyp2} \rangle \) and \( \langle B_R \ ?ca \ ?cb \ ?cc \rangle \)

have \( \cdots \) = endpoint-in-S \( ?pa \ ?pc \ ) by (simp add: between-endpoint-in-S)

finally have apply-cltn2 \( ?pp \ J = ?pp \).

from \( \langle ?pb \in \text{hyp2} \rangle \) and \( \langle ?pc \in \text{hyp2} \rangle \) and \( \langle ?pp \in S \rangle \)

have \( \text{z-non-zero} \ ?pb \) and \( \text{z-non-zero} \ ?pc \) and \( \text{z-non-zero} \ ?pp \)

by (simp add: hyp2-S-z-non-zero)

with \( \langle B_R \ ?cb \ ?cc \ ?cp \rangle \) and euclid-B-cart2-common-line \( \langle \text{of} \ ?pb \ ?pc \ ?pp \rangle \)

obtain \( l \) where proj2-incident \( ?pb \ l \) and proj2-incident \( ?pp \ l \)

and proj2-incident \( ?pc \ l \)

by auto

with \( \langle ?pb \in \text{hyp2} \rangle \) and \( \langle ?pp \in S \rangle \) and \( \langle \text{is-K2-isometry} \ J \rangle \)

and \( \langle \text{apply-cltn2} \ ?pb \ J = ?pb \rangle \) and \( \langle \text{apply-cltn2} \ ?pp \ J = ?pp \rangle \)

have apply-cltn2 \( ?pc \ J = ?pc \) by (rule hyp2-S-points-fix-line)

with \( \langle \text{apply-cltn2} \ ?pc \ J = ?pd \rangle \) have \( ?pc = ?pd \) by simp

thus \( c = d \) by (subt Rep-hyp2-inject [symmetric])

qed

\begin{enumerate}
\item \textbf{lemma line-S-match-intersections:}
\item \textbf{assumes} \( p \neq q \) and \( r \neq s \) and \( p \in S \) and \( q \in S \) and \( r \in S \) and \( s \in S \)
\item \textbf{and} proj2-set-Col \( \{ p,q,r,s \} \)
\item \textbf{shows} \( (p = r \land q = s) \lor (q = r \land p = s) \)
\item \textbf{proof –}
\item \textbf{from} \( \langle \text{proj2-set-Col} \ \{ p,q,r,s \} \rangle \)
\item \textbf{obtain} \( l \) where proj2-incident \( p \ l \) and proj2-incident \( q \ l \)
\item \textbf{and} proj2-incident \( r \ l \) and proj2-incident \( s \ l \)
\item \textbf{by} (unfold proj2-set-Col-def) auto
\item \textbf{with} \( \langle r \neq s \rangle \) and \( \langle p \in S \rangle \) and \( \langle q \in S \rangle \) and \( \langle r \in S \rangle \) and \( \langle s \in S \rangle \)
\item \textbf{have} \( p = r \lor p = s \) and \( q = r \lor q = s \)
\item \textbf{by} (simp-all add: line-S-two-intersections-only)
\item \textbf{show} \( (p = r \land q = s) \lor (q = r \land p = s) \)
\item \textbf{proof cases}
\item \textbf{assume} \( p = r \)
\item \textbf{with} \( \langle p \neq q \rangle \) and \( \langle q = r \lor q = s \rangle \)
\end{enumerate}
show \((p = r \land q = s) \lor (q = r \land p = s)\) by simp

next
assume \(p \neq r\)
with \((p = r \lor p = s)\) have \(p = s\) by simp
with \((p \neq q)\) and \((q = r \lor q = s)\)
show \((p = r \land q = s)\) by simp
qed

qed

definition are-endpoints-in-S :: \([\text{proj}2, \text{proj}2, \text{proj}2, \text{proj}2]\) ⇒ bool where
arc-endpoints-in-S \(p q a b\)
\(= p \neq q \land p \in S \land q \in S \land a \in \text{hyp}2 \land b \in \text{hyp}2 \land \text{proj}2\text{-set-Col} \{p,q,a,b\}\)

lemma arc-endpoints-in-S:
assumes \(p \neq q\) and \(a \neq b\) and \(p \in S\) and \(q \in S\) and \(a \in \text{hyp}2 \lor S\)
and \(b \in \text{hyp}2 \lor S\) and \(\text{proj}2\text{-set-Col} \{p,q,a,b\}\)
shows \((p = \text{endpoint-in-S} a b \land q = \text{endpoint-in-S} b a)\)
\(\lor (q = \text{endpoint-in-S} a b \land p = \text{endpoint-in-S} b a)\)
\((\text{is} (p = \text{?r} \land q = \text{?s}) \lor (q = \text{?r} \land p = \text{?s}))\)
proof
from \((a \neq b)\) and \((a \in \text{hyp}2 \lor S)\) and \((b \in \text{hyp}2 \lor S)\)
have \(\text{?r} \neq \text{?s}\) by (simp add: endpoint-in-S-swap)

from \((a \in \text{hyp}2 \lor S)\) and \((b \in \text{hyp}2 \lor S)\)
have \(\text{?r} \in S\) and \(\text{?s} \in S\) by (simp-all add: endpoint-in-S)

from \(\text{proj}2\text{-set-Col} \{p,q,a,b\}\)
obtain \(l\) where \(\text{proj}2\text{-incident} p l\) and \(\text{proj}2\text{-incident} q l\)
and \(\text{proj}2\text{-incident} a l\) and \(\text{proj}2\text{-incident} b l\)
by (unfold \(\text{proj}2\text{-set-Col-def}\)) auto

from \((a \neq b)\) and \((a \in \text{hyp}2 \lor S)\) and \((b \in \text{hyp}2 \lor S)\) and \(\text{proj}2\text{-incident} a l\)
and \(\text{proj}2\text{-incident} b l\)
have \(\text{proj}2\text{-incident} \text{?r} l\) and \(\text{proj}2\text{-incident} \text{?s} l\)
by (simp-all add: endpoint-in-S-incident)
with \(\text{proj}2\text{-incident} p b\) and \(\text{proj}2\text{-incident} q b\)
have \(\text{proj}2\text{-set-Col} \{p,q,\text{?r},\text{?s}\}\)
by (unfold \(\text{proj}2\text{-set-Col-def}\)) (simp add: exI \([\text{of - l}]\))
with \((p \neq q)\) and \((\text{?r} \neq \text{?s})\) and \((p \in S)\) and \((q \in S)\) and \((\text{?r} \in S)\) and \((\text{?s} \in S)\)
show \((p = \text{?r} \land q = \text{?s}) \lor (q = \text{?r} \land p = \text{?s})\)
by (rule line-S-match-intersections)
qed

lemma arc-endpoints-in-S:
assumes \(a \neq b\) and \(\text{arc-endpoints-in-S} p q a b\)
shows \((p = \text{endpoint-in-S} a b \land q = \text{endpoint-in-S} b a)\)
\(\lor (q = \text{endpoint-in-S} a b \land p = \text{endpoint-in-S} b a)\)
using assms
by (unfold are-endpoints-in-S-def) (simp add: are-endpoints-in-S')

lemma  S-intersections-endpoints-in-S:
  assumes a ≠ 0 and b ≠ 0 and proj2-abs a ≠ proj2-abs b (is ?pa ≠ ?pb) and proj2-abs a ∈ hyp2 and proj2-abs b ∈ hyp2 ∪ S
proof –
  from (a ≠ 0) and (b ≠ 0) and (?pa ≠ ?pb) and (?pa ∈ hyp2)
  have ?pp ≠ ?pq by (simp add: S-intersections-distinct)

from (a ≠ 0) and (b ≠ 0) and (?pa ≠ ?pb) and (?pa ∈ hyp2)
  have ?pp ∈ S and ?pq ∈ S
    by (simp-all add: S-intersections-in-S)

let I = proj2-line-through ?pa ?pb
have proj2-incident ?pa I and proj2-incident ?pb I
  by (rule proj2-line-through-incident)+
with (a ≠ 0) and (b ≠ 0) and (?pa ≠ ?pb)
  have proj2-incident ?pp I and proj2-incident ?pq I
    by (rule S-intersections-incident)+
with (?proj2-incident ?pa I) and (?proj2-incident ?pb I)
  have proj2-set-Col {?pp, ?pq, ?pa, ?pb}
    by (unfold proj2-set-Col-def) (simp add: exI [of - ?I])
with (?pp ≠ ?pq) and (?pa ≠ ?pb) and (?pp ∈ S) and (?pq ∈ S) and (?pa ∈ hyp2)
  and (?pb ∈ hyp2 ∪ S)
    by (simp add: are-endpoints-in-S')
qed

lemma  between-endpoints-in-S:
  assumes a ≠ b and a ∈ hyp2 ∪ S and b ∈ hyp2 ∪ S
  shows B_R
  (cart2-pt (endpoint-in-S a b)) (cart2-pt a) (cart2-pt (endpoint-in-S b a))
  (is B_R ?cp ?ca ?cq)
proof –
  let ?cb = cart2-pt b
  from (b ∈ hyp2 ∪ S) and (a ∈ hyp2 ∪ S) and (a ≠ b)
  have ?cb ≠ ?ca by (auto simp add: hyp2-S-cart2-inj)

from (a ∈ hyp2 ∪ S) and (b ∈ hyp2 ∪ S)

from (B_R ?ca ?cb ?cp) have B_R ?cp ?cb ?ca by (rule real-euclid.th3-2)
with (?cb ≠ ?ca) and (B_R ?cb ?ca ?cq)
show \( B_R \ ?cp \ ?ca \ ?cq \) by (simp add: real-euclid.th3-7-1)

qed

lemma \( S\text{-hyp2-S-cart2-append1} \):
  assumes \( p \neq q \) and \( p \in S \) and \( q \in S \) and \( a \in \text{hyp2} \)
  and \( \text{proj2-incident} \ p \ l \) and \( \text{proj2-incident} \ q \ l \) and \( \text{proj2-incident} \ a \ l \)
  shows \( \exists \ k. \ k > 0 \land k < 1 \land \text{cart2-append1} \ a = k \ast_R \text{cart2-append1} \ q + (1 - k) \ast_R \text{cart2-append1} \ p \)

proof –
  from \( p \in S \) and \( q \in S \) and \( (a \in \text{hyp2}) \)
  have \( \text{z-non-zero} \ p \) and \( \text{z-non-zero} \ q \) and \( \text{z-non-zero} \ a \)
  by (simp-all add: hyp2-S-z-non-zero)

  from assms
  have \( \text{BIR}(\text{cart2-pt} \ p) (\text{cart2-pt} \ a) (\text{cart2-pt} \ q) (\text{is \ BIR?cp?ca?cq}) \)
  by (simp add: hyp2-incident-in-middle)

  from \( p \in S \) and \( q \in S \) and \( (a \in \text{hyp2}) \)
  have \( a \neq p \) and \( a \neq q \) by (simp-all add: hyp2-S-not-equal)

  with \( \{\text{z-non-zero} \ p, \text{z-non-zero} \ a, \text{z-non-zero} \ q\} \)
  and \( \text{BIR} \ ?cp \ ?ca \ ?cq \)
  show \( \exists \ k. \ k > 0 \land k < 1 \land \text{cart2-append1} \ a = k \ast_R \text{cart2-append1} \ q + (1 - k) \ast_R \text{cart2-append1} \ p \)
  by (rule cart2-append1-between-strict)

qed

lemma \( \text{are-endpoints-in-S-swap-34} \):
  assumes \( \text{are-endpoints-in-S} \ p \ q \ a \ b \)
  shows \( \text{are-endpoints-in-S} \ p \ q \ b \ a \)

proof –
  have \( \{p, q, b, a\} = \{p, q, a, b\} \) by auto
  with \( \text{are-endpoints-in-S} \ p \ q \ a \ b \)
  show \( \text{are-endpoints-in-S} \ p \ q \ b \ a \) by (unfold are-endpoints-in-S-def) simp

qed

lemma \( \text{proj2-set-Col-endpoints-in-S} \):
  assumes \( a \neq b \) and \( a \in \text{hyp2} \cup S \) and \( b \in \text{hyp2} \cup S \)
  shows \( \text{proj2-set-Col} \ \{\text{endpoint-in-S} \ a \ b, \text{endpoint-in-S} \ b \ a\} \)
  (is \( \text{proj2-set-Col} \ \{?p, ?q, a, b\} \))

proof –
  let \( \ell = \text{proj2-line-through} \ a \ b \)
  have \( \text{proj2-incident} \ a \ \ell \) and \( \text{proj2-incident} \ b \ \ell \)
  by (rule proj2-line-through-incident)+
  with \( a \neq b \) and \( (a \in \text{hyp2} \cup S) \) and \( (b \in \text{hyp2} \cup S) \)
  have \( \text{proj2-incident} \ ?p \ \ell \) and \( \text{proj2-incident} \ ?q \ \ell \)
  by (simp-all add: endpoint-in-S-incident)
  with \( \text{proj2-incident} \ a \ \ell \) and \( \text{proj2-incident} \ b \ \ell \)
  show \( \text{proj2-set-Col} \ \{?p, ?q, a, b\} \)

174
by (unfold proj2-set-Col-def) (simp add: ex1 [of - ?l])

qed

lemma endpoints-in-S-are-endpoints-in-S:
  assumes a ≠ b and a ∈ hyp2 and b ∈ hyp2
  shows are-endpoints-in-S (endpoint-in-S a b) (endpoint-in-S b a) a b
  (is are-endpoints-in-S ?p ?q a b)
proof −
  from ⟨a ≠ b⟩ and ⟨a ∈ hyp2⟩ and ⟨b ∈ hyp2⟩
  have ?p ≠ ?q by (simp add: endpoint-in-S-swap)

from ⟨a ∈ hyp2⟩ and ⟨b ∈ hyp2⟩
have ?p ∈ S and ?q ∈ S by (simp-all add: endpoint-in-S)

from assms
  have proj2-set-Col {?p, ?q, a, b} by (simp add: proj2-set-Col-endpoints-in-S)
  with ⟨?p ≠ ?q⟩ and ⟨?p ∈ S⟩ and ⟨?q ∈ S⟩ and ⟨a ∈ hyp2⟩ and ⟨b ∈ hyp2⟩
  show are-endpoints-in-S ?p ?q a b by (unfold are-endpoints-in-S-def) simp

qed

lemma endpoint-in-S-S-hyp2-distinct:
  assumes p ∈ S and a ∈ hyp2 ∪ S and p ≠ a
  shows endpoint-in-S p a ≠ p
proof −
  from ⟨a ∈ hyp2⟩ and ⟨p ∈ S⟩ and ⟨a ∈ hyp2 ∪ S⟩
  have B_I R (cart2-pt p) (cart2-pt a) (cart2-pt (endpoint-in-S p a))
    by (simp add: endpoint-in-S)

assume endpoint-in-S p a = p
  with ⟨B_I R (cart2-pt p) (cart2-pt a) (cart2-pt (endpoint-in-S p a))⟩
  have cart2-pt p = cart2-pt a by (simp add: real-euclid.A6)
  with ⟨p ∈ S⟩ and ⟨a ∈ hyp2 ∪ S⟩ have p = a by (simp add: hyp2-S-cart2-inj)
  with ⟨p ≠ a⟩ show False ..

qed

lemma endpoint-in-S-S-strict-hyp2-distinct:
  assumes p ∈ S and a ∈ hyp2
  shows endpoint-in-S p a ≠ p
proof −
  from ⟨a ∈ hyp2⟩ and ⟨p ∈ S⟩
  have p ≠ a by (rule hyp2-S-not-equal [symmetric])
  with assms
  show endpoint-in-S p a ≠ p by (simp add: endpoint-in-S-S-hyp2-distinct)

qed

lemma end-and-opposite-are-endpoints-in-S:
  assumes a ∈ hyp2 and b ∈ hyp2 and p ∈ S
  and proj2-incident a l and proj2-incident b l and proj2-incident p l
  shows are-endpoints-in-S p (endpoint-in-S p b) a b

175
(is are-endpoints-in-S p ?q a b)

proof –
from ⟨p ∈ S⟩ and ⟨b ∈ hyp2⟩ have p ≠ ?q by (rule endpoint-in-S-S-strict-hyp2-distinct [symmetric])

from ⟨p ∈ S⟩ and ⟨b ∈ hyp2⟩ have ?q ∈ S by (simp add: endpoint-in-S)
from ⟨b ∈ hyp2⟩ and ⟨p ∈ S⟩ have p ≠ b by (rule hyp2-S-not-equal [symmetric])
with ⟨p ∈ S⟩ and ⟨b ∈ hyp2⟩ and ⟨proj2-incident p l⟩ and ⟨proj2-incident b l⟩ have proj2-incident ?q l by (simp add: endpoint-in-S-incident)
with ⟨proj2-incident p l⟩ and ⟨proj2-incident a l⟩ and ⟨proj2-incident b l⟩ have proj2-set-Col {p, ?q, a, b} by (unfold proj2-set-Col-def) (simp add: exI [of - l])

moreover from ⟨p ∈ S⟩ and ⟨b ∈ hyp2⟩ and ⟨proj2-incident a l⟩ and ⟨proj2-incident b l⟩ have proj2-set-Col {p, ?q, a, b} by (simp add: endpoint-in-S-incident)
ultimately show are-endpoints-in-S p ?q a b by (unfold are-endpoints-in-S-def) simp qed

lemma real-hyp2-B-hyp2-cltn2:
assumes is-K2-isometry J and B K a b c
shows B K (hyp2-cltn2 a J) (hyp2-cltn2 b J) (hyp2-cltn2 c J)
(is B K ?aJ ?bJ ?cJ)

proof –
from ⟨B K a b c⟩ have B R (hyp2-rep a) (hyp2-rep b) (hyp2-rep c) by (unfold real-hyp2-B-def)
with ⟨is-K2-isometry J⟩ have B R (cart2-pt (apply-cltn2 (Rep-hyp2 a) J)) (cart2-pt (apply-cltn2 (Rep-hyp2 b) J)) (cart2-pt (apply-cltn2 (Rep-hyp2 c) J)) by (unfold hyp2-rep-def) (simp add: Rep-hyp2-statement-63)
moreover from ⟨is-K2-isometry J⟩ have apply-cltn2 (Rep-hyp2 a) J ∈ hyp2 and apply-cltn2 (Rep-hyp2 b) J ∈ hyp2 and apply-cltn2 (Rep-hyp2 c) J ∈ hyp2 by (rule apply-cltn2-Rep-hyp2)+
ultimately show B K (hyp2-cltn2 a J) (hyp2-cltn2 b J) (hyp2-cltn2 c J) unfolding hyp2-cltn2-def and real-hyp2-B-def and hyp2-rep-def by (simp add: Abs-hyp2-inverse)

qed

lemma real-hyp2-C-hyp2-cltn2:
assumes is-K2-isometry J
shows a b ≡ K (hyp2-cltn2 a J) (hyp2-cltn2 b J) (is a b ≡ K ?aJ ?bJ)
using assms by (unfold real-hyp2-C-def) (simp add: exI [of - J])

8.10 Perpendicularity

definition M-perp :: proj2-line ⇒ proj2-line ⇒ bool where
M-perp l m ≜ proj2-incident (pole l) m
lemma \( \text{M-perp-sym} \):
\begin{align*}
\text{assumes } & \text{M-perp } l \; m \\
\text{shows } & \text{M-perp } m \; l \\
\text{proof} & \quad \text{from } \langle \text{M-perp } l \; m \rangle \text{ have proj2-incident (pole } l \rangle \; m \text{ by (unfold M-perp-def)} \\
\text{hence } & \text{proj2-incident (pole } m \rangle \; (\text{polar (pole } l \rangle)) \text{ by (rule incident-pole-polar)} \\
\text{hence } & \text{proj2-incident (pole } m \rangle \; l \text{ by (simp add: polar-pole)} \\
\text{thus } & \text{M-perp } m \; l \text{ by (unfold M-perp-def)} \\
\text{qed}
\end{align*}

lemma \( \text{M-perp-to-compass} \):
\begin{align*}
\text{assumes } & \text{M-perp } l \; m \text{ and } a \in \text{hyp2 and proj2-incident } a \; l \\
\text{and } & b \in \text{hyp2 and proj2-incident } b \; m \\
\text{shows } & \exists J \cdot \text{is-K2-isometry } J \\
\quad & \land \text{apply-cltn2-line equator } J = l \land \text{apply-cltn2-line meridian } J = m \\
\text{proof} & \quad \text{from } \langle a \in \text{K2} \rangle \text{ and } \langle \text{proj2-incident } a \; l \rangle \text{ and line-through-K2-intersect-S-twice } [\text{of } a \; l] \\
\text{obtain } & p \text{ and } q \text{ where } p \neq q \text{ and } p \in S \text{ and } q \in S \\
\text{and } & \text{proj2-incident } p \; l \text{ and proj2-incident } q \; l \\
\text{by } & \text{auto} \\
\text{have } & \exists r \cdot r \in S \land r \notin \{p,q\} \land \text{proj2-incident } r \; m \\
\text{proof} & \quad \text{cases} \\
\text{assume } & \text{proj2-incident } p \; m \\
\text{from } & \langle b \in \text{K2} \rangle \text{ and } \langle \text{proj2-incident } b \; m \rangle \\
\text{and } & \text{line-through-K2-intersect-S-again } [\text{of } b \; m] \\
\text{obtain } & r \text{ where } r \in S \land r \neq p \text{ and proj2-incident } r \; m \text{ by auto} \\
\text{have } & r \notin \{p,q\} \\
\text{proof} & \quad \text{assume } r \in \{p,q\} \\
\text{with } & (r \neq p) \text{ have } r = q \text{ by simp} \\
\text{with } & \langle \text{proj2-incident } r \; m \rangle \text{ have proj2-incident } q \; m \text{ by simp} \\
\text{with } & \langle \text{proj2-incident } p \; l \rangle \text{ and } \langle \text{proj2-incident } q \; l \rangle \\
\text{and } & \langle \text{proj2-incident } p \; m \rangle \text{ and } \langle \text{proj2-incident } q \; m \rangle \text{ and } (p \neq q) \\
\text{and } & \text{proj2-incident-unique } [\text{of } p \; l \; q \; m] \\
\text{have } & l = m \text{ by simp} \\
\text{with } & \langle \text{M-perp } l \; m \rangle \text{ have } \text{M-perp } l \; l \text{ by simp} \\
\text{hence } & \text{proj2-incident (pole } l \rangle \; l \text{ (is proj2-incident } ?s \; l) \\
\text{by } & \text{(unfold M-perp-def)} \\
\text{hence } & \text{proj2-incident } ?s \; (\text{polar } ?s) \text{ by (subst polar-pole)} \\
\text{hence } & ?s \in S \text{ by (simp add: incident-own-polar-in-S)} \\
\text{with } & \langle p \in S \rangle \text{ and } \langle q \in S \rangle \text{ and } \langle \text{proj2-incident } p \; l \rangle \text{ and } \langle \text{proj2-incident } q \; l \rangle \\
\text{and } & \text{point-in-S-polar-is-tangent } [\text{of } ?s] \\
\text{have } & p = ?s \text{ and } q = ?s \text{ by (auto simp add: polar-pole)} \\
\text{with } & (p \neq q) \text{ show False by simp}
\end{align*}
qed
with \( r \in S \) and \( \text{proj2-incident } r \ m \)

show \( \exists \ r. \ r \in S \land r \notin \{p,q\} \land \text{proj2-incident } r \ m \)
by (simp add: exI [of \( r \)])

next
assume \( \lnot \text{proj2-incident } p \ m \)

from \( b \in K2 \) and \( \text{proj2-incident } b \ m \)

and \text{line-through-K2-intersect-S-again } [of \ b \ m ]

obtain \( r \) where \( r \in S \) and \( r \neq q \) and \( \text{proj2-incident } r \ m \) by auto

from \( \lnot \text{proj2-incident } p \ m \) and \( \text{proj2-incident } r \ m \) have \( r \neq p \) by auto
with \( r \in S \) and \( r \neq q \) and \( \text{proj2-incident } r \ m \)

show \( \exists \ r. \ r \in S \land r \notin \{p,q\} \land \text{proj2-incident } r \ m \)
by (simp add: exI [of \( r \)])

qed
then obtain \( r \) where \( r \in S \) and \( r \notin \{p,q\} \) and \( \text{proj2-incident } r \ m \) by auto

from \( p \in S \) and \( q \in S \) and \( r \in S \) and \( p \neq q \) and \( r \notin \{p,q\} \)

and \text{statement65-special-case } [of \ p \ q \ r ]

obtain \( J \) where \text{is-K2-isometry } J \text{ and } \text{apply-cltn2 east } J = p

and \text{apply-cltn2 west } J = q \text{ and } \text{apply-cltn2 north } J = r

and \text{apply-cltn2 far-north } J = \text{proj2-intersection } (\text{polar } p ) (\text{polar } q )
by auto

from \( \text{apply-cltn2 east } J = p \) and \( \text{apply-cltn2 west } J = q \)

and \( \text{proj2-incident } p \ l \) and \( \text{proj2-incident } q \ l \)

have \( \text{proj2-incident } (\text{apply-cltn2 east } J ) \ l \)

and \( \text{proj2-incident } (\text{apply-cltn2 west } J ) \ l \)
by simp-all

with \text{east-west-distinct} and \text{east-west-on-equator}

have \text{apply-cltn2-line equator } J = l \text{ by (rule apply-cltn2-line-unique)}

from \( \text{apply-cltn2 north } J = r \) and \( \text{proj2-incident } r \ m \)

have \( \text{proj2-incident } (\text{apply-cltn2 north } J ) \ m \) by simp

from \( p \neq q \) and \text{polar-inj} have \( \text{polar } p \neq \text{polar } q \) by fast

from \( \text{proj2-incident } p \ l \) and \( \text{proj2-incident } q \ l \)

have \( \text{proj2-incident } (\text{pole } l ) (\text{polar } p ) \)

and \( \text{proj2-incident } (\text{pole } l ) (\text{polar } q ) \)
by (simp-all add: incident-pole-polar)

with \( \text{polar } p \neq \text{polar } q \)

have \( \text{pole } l = \text{proj2-intersection } (\text{polar } p ) (\text{polar } q ) \)
by (rule proj2-intersection-unique)

with \( \text{apply-cltn2 far-north } J = \text{proj2-intersection } (\text{polar } p ) (\text{polar } q ) \)

have \( \text{apply-cltn2 far-north } J = \text{pole } l \) by simp

with \( \text{M-perp } l \ m \)

have \( \text{proj2-incident } (\text{apply-cltn2 far-north } J ) \ m \) by (unfold \text{M-perp-def}) simp

178
with north-far-north-distinct and north-south-far-north-on-meridian
and ⟨proj2-incident (apply-cltn2 north J) m⟩
have apply-cltn2-line meridian J = m by (simp add: apply-cltn2-line-unique)
with ⟨is-K2-isometry J⟩ and ⟨apply-cltn2-line equator J = l⟩
show ∃ J. is-K2-isometry J
and apply-cltn2-line equator J = l ∧ apply-cltn2-line meridian J = m
by (simp add: exI [of - J])
qed

definition drop-perp :: proj2 ⇒ proj2-line ⇒ proj2-line where
drop-perp p l ≜ proj2-line-through p (pole l)

lemma drop-perp-incident: proj2-incident p (drop-perp p l)
by (unfold drop-perp-def) (rule proj2-line-through-incident)

lemma drop-perp-perp: M-perp l (drop-perp p l)
by (unfold drop-perp-def M-perp-def) (rule proj2-line-through-incident)

definition perp-foot :: proj2 ⇒ proj2-line ⇒ proj2 where
perp-foot p l ≜ proj2-intersection l (drop-perp p l)

lemma perp-foot-incident:
shows proj2-incident (perp-foot p l) l
and proj2-incident (perp-foot p l) (drop-perp p l)
by (unfold perp-foot-def) (rule proj2-intersection-incident)+

lemma M-perp-hyp2:
assumes M-perp l m and a ∈ hyp2 and proj2-incident a l and b ∈ hyp2
and proj2-incident b m and proj2-incident c l and proj2-incident c m
shows c ∈ hyp2
proof −
from ⟨M-perp l m⟩ and ⟨a ∈ hyp2⟩ and ⟨proj2-incident a l⟩ and ⟨b ∈ hyp2⟩
and ⟨proj2-incident b m⟩ and ⟨M-perp-to-compass [of l m a b]⟩
obtain J where is-K2-isometry J and apply-cltn2-line equator J = l
and apply-cltn2-line meridian J = m
by auto

from ⟨is-K2-isometry J⟩ and K2-centre-in-K2
have apply-cltn2 K2-centre J ∈ hyp2
by (rule statement60-one-way)

from ⟨proj2-incident c l⟩ and ⟨apply-cltn2-line equator J = l⟩
and ⟨proj2-incident c m⟩ and ⟨apply-cltn2-line meridian J = m⟩
have proj2-incident c (apply-cltn2-line equator J)
and proj2-incident c (apply-cltn2-line meridian J)
by simp-all
with equator-meridian-distinct and K2-centre-on-equator-meridian
have apply-cltn2 K2-centre J = c by (rule apply-cltn2-unique)
with ⟨apply-cltn2 K2-centre J ∈ hyp2⟩ show c ∈ hyp2 by simp
lemma perp-foot-hyp2:
assumes $a \in \text{hyp2}$ and $\text{proj2-incident } a \ l$ and $b \in \text{hyp2}$
shows $\text{perp-foot } b \ l \in \text{hyp2}$
using $\text{drop-perp-perp [of } b \ l]\text{ and (a } \in \text{hyp2) and (proj2-incident } a \ l)$
and $b \in \text{hyp2}$ and $\text{drop-perp-incident [of } b \ l]$ and $\text{perp-foot-incident [of } b \ l]$ by (rule M-perp-hyp2)

definition perp-up :: $\text{proj2 } \Rightarrow \text{proj2-line } \Rightarrow \text{proj2}$
where
\[
\text{perp-up } a \ l \equiv \\
\begin{cases} \\
\text{if proj2-incident } a \ l \text{ then } \epsilon \ p. \ p \in S \land \text{proj2-incident } p \ (\text{drop-perp } a \ l) \\
\text{else endpoint-in-S (perp-foot } a \ l) \\
\end{cases}
\]

lemma perp-up-degenerate-in-S-incident:
assumes $a \in \text{hyp2}$ and $\text{proj2-incident } a \ l$
shows $\text{perp-up } a \ l \in S$ (is ?p \in S)
and $\text{proj2-incident (perp-up } a \ l) \ (\text{drop-perp } a \ l)$
proof –
from $\text{proj2-incident } a \ l$
have $?p = (\epsilon \ p. \ p \in S \land \text{proj2-incident } p \ (\text{drop-perp } a \ l))$
by (unfold perp-up-def simp)
from $b \in \text{hyp2}$ and $\text{drop-perp-incident [of } a \ l]$ have $?p \in S \land \text{proj2-incident } ?p \ (\text{drop-perp } a \ l)$
by (rule line-through-K2-intersect-S)
hence unfolding $?p \equiv (\epsilon \ p. \ p \in S \land \text{proj2-incident } p \ (\text{drop-perp } a \ l))$
by (rule someI-ex)
thus $?p \in S$ and $\text{proj2-incident } ?p \ (\text{drop-perp } a \ l)$ by simp-all
qed

lemma perp-up-non-degenerate-in-S-at-end:
assumes $a \in \text{hyp2}$ and $b \in \text{hyp2}$ and $\text{proj2-incident } b \ l$
and $\sim \text{proj2-incident } a \ l$
shows $\text{perp-up } a \ l \in S$
and $B_R \ (\text{cart2-pt (perp-foot } a \ l)) \ (\text{cart2-pt } a) \ (\text{cart2-pt (perp-up } a \ l))$
proof –
from $\sim \text{proj2-incident } a \ l$
have $\text{perp-up } a \ l = \text{endpoint-in-S (perp-foot } a \ l) \ a$
by (unfold perp-up-def simp)
from $b \in \text{hyp2}$ and $\text{proj2-incident } b \ l$ and $a \in \text{hyp2}$
have $\text{perp-foot } a \ l \in \text{hyp2}$ by (rule perp-foot-hyp2)
with $a \in \text{hyp2}$
show $\text{perp-up } a \ l \in S$
and $B_R \ (\text{cart2-pt (perp-foot } a \ l)) \ (\text{cart2-pt } a) \ (\text{cart2-pt (perp-up } a \ l))$
unfolding $\text{perp-up } a \ l = \text{endpoint-in-S (perp-foot } a \ l) \ a$

qed
by (simp-all add: endpoint-in-S)
qed

lemma perp-up-in-S:
  assumes a ∈ hyp2 and b ∈ hyp2 and proj2-incident b l
  shows perp-up a l ∈ S
proof cases
  assume proj2-incident a l
  with ⟨a ∈ hyp2⟩
  show perp-up a l ∈ S by (rule perp-up-degenerate-in-S-incident)
next
  assume ¬ proj2-incident a l
  with assms
  show perp-up a l ∈ S by (rule perp-up-non-degenerate-in-S-at-end)
qed

lemma perp-up-incident:
  assumes a ∈ hyp2 and b ∈ hyp2 and proj2-incident b l
  shows proj2-incident (perp-up a l) (drop-perp a l)
  (is proj2-incident ?p ?m)
proof cases
  assume proj2-incident a l
  with ⟨a ∈ hyp2⟩
  show proj2-incident ?p ?m by (rule perp-up-degenerate-in-S-incident)
next
  assume ¬ proj2-incident a l
  hence ?p = endpoint-in-S (perp-foot a l) a (is ?p = endpoint-in-S ?c a)
    by (unfold perp-up-def) simp
  from perp-foot-incident [of a l] and ⟨¬ proj2-incident a l⟩
  have ?c ≠ a by auto
  from ⟨b ∈ hyp2⟩ and ⟨proj2-incident b l⟩ and ⟨a ∈ hyp2⟩
  have ?c ∈ hyp2 by (rule perp-foot-hyp2)
  with ⟨?c ≠ a⟩ and ⟨a ∈ hyp2⟩ and drop-perp-incident [of a l]
    and perp-foot-incident [of a l]
  show proj2-incident ?p ?m
    by (unfold l (?p = endpoint-in-S ?c a)) (simp add: endpoint-in-S-incident)
qed

lemma drop-perp-same-line-pole-in-S:
  assumes drop-perp p l = l
  shows pole l ∈ S
proof
  from ⟨drop-perp p l = l⟩
  have l = proj2-line-through p (pole l) by (unfold drop-perp-def) simp
  with proj2-line-through-incident [of pole l p]
  have proj2-incident (pole l) l by simp
  hence proj2-incident (pole l) (polar (pole l)) by (subst polar-pole)
thus \( \text{pole } l \in S \) by (unfold incident-own-polar-in-S)

\[ \text{qed} \]

**Lemma** hyp2-drop-perp-not-same-line:

assumes \( a \in \text{hyp2} \)

shows \( \text{drop-perp } a \ l \neq l \)

**Proof**

assume \( \text{drop-perp } a \ l = l \)

hence \( \text{pole } l \in S \) by (rule drop-perp-same-line-pole-in-S)

with \( \langle a \in \text{hyp2} \rangle \)

have \( \neg \text{proj2-incident } a \ (\text{polar } (\text{pole } l)) \)

by (simp add: tangent-not-through-K2)

with \( \langle \text{drop-perp } a \ l = l \rangle \)

have \( \neg \text{proj2-incident } a \ (\text{drop-perp } a \ l) \) by (simp add: polar-pole)

with \( \text{drop-perp-incident } [\text{of } a \ l] \) show False by simp

\[ \text{qed} \]

**Lemma** hyp2-incident-perp-foot-same-point:

assumes \( a \in \text{hyp2} \) and \( \text{proj2-incident } a \ l \)

shows \( \text{perp-foot } a \ l = a \)

**Proof**

from \( \langle a \in \text{hyp2} \rangle \)

have \( \text{drop-perp } a \ l \neq l \) by (rule hyp2-drop-perp-not-same-line)

with \( \text{perp-foot-incident } [\text{of } a \ l] \) and \( \langle \text{proj2-incident } a \ l \rangle \)

and \( \text{drop-perp-incident } [\text{of } a \ l] \) and \( \text{proj2-incident-unique} \)

show \( \text{perp-foot } a \ l = a \) by fast

\[ \text{qed} \]

**Lemma** perp-up-at-end:

assumes \( a \in \text{hyp2} \) and \( b \in \text{hyp2} \) and \( \text{proj2-incident } b \ l \)

shows \( B^R_\text{R} \ [\text{cart2-pt } (\text{perp-foot } a \ l)] (\text{cart2-pt } a) (\text{cart2-pt } (\text{perp-up } a \ l)) \)

**Proof**

cases

assume \( \text{proj2-incident } a \ l \)

with \( \langle a \in \text{hyp2} \rangle \)

have \( \text{perp-foot } a \ l = a \) by (rule hyp2-incident-perp-foot-same-point)

thus \( B^R_\text{R} \ [\text{cart2-pt } (\text{perp-foot } a \ l)] (\text{cart2-pt } a) (\text{cart2-pt } (\text{perp-up } a \ l)) \)

by (simp add: real-euclid.th3-1 real-euclid.th3-2)

next

assume \( \neg \text{proj2-incident } a \ l \)

with \( \text{assms} \)

show \( B^R_\text{R} \ [\text{cart2-pt } (\text{perp-foot } a \ l)] (\text{cart2-pt } a) (\text{cart2-pt } (\text{perp-up } a \ l)) \)

by (rule perp-up-non-degenerate-in-S-at-end)

\[ \text{qed} \]

**Definition** perp-down :: \( \text{proj2 } \Rightarrow \text{proj2-line } \Rightarrow \text{proj2} \) where

perp-down \( a \ l \triangleq \text{endpoint-in-S } \ (\text{perp-up } a \ l) \ a \)

**Lemma** perp-down-in-S:

assumes \( a \in \text{hyp2} \) and \( b \in \text{hyp2} \) and \( \text{proj2-incident } b \ l \)
shows \( \text{perp-down a l} \in S \)

proof –
from \( \text{assms} \) have \( \text{perp-up a l} \in S \) by (rule \( \text{perp-up-in-S} \))
with \( \langle a \in \text{hyp2} \rangle \)
show \( \text{perp-down a l} \in S \) by (unfold \( \text{perp-down-def} \)) (simp add: \( \text{endpoint-in-S} \))
qed

lemma \( \text{perp-down-incident} \):
assumes \( a \in \text{hyp2} \) and \( b \in \text{hyp2} \) and \( \text{proj2-incident b l} \)
shows \( \text{proj2-incident (perp-down a l) (drop-perp a l)} \)
proof –
from \( \text{assms} \) have \( \text{perp-up a l} \in S \) by (rule \( \text{perp-up-in-S} \))
with \( \langle a \in \text{hyp2} \rangle \) have \( \text{perp-up a l} \neq a \) by (rule \( \text{hyp2-S-not-equal [symmetric]} \))
from \( \text{assms} \)
have \( \text{proj2-incident (perp-up a l) (drop-perp a l)} \) by (rule \( \text{perp-up-incident} \))
with \( \langle \text{perp-up a l} \neq a \rangle \) and \( \langle \text{perp-up a l} \in S \rangle \) and \( \langle a \in \text{hyp2} \rangle \)
and \( \text{drop-perp-incident [of a l]} \)
show \( \text{proj2-incident (perp-down a l) (drop-perp a l)} \)
by (unfold \( \text{perp-down-def} \)) (simp add: \( \text{endpoint-in-S-incident} \))
qed

lemma \( \text{perp-up-down-distinct} \):
assumes \( a \in \text{hyp2} \) and \( b \in \text{hyp2} \) and \( \text{proj2-incident b l} \)
shows \( \text{perp-up a l} \neq \text{perp-down a l} \)
proof –
from \( \text{assms} \) have \( \text{perp-up a l} \in S \) by (rule \( \text{perp-up-in-S} \))
with \( \langle a \in \text{hyp2} \rangle \) have \( \text{perp-up a l} \neq \text{perp-down a l} \)
unfolding \( \text{perp-down-def} \)
by (simp add: \( \text{endpoint-in-S-S-strict-hyp2-distinct [symmetric]} \))
qed

lemma \( \text{perp-up-down-foot-are-endpoints-in-S} \):
assumes \( a \in \text{hyp2} \) and \( b \in \text{hyp2} \) and \( \text{proj2-incident b l} \)
shows \( \text{are-endpoints-in-S (perp-up a l) (perp-down a l) (perp-foot a l)} \)
proof –
from \( \langle b \in \text{hyp2} \rangle \) and \( \langle \text{proj2-incident b l} \rangle \) and \( \langle a \in \text{hyp2} \rangle \)
have \( \text{perp-foot a l} \in \text{hyp2} \) by (rule \( \text{perp-foot-hyp2} \))
from \( \text{assms} \) have \( \text{perp-up a l} \in S \) by (rule \( \text{perp-up-in-S} \))
from \( \text{assms} \)
have \( \text{proj2-incident (perp-up a l) (drop-perp a l)} \) by (rule \( \text{perp-up-incident} \))
with \( \langle \text{perp-foot a l} \in \text{hyp2} \rangle \) and \( \langle a \in \text{hyp2} \rangle \) and \( \langle \text{perp-up a l} \in S \rangle \)
and \( \text{perp-foot-incident (2) [of a l]} \) and \( \text{drop-perp-incident [of a l]} \)
show \( \text{are-endpoints-in-S (perp-up a l) (perp-down a l) (perp-foot a l)} \)
by (unfold \( \text{perp-down-def} \)) (rule \( \text{end-and-opposite-are-endpoints-in-S} \))
qed
lemma perp-foot-opposite-endpoint-in-S:
assumes $a \in \text{hyp2}$ and $b \in \text{hyp2}$ and $c \in \text{hyp2}$ and $a \neq b$
shows $\text{endpoint-in-S} \ (\text{endpoint-in-S} \ a \ b) \ (\text{perp-foot} \ c \ (\text{proj2-line-through} \ a \ b))$
= $\text{endpoint-in-S} \ b \ a$
(is $\text{endpoint-in-S} \ ?p \ ?d = \text{endpoint-in-S} \ b \ a$)
proof –
let $?q = \text{endpoint-in-S} \ ?p \ ?d$
from $\langle a \in \text{hyp2} \rangle$ and $\langle b \in \text{hyp2} \rangle$ have $?p \in S$ by (simp add: endpoint-in-S)
let $?l = \text{proj2-line-through} \ a \ b$
have $\text{proj2-incident} \ a \ ?l$ and $\text{proj2-incident} \ b \ ?l$
by (rule proj2-line-through-incident)+
with $\langle a \neq b \rangle$ and $\langle a \in \text{hyp2} \rangle$ and $\langle b \in \text{hyp2} \rangle$
have $\text{proj2-incident} \ ?p \ ?l$
by (simp-all add: endpoint-in-S-incident)
from $\langle a \in \text{hyp2} \rangle$ and $\langle \text{proj2-incident} \ a \ ?l \rangle$ and $\langle c \in \text{hyp2} \rangle$
have $?d \in \text{hyp2}$ by (rule perp-foot-hyp2)
with $\langle ?p \in S \rangle$ have $?q \neq ?p$ by (rule endpoint-in-S-S-strict-hyp2-distinct)
from $\langle ?p \in S \rangle$ and $\langle ?d \in \text{hyp2} \rangle$ have $?q \in S$ by (simp add: endpoint-in-S)
from $\langle ?d \in \text{hyp2} \rangle$ and $\langle ?p \in S \rangle$
have $?p \neq ?d$ by (rule hyp2-S-not-equal [symmetric])
with $\langle ?p \in S \rangle$ and $\langle ?d \in \text{hyp2} \rangle$ and $\langle \text{proj2-incident} \ ?p \ ?l \rangle$
and $\text{perp-foot-incident} \ (\.1) \ [\text{of} \ c \ ?l]$
have $\text{proj2-incident} \ ?q \ ?l$ by (simp add: endpoint-in-S-incident)
with $\langle a \neq b \rangle$ and $\langle a \in \text{hyp2} \rangle$ and $\langle b \in \text{hyp2} \rangle$ and $\langle ?q \in S \rangle$
and $\langle \text{proj2-incident} \ a \ ?l \rangle$ and $\langle \text{proj2-incident} \ b \ ?l \rangle$
have $?q = ?p \lor ?q = \text{endpoint-in-S} \ b \ a$
by (simp add: endpoints-in-S-incident-unique)
with $\langle ?q \neq ?p \rangle$ show $?q = \text{endpoint-in-S} \ b \ a$ by simp
qed

lemma endpoints-in-S-perp-foot-are-endpoints-in-S:
assumes $a \in \text{hyp2}$ and $b \in \text{hyp2}$ and $c \in \text{hyp2}$ and $a \neq b$
and $\text{proj2-incident} \ a \ l$ and $\text{proj2-incident} \ b \ l$
shows $\text{are-endpoints-in-S} \ (\text{endpoint-in-S} \ a \ b) \ (\text{endpoint-in-S} \ b \ a) \ a \ (\text{perp-foot} \ c \ l)$
proof –
define $p \ q \ d$
where $p = \text{endpoint-in-S} \ a \ b$
and $q = \text{endpoint-in-S} \ b \ a$
and $d = \text{perp-foot} \ c \ l$
from $\langle a \neq b \rangle$ and $\langle a \in \text{hyp2} \rangle$ and $\langle b \in \text{hyp2} \rangle$
have \( p \neq q \) by \((\text{unfold } p\text{-def } q\text{-def})\) \((\text{simp add: endpoint-in-S-swap})\) 

from \( \langle a \in \text{hyp2} \rangle \) and \( \langle b \in \text{hyp2} \rangle \) 
have \( p \in S \) and \( q \in S \) by \((\text{unfold } p\text{-def } q\text{-def})\) \((\text{simp-all add: endpoint-in-S})\) 

from \( \langle a \in \text{hyp2} \rangle \) and \( \langle \text{proj2-incident } a \ b \rangle \) and \( \langle c \in \text{hyp2} \rangle \) 
have \( d \in \text{hyp2} \) by \((\text{unfold } d\text{-def})\) \((\text{rule perp-foot-hyp2})\) 

from \( \langle a \neq b \rangle \) and \( \langle a \in \text{hyp2} \rangle \) and \( \langle b \in \text{hyp2} \rangle \) and \( \langle \text{proj2-incident } a \ b \rangle \) 
and \( \langle \text{proj2-incident } a \ b \rangle \) 

have \( \text{proj2-incident } p \ l \ \text{and} \ \text{proj2-incident } q \ l \) 
by \((\text{unfold } p\text{-def } q\text{-def})\) \((\text{simp-all add: endpoint-in-S-incident})\) 
with \( \langle \text{proj2-incident } a \ b \rangle \) and \( \text{perp-foot-incident}(1) \ [\text{of } c \ l] \) 

have \( \text{proj2-set-Col} \ \{p,q,a,d\} \) 
by \((\text{unfold } d\text{-def } \text{proj2-set-Col-def})\) \((\text{simp add: exI } [\text{of } - \ l])\) 
	with \( \langle p \neq q \rangle \) and \( \langle p \in S \rangle \) and \( \langle q \in S \rangle \) and \( \langle a \in \text{hyp2} \rangle \) and \( \langle d \in \text{hyp2} \rangle \) 

show \( \text{are-endpoints-in-S } p \ q \ a \ d \) 
by \((\text{unfold } \text{are-endpoints-in-S-def})\) \((\text{simp})\) 
qed

definition \( \text{right-angle} :: \text{proj2} \Rightarrow \text{proj2} \Rightarrow \text{proj2} \Rightarrow \text{bool} \) where 
\[ \text{right-angle } p \ a \ q \triangleq p \in S \land q \in S \land a \in \text{hyp2} \land M\text{-perp}(\text{proj2-line-through } p \ a)(\text{proj2-line-through } a \ q) \] 

lemma \( \text{perp-foot-up-right-angle} \): 
assumes \( p \in S \) and \( a \in \text{hyp2} \) and \( b \in \text{hyp2} \) and \( \text{proj2-incident } p \ l \) 
and \( \text{proj2-incident } b \ l \) 
shows \( \text{right-angle } p \ (\text{perp-foot } a \ l)(\text{perp-up } a \ l) \) 
proof – 

define \( c \) where \( c = \text{perp-foot } a \ l \) 
define \( q \) where \( q = \text{perp-up } a \ l \) 
from \( \langle a \in \text{hyp2} \rangle \) and \( \langle b \in \text{hyp2} \rangle \) and \( \langle \text{proj2-incident } b \ l \rangle \) 
have \( q \in S \) by \((\text{unfold } q\text{-def})\) \((\text{rule perp-foot-in-S})\) 
from \( \langle b \in \text{hyp2} \rangle \) and \( \langle \text{proj2-incident } b \ l \rangle \) and \( \langle a \in \text{hyp2} \rangle \) 
have \( c \in \text{hyp2} \) by \((\text{unfold } c\text{-def})\) \((\text{rule perp-foot-hyp2})\) 
with \( \langle p \in S \rangle \) and \( \langle q \in S \rangle \) have \( c \neq p \) and \( c \neq q \) 
by \((\text{simp-all add: hyp2-S-not-equal})\) 
	from \( \langle c \neq p \rangle \) [\text{symmetric}] and \( \langle \text{proj2-incident } p \ l \rangle \) 
and \( \text{perp-foot-incident}(1) \ [\text{of } a \ l] \) 
have \( l = \text{proj2-line-through } p \ c \) 
by \((\text{unfold } c\text{-def})\) \((\text{rule proj2-line-through-unique})\) 

define \( m \) where \( m = \text{drop-perp } a \ l \) 
from \( \langle a \in \text{hyp2} \rangle \) and \( \langle b \in \text{hyp2} \rangle \) and \( \langle \text{proj2-incident } b \ l \rangle \) 
have \( \text{proj2-incident } q \ m \) by \((\text{unfold } q\text{-def } m\text{-def})\) \((\text{rule perp-up-incident})\) 
with \( \langle c \neq q \rangle \) and \( \text{perp-foot-incident}(2) \ [\text{of } a \ l] \) 
have \( m = \text{proj2-line-through } c \ q \)
by \((\text{unfold } c\text{-def } m\text{-def})\) \((\text{rule } \text{proj2-line-through-unique})\)

with \((p \in S)\) and \((q \in S)\) and \((c \in \text{hyp2})\) and \(\text{drop-perp-perp } [\text{of } l a]\) and \(l = \text{proj2-line-through } p c\)

show \(\text{right-angle } p \ (\text{perp-foot } a \ l) \ (\text{perp-up } a \ l)\)

by \((\text{unfold } \text{right-angle-def } q\text{-def } c\text{-def } m\text{-def})\) \(\text{simp}\)

\[\text{qed}\]

\textbf{lemma } M\text{-perp-unique:}

\begin{itemize}
\item \textbf{assumes} \(a \in \text{hyp2}\) and \(b \in \text{hyp2}\) and \(\text{proj2-incident } a \ l\)
\item and \(\text{proj2-incident } b \ m\) and \(\text{proj2-incident } b \ n\) and \(M\text{-perp } l m\) and \(\text{proj2-incident } b \ n\)
\item shows \(m = n\)
\end{itemize}

\textbf{proof –}

\begin{itemize}
\item from \((a \in \text{hyp2})\) and \((\text{proj2-incident } a \ l)\)
\item have \(\text{pole } l \notin \text{hyp2}\) by \((\text{rule } \text{line-through-hyp2-pole-not-in-hyp2})\)
\item with \((b \in \text{hyp2})\) have \(b \neq \text{pole } l\) by \(\text{auto}\)
\item with \((\text{proj2-incident } b \ m)\) and \((M\text{-perp } l m)\) and \((\text{proj2-incident } b \ n)\) and \((M\text{-perp } l n)\) and \(\text{proj2-incident-unique}\)
\item show \(m = n\) by \((\text{unfold } M\text{-perp-def})\) \(\text{auto}\)
\end{itemize}

\[\text{qed}\]

\textbf{lemma } \text{perp-foot-eq-implies-drop-perp-eq:}

\begin{itemize}
\item \textbf{assumes} \(a \in \text{hyp2}\) and \(b \in \text{hyp2}\) and \(\text{proj2-incident } a \ l\)
\item and \(\text{perp-foot } b \ l = \text{perp-foot } c \ l\)
\item shows \(\text{drop-perp } b \ l = \text{drop-perp } c \ l\)
\end{itemize}

\textbf{proof –}

\begin{itemize}
\item from \((a \in \text{hyp2})\) and \((\text{proj2-incident } a \ l)\) and \((b \in \text{hyp2})\)
\item have \(\text{perp-foot } b \ l \in \text{hyp2}\) by \((\text{rule } \text{perp-foot-hyp2})\)
\item from \((\text{perp-foot } b \ l = \text{perp-foot } c \ l)\)
\item have \(\text{proj2-incident } (\text{perp-foot } b \ l) \ (\text{drop-perp } c \ l)\)
\item by \((\text{simp add: } \text{perp-foot-incident})\)
\item with \((a \in \text{hyp2})\) and \((\text{perp-foot } b \ l \in \text{hyp2})\) and \((\text{proj2-incident } a \ l)\)
\item and \(\text{perp-foot-incident } (\text{ proj2-incident } a \ l)\)
\item and \(\text{drop-perp-perp } [\text{of } l]\) and \(\text{of } l\)
\item show \(\text{drop-perp } b \ l = \text{drop-perp } c \ l\) by \((\text{simp add: } M\text{-perp-unique})\)
\end{itemize}

\[\text{qed}\]

\textbf{lemma } \text{right-angle-to-compass:}

\begin{itemize}
\item \textbf{assumes} \(\exists J, \text{is-K2-isometry } J \land \text{apply-cltn2 } p J = \text{east}\)
\item and \(\text{apply-cltn2 } a J = \text{K2-centre} \land \text{apply-cltn2 } q J = \text{north}\)
\end{itemize}

\textbf{proof –}

\begin{itemize}
\item from \((\text{right-angle } p a q)\)
\item have \(p \in S\) and \(q \in S\) and \(a \in \text{hyp2}\)
\item and \(M\text{-perp } (\text{proj2-line-through } p a) \ (\text{proj2-line-through } a q)\)
\item (is \(M\text{-perp } ?l ?m)\)
\item by \((\text{unfold } \text{right-angle-def})\) \(\text{simp-all}\)
\end{itemize}

\begin{itemize}
\item have \(\text{proj2-incident } p ?l\) and \(\text{proj2-incident } a ?l\)
\end{itemize}
and proj2-incident q ?m and proj2-incident a ?m by (rule proj2-line-through-incident)+

from (M-perp ?l ?m) and (a ∈ hyp2) and (proj2-incident a ?l) and (proj2-incident a ?m) and M-perp-to-compass [of ?l ?m a a] obtain J''i where is-K2-isometry J''i and apply-cltn2-line equator J''i = ?l and apply-cltn2-line meridian J''i = ?m by auto

let J'' = cltn2-inverse J''i

from (apply-cltn2-line equator J''i = ?l) and (apply-cltn2-line meridian J''i = ?m) and (proj2-incident p ?l) and (proj2-incident a ?l) and (proj2-incident q ?m) and (proj2-incident a ?m) have proj2-incident (apply-cltn2 a ?J'') equator and proj2-incident (apply-cltn2 a ?J'') equator and proj2-incident (apply-cltn2 q ?J'') meridian and proj2-incident (apply-cltn2 a ?J'') meridian by (simp-all add: apply-cltn2-incident [symmetric])

from (proj2-incident (apply-cltn2 a ?J'') equator) and (proj2-incident (apply-cltn2 a ?J'') meridian) have apply-cltn2 a ?J'' = K2-centre by (rule on-equator-meridian-is-K2-centre)

from (is-K2-isometry J''i) have is-K2-isometry ?J'' by (rule cltn2-inverse-is-K2-isometry)

with (p ∈ S) and (q ∈ S) have apply-cltn2 p ?J'' ∈ S and apply-cltn2 q ?J'' ∈ S by (unfold is-K2-isometry-def) simp-all with east-west-distinct and north-south-distinct and compass-in-S and east-west-on-equator and north-south-far-north-on-meridian and (proj2-incident (apply-cltn2 p ?J'') equator) and (proj2-incident (apply-cltn2 q ?J'') meridian) have apply-cltn2 p ?J'' = east ∨ apply-cltn2 p ?J'' = west and apply-cltn2 q ?J'' = north ∨ apply-cltn2 q ?J'' = south by (simp-all add: line-S-two-intersections-only)

have ∃ J'. is-K2-isometry J' ∧ apply-cltn2 p J' = east ∧ apply-cltn2 a J' = K2-centre ∧ (apply-cltn2 q J' = north ∨ apply-cltn2 q J' = south)

proof cases
assume apply-cltn2 p ?J'' = east with (is-K2-isometry ?J'') and (apply-cltn2 a ?J'' = K2-centre) and (apply-cltn2 q ?J'' = north ∨ apply-cltn2 q ?J'' = south) show ∃ J'. is-K2-isometry J' ∧ apply-cltn2 p J' = east ∧ apply-cltn2 a J' = K2-centre ∧ (apply-cltn2 q J' = north ∨ apply-cltn2 q J' = south)
by (simp add: exI [of - ?J'])

next
assumption apply-cltn2 p ?J'' ≠ east
with (apply-cltn2 p ?J'' = east ∨ apply-cltn2 p ?J'' = west)

have apply-cltn2 p ?J'' = west by simp

let ?J' = cltn2-compose ?J'' meridian-reflect
from is-K2-isometry ?J'' and meridian-reflect-K2-isometry
have is-K2-isometry ?J' by (rule cltn2-compose-is-K2-isometry)

moreover
from (apply-cltn2 p ?J'' = east) and (apply-cltn2 a ?J'' = K2-centre)
and (apply-cltn2 q ?J'' = north ∨ apply-cltn2 q ?J'' = south)
and compass-reflect-compass
have apply-cltn2 a ?J' = K2-centre
and apply-cltn2 p ?J' = east
and apply-cltn2 q ?J' = north
by (auto simp add: cltn2.act-act [simplified, symmetric])

ultimately

show ∃ J'. is-K2-isometry J' ∧ apply-cltn2 p J' = east
∧ apply-cltn2 a J' = K2-centre
∧ (apply-cltn2 q J' = north ∨ apply-cltn2 q J' = south)
by (simp add: exI [of - ?J'])

qed

then obtain J' where is-K2-isometry J' and apply-cltn2 p J' = east
and apply-cltn2 a J' = K2-centre
and apply-cltn2 q J' = north ∨ apply-cltn2 q J' = south
by auto

show ∃ J. is-K2-isometry J ∧ apply-cltn2 p J = east
∧ apply-cltn2 a J = K2-centre ∧ apply-cltn2 q J = north

proof cases
assumption apply-cltn2 q J' = north
with is-K2-isometry J' and (apply-cltn2 p J' = east)
and (apply-cltn2 a J' = K2-centre)

show ∃ J. is-K2-isometry J ∧ apply-cltn2 p J = east
∧ apply-cltn2 a J = K2-centre ∧ apply-cltn2 q J = north
by (simp add: exI [of - J'])

next
assumption apply-cltn2 q J' ≠ north
with (apply-cltn2 q J' = north ∨ apply-cltn2 q J' = south)

have apply-cltn2 q J' = south by simp

let ?J = cltn2-compose J' equator-reflect
from is-K2-isometry J' and equator-reflect-K2-isometry
have is-K2-isometry ?J by (rule cltn2-compose-is-K2-isometry)

moreover
from (apply-cltn2 p J' = east) and (apply-cltn2 a J' = K2-centre)
and (apply-cltn2 q J' = south) and compass-reflect-compass
have apply-cltn2 p ?J = east
and apply-cltn2 a ?J = K2-centre
and apply-cltn2 q ?J = north

188
ultimately
show ∃ J. is-K2-isometry J ∧ apply-cltn2 p J = east ∧ apply-cltn2 a J = K2-centre ∧ apply-cltn2 q J = north
by (simp add: exI [of - ?J])
qed

lemma right-angle-to-right-angle:
assumes right-angle p a q and right-angle r b s
shows ∃ J. is-K2-isometry J ∧ apply-cltn2 p J = r ∧ apply-cltn2 a J = b ∧ apply-cltn2 q J = s
proof –
from ⟨right-angle p a q ⟩ and right-angle-to-compass [of p a q]
obtain H where is-K2-isometry H and apply-cltn2 p H = east
and apply-cltn2 a H = K2-centre and apply-cltn2 q H = north
by auto
from ⟨right-angle r b s ⟩ and right-angle-to-compass [of r b s]
obtain K where is-K2-isometry K and apply-cltn2 r K = east
and apply-cltn2 b K = K2-centre and apply-cltn2 s K = north
by auto
let ?Ki = cltn2-inverse K
let ?J = cltn2-compose H ?Ki
from ⟨is-K2-isometry H⟩ and ⟨is-K2-isometry K⟩
have is-K2-isometry ?J
by (simp add: cltn2-inverse-is-K2-isometry cltn2-compose-is-K2-isometry)
from ⟨apply-cltn2 r K = east⟩ and ⟨apply-cltn2 b K = K2-centre⟩
and ⟨apply-cltn2 s K = north⟩
have apply-cltn2 east ?Ki = r and apply-cltn2 K2-centre ?Ki = b
and apply-cltn2 north ?Ki = s
by (simp-all add: cltn2.act-inv-iff [simplified])
with ⟨apply-cltn2 p H = east⟩ and ⟨apply-cltn2 a H = K2-centre⟩
and ⟨apply-cltn2 q H = north⟩
have apply-cltn2 p ?J = r and apply-cltn2 a ?J = b
and apply-cltn2 q ?J = s
by (simp-all add: cltn2.act-act [simplified,symmetric])
with ⟨is-K2-isometry ?J⟩
show ∃ J. is-K2-isometry J
∧ apply-cltn2 p J = r ∧ apply-cltn2 a J = b ∧ apply-cltn2 q J = s
by (simp add: exI [of - ?J])
qed

8.11 Functions of distance

definition exp-2dist :: proj2 ⇒ proj2 ⇒ real where
exp-2dist a b
\( \triangleq \) if \( a = b \)

then 1

else cross-ratio \((\text{endpoint-in-S} \ a \ b) \ (\text{endpoint-in-S} \ b \ a)\) \(a \ b\)

**definition** \(\text{cosh-dist} :: \text{proj2} \Rightarrow \text{proj2} \Rightarrow \text{real} \)**

\(\text{cosh-dist} \ a \ b \triangleq (\sqrt{\exp-2\text{dist} \ a \ b} + \sqrt{1 / (\exp-2\text{dist} \ a \ b)}) / 2\)

**lemma** \(\exp-2\text{dist}-\text{formula}::\)**

assumes \(a \neq 0 \) and \(b \neq 0 \) and \(\text{proj2-abs} \ a \in \text{hyp2} \) (is \(?pa \in \text{hyp2}\))

and \(\text{proj2-abs} \ b \in \text{hyp2} \) (is \(?pb \in \text{hyp2}\))

shows \(\exp-2\text{dist} \ (\text{proj2-abs} \ a) \ (\text{proj2-abs} \ b)\)

\[\triangleq (a \cdot (M \ast v \ b) + \sqrt{\text{quarter-discrim} \ a \ b}) / (a \cdot (M \ast v \ b) - \sqrt{\text{quarter-discrim} \ a \ b})\]

\(\lor\)

\(\exp-2\text{dist} \ (\text{proj2-abs} \ a) \ (\text{proj2-abs} \ b)\)

\[\triangleq (a \cdot (M \ast v \ b) - \sqrt{\text{quarter-discrim} \ a \ b}) / (a \cdot (M \ast v \ b) + \sqrt{\text{quarter-discrim} \ a \ b})\]

(is \(?e2d = (\text{?aMb + ?sqd}) / (\text{?aMb - ?sqd})\)

\(\lor\)

\(?e2d = (\text{?aMb - ?sqd}) / (\text{?aMb + ?sqd})\)

**proof**

**cases**

**assume** \(?pa = \?pb\)

**hence** \(?e2d = 1\) by (unfold \(\exp-2\text{dist}-\text{def}\), simp)

from \(?pa = \?pb\)

have \(\text{quarter-discrim} \ a \ b = 0\) by (rule \(\text{quarter-discrim-self-zero}\))

**hence** \(?sqd = 0\) by simp

from \(\text{proj2-abs} \ a = \text{proj2-abs} \ b\) and \((b \neq 0)\) and \(\text{proj2-abs-abs-mult}\)

obtain \(k\) where \(a = k \ast_R b\) by auto

from \((b \neq 0)\) and \(\text{proj2-abs} \ b \in \text{hyp2}\)

have \(b \cdot (M \ast v \ b) < 0\) by (subst \(\text{K2-abs [symmetric]}\))

with \((a \neq 0)\) and \((a = k \ast_R b)\) have \(\text{?aMb \neq 0}\) by simp

with \(?e2d = 1\) and \(?sqd = 0\)

**show** \(?e2d = (\text{?aMb + ?sqd}) / (\text{?aMb - ?sqd})\)

\(\lor\)

\(?e2d = (\text{?aMb - ?sqd}) / (\text{?aMb + ?sqd})\)

by simp

next

**assume** \(?pa \neq \?pb\)

let \(\?l = \text{proj2-line-through} \ ?pa \ ?pb\)

**have** \(\text{proj2-incident} \ ?pa \ ?l \ and \ \text{proj2-incident} \ ?pb \ ?l\)

by (rule \(\text{proj2-line-through-incident}\))

with \((a \neq 0)\) and \((b \neq 0)\) and \(?pa \neq \?pb\)

**have** \(\text{proj2-incident} \ (\text{S-intersection1} \ a \ b) \ ?l\) (is \(\text{proj2-incident} \ ?Si1 \ ?l\))

and \(\text{proj2-incident} \ (\text{S-intersection2} \ a \ b) \ ?l\) (is \(\text{proj2-incident} \ ?Si2 \ ?l\))

by (rule \(\text{S-intersections-incident}\))

with \(\text{proj2-incident} \ ?pa \ ?l\) and \(\text{proj2-incident} \ ?pb \ ?l\)

**have** \(\text{proj2-set-Col} \ \{\text{?pa}, \text{?pb}, \text{?Si1}, \text{?Si2}\\} \) by (unfold \(\text{proj2-set-Col-def}\), auto)

**have** \(\{\text{?pa, ?pb, ?Si2, ?Si1}\} = \{\text{?pa, ?pb, ?Si1, ?Si2}\}\) by auto
from \(a \neq 0\) and \(b \neq 0\) and \(?pa \neq ?pb\) and \(?pa \in \text{hyp2}\)

have \(?Si1 \in S\) and \(?Si2 \in S\)

by (simp-all add: S-intersections-in-S)

with \(?pa \in \text{hyp2}\) and \(?pb \in \text{hyp2}\)

have \(?Si1 \neq ?pa\) and \(?Si2 \neq ?pa\) and \(?Si1 \neq ?pb\) and \(?Si2 \neq ?pb\)

by (simp-all add: hyp2-S-not-equal [symmetric])

with \(?\langle ?pa, ?pb, ?Si1, ?Si2 \rangle\) and \(?\langle ?pa \neq ?pb \rangle\)

have cross-ratio-correct \(?pa \ ?pb \ ?Si1 \ ?Si2\)

and cross-ratio-correct \(?pa \ ?pb \ ?Si2 \ ?Si1\)

unfolding cross-ratio-correct-def


from \(a \neq 0\) and \(b \neq 0\) and \(?pa \neq ?pb\) and \(?pa \in \text{hyp2}\)

have \(?Si1 \neq ?Si2\) by (simp add: S-intersections-distinct)

with (cross-ratio-correct \(?pa \ ?pb \ ?Si1 \ ?Si2\))

and (cross-ratio-correct \(?pa \ ?pb \ ?Si2 \ ?Si1\))

have cross-ratio \(?Si1 \ ?Si2 \ ?pa \ ?pb = \text{cross-ratio} \ ?pa \ ?pb \ ?Si1 \ ?Si2\)

and cross-ratio \(?Si2 \ ?Si1 \ ?pa \ ?pb = \text{cross-ratio} \ ?pa \ ?pb \ ?Si2 \ ?Si1\)

by (simp-all add: cross-ratio-swap-13-24)

from \(a \neq 0\) and (proj2-abs \(a \in \text{hyp2}\))

have \(a \cdot (M * v a) < 0\) by (subst K2-abs [symmetric])

with \(a \neq 0\) and \(b \neq 0\) and \(?pa \neq ?pb\) and cross-ratio-abs \([of \ a b 1 1]\)

have cross-ratio \(?pa \ ?pb \ ?Si1 \ ?Si2 = (\neg aMb - \ ?sqd) / (\neg aMb + \ ?sqd)\)

by (unfold S-intersections-defs S-intersection-coeffs-defs, simp)

with times-divide-times-eq \([of \ -1 -1 -aMb - \ ?sqd -aMb + \ ?sqd]\)

have cross-ratio \(?pa \ ?pb \ ?Si1 \ ?Si2 = (\neg aMb + \ ?sqd) / (\neg aMb - \ ?sqd)\) by (simp add: ac-simps)

with (cross-ratio \(?pa \ ?pb \ ?Si1 \ ?Si2 \ ?pa \ ?pb = \text{cross-ratio} \ ?pa \ ?pb \ ?Si1 \ ?Si2\))

have cross-ratio \(?Si1 \ ?Si2 \ ?pa \ ?pb = (\neg aMb + \ ?sqd) / (\neg aMb - \ ?sqd)\) by simp

from (cross-ratio \(?pa \ ?pb \ ?Si1 \ ?Si2 = (\neg aMb + \ ?sqd) / (\neg aMb - \ ?sqd)\))

and cross-ratio-swap-34 \([of \ ?pa \ ?pb \ ?Si2 \ ?Si1]\)

have cross-ratio \(?pa \ ?pb \ ?Si2 \ ?Si1 = (\neg aMb - \ ?sqd) / (\neg aMb + \ ?sqd)\) by simp

with (cross-ratio \(?Si2 \ ?Si1 \ ?pa \ ?pb = \text{cross-ratio} \ ?pa \ ?pb \ ?Si2 \ ?Si1\))

have cross-ratio \(?Si2 \ ?Si1 \ ?pa \ ?pb = (\neg aMb - \ ?sqd) / (\neg aMb + \ ?sqd)\) by simp

from \(a \neq 0\) and \(b \neq 0\) and \(?pa \neq ?pb\) and \(?pa \in \text{hyp2}\) and \(?pb \in \text{hyp2}\)

have \((\neg Si2 = \text{endpoint-in-S} \ ?pa \ ?pb \wedge \ ?Si2 = \text{endpoint-in-S} \ ?pb \ ?pa)\)

\(\lor (\neg Si2 = \text{endpoint-in-S} \ ?pa \ ?pb \wedge \ ?Si1 = \text{endpoint-in-S} \ ?pb \ ?pa)\)

by (simp add: S-intersections-endpoints-in-S)

with (cross-ratio \(?Si1 \ ?Si2 \ ?pa \ ?pb = (\neg aMb + \ ?sqd) / (\neg aMb - \ ?sqd)\))

and (cross-ratio \(?Si2 \ ?Si1 \ ?pa \ ?pb = (\neg aMb - \ ?sqd) / (\neg aMb + \ ?sqd)\))

and \(?\langle ?pa \neq ?pb \rangle\)

show \(?e2d = (\neg aMb + \ ?sqd) / (\neg aMb - \ ?sqd)\)

\(\lor \ e2d = (\neg aMb - \ ?sqd) / (\neg aMb + \ ?sqd)\)

by (unfold exp-2dist-def, auto)

qed
lemma cosh-dist-formula:

assumes \( a \neq 0 \) and \( b \neq 0 \) and \( \text{proj2-abs } a \in \text{hyp2} \) (is \( \text{pa } \in \text{hyp2} \))
and \( \text{proj2-abs } b \in \text{hyp2} \) (is \( \text{pb } \in \text{hyp2} \))

shows \( \text{cosh-dist } (\text{proj2-abs } a) (\text{proj2-abs } b) = |a \cdot (M \ast v b)| / \sqrt{a \cdot (M \ast v a) \ast (b \cdot (M \ast v b))} \)
(is cosh-dist \( \text{?pa } \text{?pb } = |\text{aMb} \ast / \sqrt{\text{aMa} \ast \text{?bMb}} |)

proof –

let \( \text{qd } = \text{quarter-discrim } a \)
let \( \text{sqd } = \sqrt{\text{qd}} \)
let \( \text{e2d } = \text{exp-2dist } \text{pa } \text{pb} \)

from assms

have \( \text{e2d } = (\text{aMb } + \text{sqd}) / (\text{aMb } - \text{sqd}) \)
  \( \vee \) \( \text{e2d } = (\text{aMb } - \text{sqd}) / (\text{aMb } + \text{sqd}) \)

by (rule exp-2dist-formula)

hence \( \text{cosh-dist } \text{pa } \text{pb } = (\sqrt{(\text{aMb } + \text{sqd}) / (\text{aMb } - \text{sqd})}) \)
\( + \sqrt{(\text{aMb } - \text{sqd}) / (\text{aMb } + \text{sqd})}) / 2 \)

by (unfold cosh-dist-def, auto)

have \( \text{qd } \geq 0 \)

proof cases

assume \( \text{pa } = \text{pb} \)

thus \( \text{qd } \geq 0 \) by (simp add: quarter-discrim-self-zero)

next

assume \( \text{pa } \neq \text{pb} \)

with \( \langle a \neq 0 \rangle \) and \( \langle b \neq 0 \rangle \) and \( \langle \text{pa } \in \text{hyp2} \rangle \)

have \( \text{qd } > 0 \) by (simp add: quarter-discrim-positive)

thus \( \text{qd } \geq 0 \) by simp

qed

with \( \text{real-sqrt-pow2 } [\text{of } \text{qd}] \) have \( \text{sqd}^2 = \text{qd} \) by simp

hence \( \text{aMb } + \text{sqd} \ast (\text{aMb } - \text{sqd}) = \text{aMa } \ast \text{bMb} \)

by (unfold quarter-discrim-def, simp add: algebra-simps power2-eq-square)

from times-divide-times-eq [of \( \text{aMb } + \text{sqd} \text{aMb } + \text{sqd} \text{aMb } - \text{sqd} \)]

have \( (\text{aMb } + \text{sqd}) \ast (\text{aMb } - \text{sqd}) \)
  \( = (\text{aMb } + \text{sqd})^2 / (\text{aMb } + \text{sqd}) \ast (\text{aMb } - \text{sqd}) \)

by (simp add: power2-eq-square)

with \( \text{aMb } + \text{sqd} \ast (\text{aMb } - \text{sqd}) = \text{aMa } \ast \text{bMb} \)

have \( (\text{aMb } + \text{sqd}) / (\text{aMb } - \text{sqd}) = (\text{aMb } + \text{sqd})^2 / (\text{aMa } \ast \text{bMb}) \) by simp

hence \( \text{sqrt } ((\text{aMb } + \text{sqd}) / (\text{aMb } - \text{sqd})) \)
  \( = |\text{aMb } + \text{sqd} | / \sqrt{\text{aMa } \ast \text{bMb}} \)

by (simp add: real-sqrt-divide)

from times-divide-times-eq [of \( \text{aMb } + \text{sqd} \text{aMb } - \text{sqd} \text{aMb } - \text{sqd} \)]
have \((?aMb - ?sqd) / (?aMb + ?sqd)\) 
\[= (?aMb - ?sqd)^2 / ((?aMb + ?sqd) * (?aMb - ?sqd))\] 
by (simp add: power2-cq-square)

with \((?aMb + ?sqd) * (?aMb - ?sqd) = ?aMa * ?bMb\)

have \((?aMb - ?sqd) / (?aMb + ?sqd) = (?aMb - ?sqd)^2 / (?aMb * ?bMb)\) by simp

hence \(\sqrt{((?aMb - ?sqd) / (?aMb + ?sqd))}\) 
\[= |?aMb - ?sqd| / \sqrt{(?aMa * ?bMb)}\] 
by (simp add: real-sqrt-divide)

from \((a \neq 0\) and \((b \neq 0\) and \((?pa \in \text{hyp2})\) and \((?pb \in \text{hyp2})\) 

have \(?aMa < 0\) and \(?bMb < 0\) by simp

with \((?aMb + ?sqd) * (?aMb - ?sqd) = ?aMa * ?bMb\)

have \((?aMb + ?sqd) * (?aMb - ?sqd) > 0\) by (simp add: mult-neg-neg)

hence \(?aMb + ?sqd \neq 0\) and \(?aMb - ?sqd \neq 0\) by auto

hence \(\text{sgn} (\sqrt{((?aMb + ?sqd)}) \in \{-1,1\}\) and \(\text{sgn} (\sqrt{(?aMb - ?sqd)}) \in \{-1,1\}\)

by (simp add: real-sqrt-real-def)

from \((\sqrt{((?aMb + ?sqd)}) * (\sqrt{(?aMb - ?sqd)}) > 0\)

have \(\text{sgn} ((\sqrt{((?aMb + ?sqd)}) * (\sqrt{(?aMb - ?sqd)}) = 1\) by simp

hence \(\text{sgn} (\sqrt{((?aMb + ?sqd)}) * \text{sgn} (\sqrt{(?aMb - ?sqd)}) = 1\) by (simp add: sgn-sgn)

with \(\text{sgn} (\sqrt{((?aMb + ?sqd)}) \in \{-1,1\}\) and \(\text{sgn} (\sqrt{(?aMb - ?sqd)}) \in \{-1,1\}\)

have \(\text{sgn} (\sqrt{((?aMb + ?sqd)}) = \text{sgn} (\sqrt{(?aMb - ?sqd)})\) by auto

with \text{abs-plus} \([\text{of} (?aMb + ?sqd) ?aMb - ?sqd]\)

have \((?aMb + ?sqd) + |(?aMb - ?sqd)| = 2 * |?aMb|\) by simp

with \(\sqrt{((?aMb + ?sqd)}) / (\sqrt{?aMb - ?sqd})\) 
\[= |?aMb + ?sqd| / \sqrt{(?aMa * ?bMb)}\] 
and \(\sqrt{((?aMb - ?sqd)) / (?aMb + ?sqd)}\) 
\[= |?aMb - ?sqd| / \sqrt{(?aMa * ?bMb)}\] 
and \text{add-divide-distrib} \([\text{of} \] 
\(|?aMb + ?sqd| |?aMb - ?sqd| \sqrt{(|?aMa * ?bMb|)}\)

have \(\sqrt{((?aMb + ?sqd)}) / (\sqrt{?aMb - ?sqd})\) 
+ \(\sqrt{((?aMb - ?sqd)) / (?aMb + ?sqd))}\) 
\[= 2 * |?aMb| / \sqrt{(?aMa * ?bMb)}\] 
by simp

with \(\text{cosh-dist} \ ?pa \ ?pb\) 
\[= (\sqrt{((?aMb + ?sqd)}) / (?aMb - ?sqd))\] 
+ \(\sqrt{((?aMb - ?sqd)) / (?aMb + ?sqd))}\) 
/ 2.

show \text{cosh-dist} \ ?pa \ ?pb = |?aMb| / \sqrt{(?aMa * ?bMb)}\) by simp

qed

lemma \(\text{cosh-dist-perp-special-case}:\)

assumes \(|x| < 1\) and \(|y| < 1\)

shows \(\text{cosh-dist} \ (\text{proj2-abs} \ (\text{vector} [x,0,1])) \ (\text{proj2-abs} \ (\text{vector} [0,y,1]))\)
\[= (\text{cosh-dist} \ K2-centre \ (\text{proj2-abs} \ (\text{vector} [x,0,1])))\]
* \(\text{cosh-dist} \ K2-centre \ (\text{proj2-abs} \ (\text{vector} [0,y,1])))\)
(is \(\text{cosh-dist} \ ?pa \ ?pb = (\text{cosh-dist} \ ?po \ ?pa) * (\text{cosh-dist} \ ?po \ ?pb)\))
proof –

have \( \text{vector } [0,0,1] \neq (0::\text{real}^3) \) (is \(?a \neq 0\))
and \( \text{vector } [0,y,1] \neq (0::\text{real}^3) \) (is \(?b \neq 0\))
by (unfold vector-def, simp-all add: vec-eq-iff forall-3)

have \(?a \cdot (M * v ?a) = x^2 - 1\) (is \(?a Ma = x^2 - 1\))
and \(?b \cdot (M * v ?b) = y^2 - 1\) (is \(?b Mb = y^2 - 1\))
unfolding vector-def and M-def and inner-vec-def
and matrix-vector-mult-def
by (simp-all add: sum-3 power2-eq-square)

with \(|x| < 1\) and \(|y| < 1\);
have \(?a Ma < 0\) and \(?b Mb < 0\) by (simp-all add: abs-square-less-1)
hence \(?pa \in \text{hyp2}\) and \(?pb \in \text{hyp2}\)
by (simp-all add: M-neg-imp-K2)

with \(?a \neq 0\) and \(?b \neq 0\);

have \(\text{cosh-dist } ?pa \cdot ?pb = |?a \cdot (M * v ?b)| / \sqrt{(|?aMa * ?bMb)}\)
(is \(\text{cosh-dist } ?pa \cdot ?pb = |?aMb| / \sqrt{(|?aMa * ?bMb)}\))
by (rule cosh-dist-formula)
also from \(?aMa = x^2 - 1\) and \(?bMb = y^2 - 1\);
have \(= |?aMb| / \sqrt{(x^2 - 1) \cdot (y^2 - 1)}\) by simp
finally have \(\text{cosh-dist } ?pa \cdot ?pb = 1 / \sqrt{(1 - x^2) \cdot (1 - y^2)}\)
unfolding vector-def and M-def and inner-vec-def
and matrix-vector-mult-def
by (simp add: sum-3 algebra-simps)

let \(?o = \text{vector } [0,0,1]\)
let \(?oMa = ?o \cdot (M * v ?a)\)
let \(?oMb = ?o \cdot (M * v ?b)\)
let \(?oMo = ?o \cdot (M * v ?o)\)

from K2-centre-non-zero and \(?a \neq 0\) and \(?b \neq 0\);
and K2-centre-in-K2 and \(?pa \in \text{hyp2}\) and \(?pb \in \text{hyp2}\);
and cosh-dist-formula [of \(?o\)]

have \(\text{cosh-dist } ?pa \cdot ?pb = |?oMa| / \sqrt{(|?oMa * ?aMa)}\)
and \(\text{cosh-dist } ?pa \cdot ?pb = |?oMb| / \sqrt{(|?oMo * ?bMb)}\)
by (unfold K2-centre-def, simp-all)
hence \(\text{cosh-dist } ?pa \cdot ?pb = 1 / \sqrt{(1 - x^2)}\)
and \(\text{cosh-dist } ?pa \cdot ?pb = 1 / \sqrt{(1 - y^2)}\)
unfolding vector-def and M-def and inner-vec-def
and matrix-vector-mult-def
by (simp-all add: sum-3 power2-eq-square)

with \(\text{cosh-dist } ?pa \cdot ?pb = 1 / \sqrt{(1 - x^2) \cdot (1 - y^2)}\);
show \(\text{cosh-dist } ?pa \cdot ?pb = \text{cosh-dist } ?pa \cdot \text{cosh-dist } ?pa \cdot ?pb\)
by (simp add: real-sqrt-mult)

qed

lemma \(\text{K2-isometry-cross-ratio-endpoints-in-S}:\)
assumes \(a \in \text{hyp2}\) and \(b \in \text{hyp2}\) and is-K2-isometry \(J\) and \(a \neq b\)
shows cross-ratio \(\text{(apply-cltn2 (endpoint-in-S a b) J)}\)
\(\text{(apply-cltn2 (endpoint-in-S b a) J)}\)
\(\text{(apply-cltn2 a J)}\)
\(\text{(apply-cltn2 b J)}\)
cross-ratio (endpoint-in-S a b) (endpoint-in-S b a) a b

proof
  let ?l = proj2-line-through a b
  have proj2-incident a ?l and proj2-incident b ?l
    by (rule proj2-line-through-incident)+
  with ⟨a ≠ b⟩ and ⟨a ∈ hyp2⟩ and ⟨b ∈ hyp2⟩
  have proj2-incident ?p ?l and proj2-incident ?q ?l
    by (simp-all add: endpoint-in-S-incident)
  have proj2-set-Col {?p, ?q, a, b}
    by (unfold proj2-set-Col-def) (simp add: exI[of - ?l])
  from ⟨a ≠ b⟩ and ⟨a ∈ hyp2⟩ and ⟨b ∈ hyp2⟩
  have ?p ≠ ?q by (simp add: endpoint-in-S-swap)

from ⟨a ∈ hyp2⟩ and ⟨b ∈ hyp2⟩ have ?p ∈ S by (simp add: endpoint-in-S)
with ⟨a ∈ hyp2⟩ and ⟨b ∈ hyp2⟩
have a ≠ ?p and b ≠ ?p by (simp-all add: hyp2-S-not-equal)
with ⟨proj2-set-Col {?p, ?q, a, b}⟩ and ⟨?p ≠ ?q⟩
  by (rule cross-ratio-cltn2)

qed

lemma K2-isometry-exp-2dist:
  assumes a ∈ hyp2 and b ∈ hyp2 and is-K2-isometry J
  shows exp-2dist (apply-cltn2 a J) (apply-cltn2 b J) = exp-2dist a b
  (is exp-2dist ?aJ ?bJ = -)
proof cases
  assume a = b
  thus exp-2dist ?aJ ?bJ = exp-2dist a b by (unfold exp-2dist-def) simp
next
  assume a ≠ b
  with apply-cltn2-injective have ?aJ ≠ ?bJ by fast

let ?p = endpoint-in-S a b
let ?q = endpoint-in-S b a
let ?aJ = apply-cltn2 a J
  and ?bJ = apply-cltn2 b J
  and ?qJ = apply-cltn2 ?q J
from ⟨a ≠ b⟩ and ⟨a ∈ hyp2⟩ and ⟨b ∈ hyp2⟩ and ⟨is-K2-isometry J⟩
  by (simp-all add: K2-isometry-endpoint-in-S)
from assms and ⟨a ≠ b⟩
  by (rule K2-isometry-cross-ratio-endpoints-in-S)
and \(a \neq b\) and \(\forall a_j \neq b_j\) show \(\text{exp-dist} a J b J = \text{exp-dist} a b\) by (unfold \text{exp-dist-def}) simp qed

**lemma K2-isometry-cosh-dist:**
assumes \(a \in \text{hyp2} \) and \(b \in \text{hyp2}\) and \(\text{is-K2-isometry} J\)
shows \(\text{cosh-dist} (\text{apply-cltn2} a J) (\text{apply-cltn2} b J) = \text{cosh-dist} a b\)
using assms
by (unfold \text{cosh-dist-def}) (simp add: K2-isometry-exp-2dist)

**lemma cosh-dist-perp:**
assumes \(M\text{-perp} l m\) and \(a \in \text{hyp2}\) and \(b \in \text{hyp2}\) and \(c \in \text{hyp2}\) and \(\text{proj2-incident} a l\) and \(\text{proj2-incident} b l\) and \(\text{proj2-incident} b m\) and \(\text{proj2-incident} c m\)
shows \(\text{cosh-dist} a c = \text{cosh-dist} b a \ast \text{cosh-dist} b c\)

proof –
from \(\langle M\text{-perp} l m \rangle\) and \(\langle b \in \text{hyp2} \rangle\) and \(\langle \text{proj2-incident} b l \rangle\) and \(\langle \text{proj2-incident} b m \rangle\) and \(M\text{-perp-to-compass} [of l m b b]\)
obtain \(J\) where \(\text{is-K2-isometry} J\) and \(\text{apply-cltn2-line equator} J = l\) and \(\text{apply-cltn2-line meridian} J = m\)
by auto

let \(?Ji = \text{cltn2-inverse} J\)
let \(?aJi = \text{apply-cltn2} a \ ?Ji\)
let \(?bJi = \text{apply-cltn2} b \ ?Ji\)
let \(?cJi = \text{apply-cltn2} c \ ?Ji\)
from \(\langle \text{apply-cltn2-line equator} J = l \rangle\) and \(\langle \text{apply-cltn2-line meridian} J = m \rangle\) and \(\langle \text{proj2-incident} a b \rangle\) and \(\langle \text{proj2-incident} b l \rangle\) and \(\langle \text{proj2-incident} b m \rangle\) and \(\langle \text{proj2-incident} c m \rangle\)
have \(\text{proj2-incident} ?aJi \text{ equator and proj2-incident} ?bJi \text{ equator and proj2-incident} ?bJi \text{ meridian and proj2-incident} ?cJi \text{ meridian}\)
by (auto simp add: apply-cltn2-incident)

from \(\langle \text{is-K2-isometry} J \rangle\)
have \(\text{is-K2-isometry} ?aJi\) by (rule cltn2-inverse-is-K2-isometry)
with \(\langle a \in \text{hyp2} \rangle\) and \(\langle c \in \text{hyp2} \rangle\)
have \(?aJi \in \text{hyp2}\) and \(?cJi \in \text{hyp2}\)
by (simp-all add: statement60-one-way)

from \(\langle ?aJi \in \text{hyp2} \rangle\) and \(\langle \text{proj2-incident} ?aJi \text{ equator} \rangle\)
and \(\text{on-equator-in-hyp2-rep}\)
obtain \(x\) where \(|x| < 1\) and \(?aJi = \text{proj2-abs} (\text{vector} [x,0,1])\) by auto
moreover
from \(\langle ?cJi \in \text{hyp2} \rangle\) and \(\langle \text{proj2-incident} ?cJi \text{ meridian} \rangle\)
and \(\text{on-meridian-in-hyp2-rep}\)
obtain \(y\) where \(|y| < 1\) and \(?cJi = \text{proj2-abs} (\text{vector} [0,y,1])\) by auto
moreover
from \(\langle \text{proj2-incident} ?bJi \text{ equator} \rangle\) and \(\langle \text{proj2-incident} ?bJi \text{ meridian} \rangle\)
have \(?bJi = K2\text{-centre}\) by (rule on-equator-meridian-is-K2-centre)
ultimately
have \( \cosh\text{-dist } ?a_Ji \ ?c_Ji = \cosh\text{-dist } ?b_Ji \ ?a_Ji * \cosh\text{-dist } ?b_Ji \ ?c_Ji \)
  by (simp add: \textit{cosh-dist-perp-special-case})
with \( a \in \text{hyp2} \) \textbf{and} \( b \in \text{hyp2} \) \textbf{and} \( c \in \text{hyp2} \) \textbf{and} \( \textit{is-K2-isometry } ?Ji \)
show \( \cosh\text{-dist } a \ c = \cosh\text{-dist } b \ a * \cosh\text{-dist } b \ c \)
  by (simp add: \textit{K2-isometry-cosh-dist})
qed

lemma \textit{are-endpoints-in-S-ordered-cross-ratio}:
assumes \( \text{are-endpoints-in-S } p \ q \ a \ b \)
and \( B_{\text{R}} (\text{cart2-pt } a) (\text{cart2-pt } b) (\text{cart2-pt } p) (\text{is } B_{\text{R}} \ ?ca \ ?cb \ ?cp) \)
shows \( \text{cross-ratio } p \ q \ a \ b \geq 1 \)
proof
from \( \langle \text{are-endpoints-in-S } p \ q \ a \ b \rangle \)
have \( p \neq q \) \textbf{and} \( p \in S \) \textbf{and} \( q \in S \) \textbf{and} \( a \in \text{hyp2} \) \textbf{and} \( b \in \text{hyp2} \)
  and proj2-set-Col \( \{p,q,a,b\} \)
  by (unfold \textit{are-endpoints-in-S-def} simp-all)
from \( \langle a \in \text{hyp2} \rangle \) \textbf{and} \( \langle b \in \text{hyp2} \rangle \) \textbf{and} \( \langle p \in S \rangle \) \textbf{and} \( \langle q \in S \rangle \)
have \( \text{z-non-zero } a \) \textbf{and} \( \text{z-non-zero } b \) \textbf{and} \( \text{z-non-zero } p \) \textbf{and} \( \text{z-non-zero } q \)
  by (simp-all add: \textit{hyp2-S-z-non-zero})
hence \( \text{proj2-abs } (\text{cart2-append1 } p) = p \) (\textit{is proj2-abs } ?cp1 = p)
and \( \text{proj2-abs } (\text{cart2-append1 } q) = q \) (\textit{is proj2-abs } ?cq1 = q)
and \( \text{proj2-abs } (\text{cart2-append1 } a) = a \) (\textit{is proj2-abs } ?ca1 = a)
and \( \text{proj2-abs } (\text{cart2-append1 } b) = b \) (\textit{is proj2-abs } ?cb1 = b)
  by (simp-all add: \textit{proj2-abs-cart2-append1})
from \( \langle b \in \text{hyp2} \rangle \) \textbf{and} \( \langle \text{proj2-set-Col } \{p,q,a,b\} \rangle \)
  obtain \( \langle \text{proj2-set-Col-def} \rangle \)
obtain \( l \) where \( \text{proj2-incident } q \ l \) \textbf{and} \( \text{proj2-incident } p \ l \)
  and \( \text{proj2-incident } a \ l \)
  by (unfold \textit{proj2-set-Col-def} auto)
with \( \langle p \neq q \rangle \) \textbf{and} \( \langle q \in S \rangle \) \textbf{and} \( \langle p \in S \rangle \) \textbf{and} \( \langle a \in \text{hyp2} \rangle \)
and \( S_{\text{hyp2-S-cart2-append1}} \of q \ p \ a \)
  obtain \( k \) where \( k > 0 \) \textbf{and} \( k < 1 \) \textbf{and} \( ?ca1 = k *_R ?cp1 + (1-k) *_R ?cq1 \)
  by auto
from \( \langle \text{z-non-zero } p \rangle \) \textbf{and} \( \langle \text{z-non-zero } q \rangle \)
  have \( ?cp1 \neq 0 \) \textbf{and} \( ?cq1 \neq 0 \)
  by (simp-all add: \textit{cart2-append1-non-zero})
from \( \langle p \neq q \rangle \) \textbf{and} \( \langle \text{proj2-abs } ?cp1 = p \rangle \) \textbf{and} \( \langle \text{proj2-abs } ?cq1 = q \rangle \)
  have \( \text{proj2-abs } ?cp1 \neq \text{proj2-abs } ?cq1 \)
  by simp
from \( k < 1 \) \textbf{have} \( 1-k \neq 0 \) by simp
with \( j < 1 \) have \((1-j) \ast (1-k) \neq 0\) by simp

from \( j < 1 \) and \( k > 0 \) have \((1-j) \ast k > 0\) by simp

from \( ?cb1 = j \ast R \ ?cp1 + (1-j) \ast R \ ?ca1\)
have \( ?cb1 = (j+(1-j) \ast k) \ast R \ ?cp1 + ((1-j) \ast (1-k)) \ast R \ ?cq1\)
by (unfold \( ?ca1 = k \ast R \ ?cp1 + (1-k) \ast R \ ?cq1\); simp add: algebra-simps)
with \( ?ca1 = k \ast R \ ?cp1 + (1-k) \ast R \ ?cq1\)
have proj2-abs \( ?ca1 = proj2-abs \ (k \ast R \ ?cp1 + (1-k) \ast R \ ?cq1)\)
and proj2-abs \( \ ?cb1\)
= proj2-abs \((j+(1-j) \ast k) \ast R \ ?cp1 + ((1-j) \ast (1-k)) \ast R \ ?cq1\)
by simp-all
with \( \langle \text{proj2-abs} \ ?ca1 = a \rangle \) and \( \langle \text{proj2-abs} \ ?cb1 = b \rangle \)
have \( a = \text{proj2-abs} \ (k \ast R \ ?cp1 + (1-k) \ast R \ ?cq1)\)
and \( b = \text{proj2-abs} \ ((j+(1-j) \ast k) \ast R \ ?cp1 + ((1-j) \ast (1-k)) \ast R \ ?cq1)\)
by simp-all
with \( \langle \text{proj2-abs} \ ?cp1 = p \rangle \) and \( \langle \text{proj2-abs} ?cq1 = q \rangle \)
have cross-ratio \( p \ a \ b \)
= cross-ratio \( \langle \text{proj2-abs} \ ?cp1 \rangle \) \( \langle \text{proj2-abs} \ ?cq1 \rangle \)
\( \langle \text{proj2-abs} \ (k \ast R \ ?cp1 + (1-k) \ast R \ ?cq1) \rangle \)
\( \langle \text{proj2-abs} \ ((j+(1-j) \ast k) \ast R \ ?cp1 + ((1-j) \ast (1-k)) \ast R \ ?cq1) \rangle \)
by simp
also from \( \langle \text{proj2-abs} \ ?cp1 \neq 0 \rangle \) and \( \langle \text{proj2-abs} ?cq1 \neq 0 \rangle \) and \( \langle \text{proj2-abs} \ ?cp1 \neq \text{proj2-abs} \ ?cq1 \rangle \)
and \( (1-k) \neq 0 \) and \( (1-j) \ast (1-k) \neq 0 \)
have \( \ldots = (1-k) \ast (j+(1-j) \ast k) \ast R \ ?cp1 + ((1-j) \ast (1-k)) \ast R \ ?cq1 \)
by (rule cross-ratio-abs)
also from \( (1-k) \neq 0 \) have \( \ldots = (j+(1-j) \ast k) / ((1-j) \ast k) \) by simp
also from \( j \geq 0 \) and \( (1-j) \ast k > 0 \) have \( \ldots \geq 1 \) by simp
finally show cross-ratio \( p \ a \ b \geq 1 \).

qed

lemma cross-ratio-S-S-hyp2-hyp2-positive:
assumes are-endpoints-in-S \( p \ a \ b \)
shows cross-ratio \( p \ a \ b \geq 0 \)

proof cases
assume \( B_R \ (\text{cart2-pt} p) \ (\text{cart2-pt} b) \ (\text{cart2-pt} a)\)
hence \( B_R \ (\text{cart2-pt} a) \ (\text{cart2-pt} b) \ (\text{cart2-pt} p)\)
by (rule real-euclid.th3-2)
with assms have cross-ratio \( p \ a \ b \geq 1 \)
by (rule are-endpoints-in-S-ordered-cross-ratio)
thus cross-ratio \( p \ a \ b > 0 \) by simp

next
assume \( \neg B_R \ (\text{cart2-pt} p) \ (\text{cart2-pt} b) \ (\text{cart2-pt} a) \) (is \( \neg B_R \ ?cp \ ?cb \ ?ca))

from \( \langle \text{are-endpoints-in-S} \ p \ a \ b \rangle \)
have \( \langle \text{are-endpoints-in-S} \ p \ b \ a \rangle \) by (rule are-endpoints-in-S-swap-34)

from \( \langle \text{are-endpoints-in-S} \ p \ a \ b \rangle \)
have \( p \in S \) and \( a \in \text{hyp2} \) and \( b \in \text{hyp2} \) and \( \text{proj2-set-Col} \ \{p,q,a,b\} \)
by (unfold \( \text{are-endpoints-in-S-def} \)) simp-all
lemma cosh-dist-general:
assumes are-endpoints-in-S p q a b
shows cosh-dist a b
  = (sqrt (cross-ratio p q a b) + 1 / sqrt (cross-ratio p q a b)) / 2
proof
  from (are-endpoints-in-S p q a b)
  have p ≠ q and p ∈ S and q ∈ S and a ∈ hyp2 and b ∈ hyp2
    and proj2-set-Col {p,q,a,b}
      by (unfold are-endpoints-in-S-def simp-all)
  from (a ∈ hyp2) and (b ∈ hyp2) and (p ∈ S) and (q ∈ S)
  have a ≠ p and a ≠ q and b ≠ p and b ≠ q
    by (simp-all add: hyp2-S-not-equal)
  show cosh-dist a b
    = (sqrt (cross-ratio p q a b) + 1 / sqrt (cross-ratio p q a b)) / 2
  proof
    assume a = b
    hence cosh-dist a b = 1 by (unfold cosh-dist-def exp-2dist-def simp)
  next
    assume a ≠ b

let \( ?r = \text{endpoint-in-S} \ a \ b \)
let \( ?s = \text{endpoint-in-S} \ b \ a \)

from \( \langle a \neq b \rangle \)
have \( \exp-2\text{dist} \ a \ b = \text{cross-ratio} \ ?r \ ?s \ a \ b \) by \( \text{unfold \ exp-2\text{dist-def}} \) \( \text{simp} \)

from \( \langle a \neq b \rangle \) and \( \langle \text{are-endpoints-in-S} \ p \ q \ a \ b \rangle \)
have \( (p = ?r \land q = ?s) \lor (q = ?r \land p = ?s) \) by \( \text{rule \ are-endpoints-in-S} \)

show \( \coshdist a \ b \)
\( = (\sqrt{\text{cross-ratio} p \ q \ a \ b}) + 1 / \sqrt{\text{cross-ratio} p \ q \ a \ b}) / 2 \)
\( \text{proof \ cases} \)
assume \( p = ?r \land q = ?s \)
with \( \langle \exp-2\text{dist} \ a \ b = \text{cross-ratio} \ ?r \ ?s \ a \ b \rangle \)
have \( \exp-2\text{dist} \ a \ b = \text{cross-ratio} p \ q \ a \ b \) by \( \text{simp} \)
thus \( \coshdist a \ b \)
\( = (\sqrt{\text{cross-ratio} p \ q \ a \ b}) + 1 / \sqrt{\text{cross-ratio} p \ q \ a \ b}) / 2 \)
by \( \text{unfold \ cosh-dist-def} \) \( \text{simp add: real-sqrt-divide} \)

next
assume \( \neg (p = ?r \land q = ?s) \)
with \( \langle (p = ?r \land q = ?s) \lor (q = ?r \land p = ?s) \rangle \)
have \( q = ?r \land p = ?s \) by \( \text{simp-all} \)
with \( \langle \exp-2\text{dist} \ a \ b = \text{cross-ratio} \ ?r \ ?s \ a \ b \rangle \)
have \( \exp-2\text{dist} \ a \ b = \text{cross-ratio} q \ p \ a \ b \) by \( \text{simp} \)

have \( \{q,p,a,b\} = \{p,q,a,b\} \) by \( \text{auto} \)
with \( \langle \text{proj2-set-Col} \ \{p,q,a,b\} \rangle \) and \( \langle p \neq q \rangle \) and \( \langle a \neq p \rangle \) and \( \langle b \neq p \rangle \)
and \( \langle a \neq q \rangle \) and \( \langle b \neq q \rangle \)
have \( \text{cross-ratio-correct} p \ a \ b \) and \( \text{cross-ratio-correct} q \ p \ a \ b \)
by \( \text{unfold \ cross-ratio-correct-def} \) \( \text{simp-all} \)
\( \text{hence} \ \text{cross-ratio} q \ p \ a \ b = 1 / (\text{cross-ratio} p \ q \ a \ b) \)
by \( \text{rule \ cross-ratio-swap-12} \)
with \( \langle \exp-2\text{dist} \ a \ b = \text{cross-ratio} q \ p \ a \ b \rangle \)
have \( \exp-2\text{dist} \ a \ b = 1 / (\text{cross-ratio} p \ q \ a \ b) \) by \( \text{simp} \)
thus \( \coshdist a \ b \)
\( = (\sqrt{\text{cross-ratio} p \ q \ a \ b}) + 1 / \sqrt{\text{cross-ratio} p \ q \ a \ b}) / 2 \)
by \( \text{unfold \ cosh-dist-def} \) \( \text{simp add: real-sqrt-divide} \)
qed
qed

lemma \( \exp-2\text{dist-positive} \):
assumes \( a \in \text{hyp2} \) and \( b \in \text{hyp2} \)
shows \( \exp-2\text{dist} a \ b > 0 \)
\( \text{proof \ cases} \)
assume \( a = b \)
thus \( \exp-2\text{dist} a \ b > 0 \) by \( \text{unfold \ exp-2\text{dist-def}} \) \( \text{simp} \)
next
assume \( a \neq b \)

200
let \( ?p = \text{endpoint-in-S} \ a \ b \)
let \( ?q = \text{endpoint-in-S} \ b \ a \)
from \( (a \neq b) \) and \((a \in \text{hyp2})\) and \((b \in \text{hyp2})\)
have \(\text{are-endpoints-in-S} \ ?p \ ?q \ a \ b\)
by \(\text{rule endpoints-in-S-are-endpoints-in-S}\)
hence \(\text{cross-ratio} \ ?p \ ?q \ a \ b > 0\)
by \(\text{rule cross-ratio-S-S-hyp2-hyp2-positive}\)
with \((a \neq b)\)

lemma \(\text{cosh-dist-at-least-1}\):
assumes \(a \in \text{hyp2}\) and \(b \in \text{hyp2}\)
shows \(\text{cosh-dist} \ a \ b \geq 1\)
proof –
from assms have \(\text{exp-2dist} \ a \ b > 0\)
by \(\text{rule exp-2dist-positive}\)
with \(\text{am-gm2}(1)\)
[of \(\sqrt{\text{exp-2dist} \ a \ b} \) \(\sqrt{1/\text{exp-2dist} \ a \ b}\)]
show \(\text{cosh-dist} \ a \ b \geq 1\)
by \(\text{unfold cosh-dist-def}\) \(\text{(simp add: real-sqrt-mult [symmetric])}\)
qed

lemma \(\text{cosh-dist-positive}\):
assumes \(a \in \text{hyp2}\) and \(b \in \text{hyp2}\)
shows \(\text{cosh-dist} \ a \ b > 0\)
proof –
from assms have \(\text{cosh-dist} \ a \ b \geq 1\)
by \(\text{rule cosh-dist-at-least-1}\)
thus \(\text{cosh-dist} \ a \ b > 0\)
by \(\text{simp}\)
qed

lemma \(\text{cosh-dist-perp-divide}\):
assumes \(M\text{-perp} l m\) and \(a \in \text{hyp2}\) and \(b \in \text{hyp2}\) and \(c \in \text{hyp2}\)
and \(\text{proj2-incident} \ a \ l\) and \(\text{proj2-incident} \ b \ l\) and \(\text{proj2-incident} \ b \ m\)
and \(\text{proj2-incident} \ c \ m\)
shows \(\text{cosh-dist} \ b \ c = \text{cosh-dist} \ a \ c / \text{cosh-dist} \ b \ a\)
proof –
from \(\langle b \in \text{hyp2}\rangle\) and \(\langle a \in \text{hyp2}\rangle\)
have \(\text{cosh-dist} \ b \ a > 0\)
by \(\text{rule cosh-dist-positive}\)

from assms
have \(\text{cosh-dist} \ a \ c = \text{cosh-dist} \ b \ a * \text{cosh-dist} \ b \ c\)
by \(\text{rule cosh-dist-perp}\)
with \(\langle \text{cosh-dist} \ b \ a > 0\rangle\)
show \(\text{cosh-dist} \ b \ c = \text{cosh-dist} \ a \ c / \text{cosh-dist} \ b \ a\)
by \(\text{simp}\)
qed

lemma \(\text{real-hyp2-C-cross-ratio-endpoints-in-S}\):
assumes \(a \neq b\) and \(a \ b \equiv \mathcal{K} \ c \ d\)
shows \(\text{cross-ratio} \ \text{(endpoint-in-S} \ (\text{Rep-hyp2} \ a) \ (\text{Rep-hyp2} \ b))\)
\(\text{(endpoint-in-S} \ (\text{Rep-hyp2} \ b) \ (\text{Rep-hyp2} \ a)) \ (\text{Rep-hyp2} \ a) \ (\text{Rep-hyp2} \ b)\)
\(= \text{cross-ratio} \ \text{(endpoint-in-S} \ (\text{Rep-hyp2} \ c) \ (\text{Rep-hyp2} \ d))\)
\(\text{(endpoint-in-S} \ (\text{Rep-hyp2} \ d) \ (\text{Rep-hyp2} \ c)) \ (\text{Rep-hyp2} \ c) \ (\text{Rep-hyp2} \ d)\)
\(\text{(is cross-ratio} \ ?p \ ?q \ ?a' \ ?b' = \text{cross-ratio} \ ?r \ ?s \ ?c' \ ?d')\)
proof –

from \langle a \neq b \rangle \quad \text{and} \quad \langle a \equiv_K c \quad b \equiv_K d \rangle \quad \text{have} \quad c \neq d \quad \text{by} \quad (\text{auto simp add: hyp2.A3'})

with \langle a \neq b \rangle \quad \text{have} \quad ?a' \neq ?b' \quad \text{and} \quad ?c' \neq ?d' \quad \text{by} \quad (\text{unfold Rep-hyp2-inject})

from \langle a \equiv_K c \quad b \equiv_K d \rangle

obtain \ J \quad \text{where} \quad \text{is-K2-isometry} \ J \quad \text{and} \quad \text{hyp2-cltn2} \ a \ J = c

\quad \text{and} \quad \text{hyp2-cltn2} \ b \ J = d

\quad \text{by} \quad (\text{unfold real-hyp2-C-def}) \quad \text{auto}

hence \text{apply-cltn2} \ ?a' J = ?c' \quad \text{and} \quad \text{apply-cltn2} \ ?b' J = ?d'

\quad \text{by} \quad (\text{simp-all add: Rep-hyp2-cltn2 [symmetric]})

with \langle ?a' \neq ?b' \rangle \quad \text{and} \quad \langle \text{is-K2-isometry} \ J \rangle

have \text{apply-cltn2} \ ?p J = ?r \quad \text{and} \quad \text{apply-cltn2} \ ?q J = ?s

\quad \text{by} \quad (\text{simp-all add: Rep-hyp2 K2-isometry-endpoint-in-S})

from \langle ?a' \neq ?b' \rangle

have \text{proj2-set-Col} \ \{ ?p, ?q, ?a', ?b' \}

\quad \text{by} \quad (\text{simp add: Rep-hyp2 proj2-set-Col-endpoints-in-S})

from \langle ?a' \neq ?b' \rangle

have \ ?p \neq ?q \quad \text{by} \quad (\text{simp add: Rep-hyp2 endpoint-in-S-swap})

have \ ?p \in S \quad \text{by} \quad (\text{simp add: Rep-hyp2 endpoint-in-S})

hence \ ?a' \neq ?p \quad \text{and} \quad ?b' \neq ?p \quad \text{by} \quad (\text{simp-all add: Rep-hyp2 hyp2-S-not-equal})

with \langle \text{proj2-set-Col} \ \{ ?p, ?q, ?a', ?b' \} \rangle \quad \text{and} \quad \langle ?p \neq ?q \rangle

have \ cross-ratio \ ?p \ ?q \ ?a' \ ?b' = cross-ratio \ (\text{apply-cltn2} \ ?p J) \ (\text{apply-cltn2} \ ?q J)

\quad (\text{apply-cltn2} \ ?a' J) \ (\text{apply-cltn2} \ ?b' J)

\quad \text{by} \quad (\text{rule cross-ratio-cltn2 [symmetric]})

with \langle \text{apply-cltn2} \ ?p J = ?r \rangle \quad \text{and} \quad \langle \text{apply-cltn2} \ ?q J = ?s \rangle

\quad \text{and} \quad \langle \text{apply-cltn2} \ ?a' J = ?c' \rangle \quad \text{and} \quad \langle \text{apply-cltn2} \ ?b' J = ?d' \rangle

show \ cross-ratio \ ?p \ ?q \ ?a' \ ?b' = cross-ratio \ ?r \ ?s \ ?c' \ ?d' \quad \text{by} \quad \text{simp}

qed

lemma real-hyp2-C-exp-2dist:

assumes \ a \ b \equiv_K c \ d

does \ \text{exp-2dist} \ (\text{Rep-hyp2} \ a) \ (\text{Rep-hyp2} \ b)

= \text{exp-2dist} \ (\text{Rep-hyp2} \ c) \ (\text{Rep-hyp2} \ d)
\ (\text{is exp-2dist} \ ?a' \ ?b' = \text{exp-2dist} \ ?c' \ ?d')

proof –

from \langle a \equiv_K c \quad b \equiv_K d \rangle

obtain \ J \quad \text{where} \quad \text{is-K2-isometry} \ J \quad \text{and} \quad \text{hyp2-cltn2} \ a \ J = c

\quad \text{and} \quad \text{hyp2-cltn2} \ b \ J = d

\quad \text{by} \quad (\text{unfold real-hyp2-C-def}) \quad \text{auto}

hence \text{apply-cltn2} \ ?a' J = ?c' \quad \text{and} \quad \text{apply-cltn2} \ ?b' J = ?d'

\quad \text{by} \quad (\text{simp-all add: Rep-hyp2-cltn2 [symmetric]})

from \text{Rep-hyp2} \ [\text{of} \ a] \ \text{and} \ \text{Rep-hyp2} \ [\text{of} \ b] \ \text{and} \ \langle \text{is-K2-isometry} \ J \rangle

have \text{exp-2dist} \ (\text{apply-cltn2} \ ?a' J) \ (\text{apply-cltn2} \ ?b' J) = \text{exp-2dist} \ ?a' ?b'

\quad \text{by} \quad (\text{rule K2-isometry-exp-2dist})

with \langle \text{apply-cltn2} \ ?a' J = ?c' \rangle \quad \text{and} \quad \langle \text{apply-cltn2} \ ?b' J = ?d' \rangle
lemma real-hyp2-C-cosh-dist:
assumes a b \equiv k c d
shows \text{cosh-dist (Rep-hyp2 a) (Rep-hyp2 b) = cosh-dist (Rep-hyp2 c) (Rep-hyp2 d)}
using assms
by (unfold \text{cosh-dist-def}) (simp add: \text{real-hyp2-C-exp-2dist})

lemma cross-ratio-in-terms-of-cosh-dist:
assumes \text{are-endpoints-in-S p q a b}
and \text{B\textsubscript{R} (cart2-\text{pt} a) (cart2-\text{pt} b) (cart2-\text{pt} p)}
shows \text{cross-ratio p q a b = 2 * (cosh-dist a b)\(^2\) + 2 * cosh-dist a b * sqrt ((cosh-dist a b)\(^2\) - 1) - 1}
(is \?pqab = 2 * \?ab\(^2\) + 2 * \?ab * sqrt (\?ab\(^2\) - 1) - 1)
proof –
from \text{are-endpoints-in-S p q a b}
have \text{?ab = (sqrt ?pqab + 1 / sqrt ?pqab) / 2 by (rule \text{cosh-dist-general})}
hence \text{sqrt ?pqab - 2 * \?ab + 1 / sqrt ?pqab = 0 by simp}
hence \text{sqrt ?pqab * (sqrt ?pqab - 2 * \?ab + 1 / sqrt ?pqab) = 0 by simp}
moreover from \text{assms}
have \text{?pqab \geq 1 by (rule \text{are-endpoints-in-S-ordered-cross-ratio)}
ultimately have \text{?pqab - 2 * \?ab * (sqrt ?pqab) + 1 = 0}
by (simp add: algebra-simps real-sqrt-mult \text{[symmetric]}]
with \text{?pqab \geq 1} and \text{discriminant-iff [of 1 sqrt ?pqab - 2 * \?ab 1]}
have \text{sqrt ?pqab = (2 * \?ab + sqrt (4 * \?ab\(^2\) - 4)) / 2}
\lor \text{sqrt ?pqab = (2 * \?ab - sqrt (4 * \?ab\(^2\) - 4)) / 2}
unfolding \text{discrim-def}
by (simp add: real-sqrt-mult \text{[symmetric]} \text{power2-\text{eq-square]}]
moreover have \text{sqrt (4 * \?ab\(^2\) - 4) = sqrt (4 * (\?ab\(^2\) - 1)) by simp}
hence \text{sqrt (4 * \?ab\(^2\) - 4) = 2 * sqrt (\?ab\(^2\) - 1)}
by (unfold real-sqrt-mult simp)
ultimately have \text{sqrt ?pqab = 2 * (\?ab + sqrt (\?ab\(^2\) - 1)) / 2}
\lor \text{sqrt ?pqab = 2 * (\?ab - sqrt (\?ab\(^2\) - 1)) / 2}
by simp
hence \text{sqrt ?pqab = \?ab + sqrt (\?ab\(^2\) - 1)}
\lor \text{sqrt ?pqab = \?ab - sqrt (\?ab\(^2\) - 1)}
by (simp only: nonzero-mult-div-cancel-left [of 2])

from \text{are-endpoints-in-S p q a b}
have \text{a \in \text{hyp2 and b \in \text{hyp2 by (unfold are-endpoints-in-S-def)} simp-all}
hence \text{\?ab \geq 1 by (rule \text{cosh-dist-at-least-1)}
hence \text{\?ab\(^2\) \geq 1 by simp}
hence \text{sqrt (\?ab\(^2\) - 1) \geq 0 by simp}
hence \text{sqrt (\?ab\(^2\) - 1) * sqrt (\?ab\(^2\) - 1) = \?ab\(^2\) - 1}
by (simp add: real-sqrt-mult \text{[symmetric]}]
hence (\?ab + sqrt (\?ab\(^2\) - 1)) * (\?ab - sqrt (\?ab\(^2\) - 1)) = 1
by (simp add: algebra-simps power2-\text{eq-square]}
have $?ab - \sqrt{(?ab^2 - 1)} \leq 1$

proof (rule econtr)

  assume $\neg (\sqrt{(?ab^2 - 1)} \leq 1)$
  hence $1 < \sqrt{(?ab^2 - 1)}$ by simp
  also from $\sqrt{(?ab^2 - 1)} \geq 0$
  have $\ldots \leq \sqrt{(?ab^2 - 1)}$ by simp
  finally have $1 < \sqrt{(?ab^2 - 1)}$ by simp

with $1 < \sqrt{(?ab^2 - 1)}$
and \multistdmono{of $1 \sqrt{(?ab^2 - 1)} 1 \sqrt{(?ab^2 - 1)}$}

have $1 < (\sqrt{(?ab^2 - 1)}) (\sqrt{(?ab^2 - 1)})$ by simp
with $(\sqrt{(?ab^2 - 1)}) = (\sqrt{(?ab^2 - 1)}) = 1$
show False by simp

qed

have $\sqrt{?pqab} = ?ab + \sqrt{(?ab^2 - 1)}$
proof (rule econtr)

  assume $\sqrt{?pqab} \neq ?ab + \sqrt{(?ab^2 - 1)}$
  with $(\sqrt{?pqab} = ?ab + \sqrt{(?ab^2 - 1)})$
  $\lor \sqrt{?pqab} = \sqrt{(?ab^2 - 1)}$

have $(?pqab = ?ab - \sqrt{(?ab^2 - 1)})$ by simp
with $(?ab - \sqrt{(?ab^2 - 1)} \leq 1)$ have $(?pqab \leq 1)$ by simp
with $(?pqab \geq 1)$ have $(?pqab = 1)$ by simp
with $(?pqab = ?ab - \sqrt{(?ab^2 - 1)})$
  and $(?ab + \sqrt{(?ab^2 - 1)}) (\sqrt{(?ab^2 - 1)}) = 1$
have $(?ab + \sqrt{(?ab^2 - 1)}) = 1$ by simp
with $(?pqab = 1)$ have $(?pqab = ?ab + \sqrt{(?ab^2 - 1)})$ by simp
with $(?pqab \neq ?ab + \sqrt{(?ab^2 - 1)})$ show False ..

qed

moreover from $(?pqab \geq 1)$ have $(?pqab = (\sqrt{?pqab})^2)$ by simp
ultimately have $(?pqab = (\sqrt{(?ab^2 - 1)})^2)$ by simp
with $(\sqrt{(?ab^2 - 1)} \cdot \sqrt{(?ab^2 - 1)} = ?ab^2 - 1)$
show $(?pqab = 2 * ?ab^2 + 2 * ?ab \cdot \sqrt{(?ab^2 - 1)}) - 1$
  by (simp add: power2-eq-square algebra-simps)

qed

lemma arc-endpoints-in-S-cross-ratio-correct:
  assumes arc-endpoints-in-S p q a b
  shows cross-ratio-correct p q a b

proof
  from arc-endpoints-in-S p q a b

  have $p \neq q$ and $p \in S$ and $q \in S$ and $a \in hyp2$ and $b \in hyp2$
and proj2-set-Col {p,q,a,b}
  by (unfold arc-endpoints-in-S-def) simp-all

  from $(a \in hyp2)$ and $(b \in hyp2)$ and $(p \in S)$ and $(q \in S)$
  have $(a \neq p)$ and $b \neq p$ and $(a \neq q)$ by (simp-all add: hyp2-S-not-equal)
  with $(proj2-set-Col \{p,q,a,b\})$ and $(p \neq q)$
show cross-ratio-correct p q a b by (unfold cross-ratio-correct-def) simp
qed

lemma endpoints-in-S-cross-ratio-correct:
  assumes a ≠ b and a ∈ hyp2 and b ∈ hyp2
  shows cross-ratio-correct (endpoint-in-S a b) (endpoint-in-S b a) a b
proof –
  from assms
  have are-endpoints-in-S (endpoint-in-S a b) (endpoint-in-S b a) a b
    by (rule endpoints-in-S-are-endpoints-in-S)
  thus cross-ratio-correct (endpoint-in-S a b) (endpoint-in-S b a) a b
    by (rule are-endpoints-in-S-cross-ratio-correct)
qed

lemma endpoints-in-S-perp-foot-cross-ratio-correct:
  assumes a ∈ hyp2 and b ∈ hyp2 and c ∈ hyp2 and a ≠ b
  and proj2-incident a l and proj2-incident b l
  shows cross-ratio-correct (endpoint-in-S a b) (endpoint-in-S b a) a (perp-foot c l)
  (is cross-ratio-correct ?p ?q a ?d)
proof –
  from assms
  have are-endpoints-in-S ?p ?q a ?d
    by (rule endpoints-in-S-perp-foot-are-endpoints-in-S)
  thus cross-ratio-correct ?p ?q a ?d
    by (rule are-endpoints-in-S-cross-ratio-correct)
 qed

lemma cosh-dist-unique:
  assumes a ∈ hyp2 and b ∈ hyp2 and c ∈ hyp2 and p ∈ S
  and B (cart2-pt a) (cart2-pt b) (cart2-pt p) (is B ?ca ?cb ?cp)
  and B (cart2-pt a) (cart2-pt c) (cart2-pt p) (is B ?ca ?cc ?cp)
  and cosh-dist a b = cosh-dist a c (is ?ab = ?ac)
  shows b = c
proof –
  let ?q = endpoint-in-S p a

  from (a ∈ hyp2) and (b ∈ hyp2) and (c ∈ hyp2) and (p ∈ S)
  have z-non-zero a and z-non-zero b and z-non-zero c and z-non-zero p
    by (simp-all add: hyp2-S-z-non-zero)
  with (B ?ca ?cb ?cp) and (B ?ca ?cc ?cp)
  have ∃ l. proj2-incident a l ∧ proj2-incident b l ∧ proj2-incident p l
    and ∃ m. proj2-incident a m ∧ proj2-incident c m ∧ proj2-incident p m
    by (simp-all add: euclid-B-cart2-common-line)
  then obtain l and m where
    proj2-incident a l and proj2-incident b l and proj2-incident p l
    and proj2-incident a m and proj2-incident c m and proj2-incident p m
    by auto
from \( \langle a \in \text{hyp2} \rangle \) and \( \langle p \in S \rangle \) have \( a \neq p \) by (rule hyp2-S-not-equal)

with \( \langle \text{proj2-incident } a \ b \rangle \) and \( \langle \text{proj2-incident } p \ b \rangle \)
  and \( \langle \text{proj2-incident } a \ m \rangle \) and \( \langle \text{proj2-incident } p \ m \rangle \) and \( \text{proj2-incident-unique} \)
have \( l = m \) by fast

with \( \langle \text{proj2-incident } c \ m \rangle \) have \( \text{proj2-incident } c \ l \) by simp

with \( \langle a \in \text{hyp2} \rangle \) and \( \langle b \in \text{hyp2} \rangle \) and \( \langle c \in \text{hyp2} \rangle \) and \( \langle p \in S \rangle \)
  and \( \langle \text{proj2-incident } a \ b \rangle \) and \( \langle \text{proj2-incident } b \ b \rangle \) and \( \langle \text{proj2-incident } p \ b \rangle \)
have \( \text{are-endpoints-in-S } p \ ?q \ a \ b \) and \( \text{are-endpoints-in-S } p \ ?q \ a \ c \)
  by (simp-all add: end-and-opposite-are-endpoints-in-S)

with \( \text{are-endpoints-in-S-swap-34} \)
have \( \text{are-endpoints-in-S } p \ ?q \ a \ b \) and \( \text{are-endpoints-in-S } p \ ?q \ a \ c \) by fast+

hence \( \text{cross-ratio-correct } p \ ?q \ a \ b \) and \( \text{cross-ratio-correct } p \ ?q \ a \ c \)
  by (simp-all add: are-endpoints-in-S-cross-ratio-correct)

moreover
from \( \langle \text{are-endpoints-in-S } p \ ?q \ a \ b \rangle \) and \( \langle \text{are-endpoints-in-S } p \ ?q \ a \ c \rangle \)
  and \( \langle B \ I \ R \ ?ca \ ?cb \ ?cp \rangle \) and \( \langle B \ I \ R \ ?ca \ ?cc \ ?cp \rangle \)
have \( \text{cross-ratio } p \ ?q \ a \ b \) = \( 2 \ast \frac{?ab}{?ab} + 2 \ast \frac{?ac}{?ac} \ast \sqrt{?ab^2 - 1} - 1 \)
  and \( \text{cross-ratio } p \ ?q \ a \ c \) = \( 2 \ast \frac{?ac}{?ac} + 2 \ast \frac{?ac}{?ac} \ast \sqrt{?ac^2 - 1} - 1 \)
  by (simp-all add: cross-ratio-in-terms-of-cosh-dist)

with \( \langle ?ab = ?ac \rangle \) have \( \text{cross-ratio } p \ ?q \ a \ b = \text{cross-ratio } p \ ?q \ a \ c \) by simp

ultimately show \( b = c \) by (rule cross-ratio-unique)

qed

lemma cosh-dist-swap:
  assumes \( a \in \text{hyp2} \) and \( b \in \text{hyp2} \)
  shows \( \text{cosh-dist } a \ b = \text{cosh-dist } b \ a \)
proof (rule ccontr)
  assume \( a \neq b \)
  with \( \langle a \in \text{hyp2} \rangle \) and \( \langle b \in \text{hyp2} \rangle \)
  have \( \text{cross-ratio-correct } \langle \text{endpoint-in-S } a \ b \rangle \) \( \langle \text{endpoint-in-S } b \ a \rangle \) \( a \ b \)
    (is cross-ratio-correct ?p ?q \ a \ b)
    by (simp add: endpoints-in-S-cross-ratio-correct)
moreover

lemma exp-2dist-1-equal:
  assumes \( a \in \text{hyp2} \) and \( b \in \text{hyp2} \) and \( \text{exp-2dist } a \ b = 1 \)
  shows \( a = b \)
proof (rule ccontr)
  assume \( a \neq b \)
  with \( \langle a \in \text{hyp2} \rangle \) and \( \langle b \in \text{hyp2} \rangle \)
  have \( \text{cross-ratio-correct } \langle \text{endpoint-in-S } a \ b \rangle \) \( \langle \text{endpoint-in-S } b \ a \rangle \) \( a \ b \)
    (is cross-ratio-correct ?p ?q \ a \ b)
    by (simp add: endpoints-in-S-cross-ratio-correct)
moreover
from \( a \neq b \) 

have \( \exp{2\text{dist}} \ a \ b = \text{cross-ratio} \ ?p \ ?q \ a \ b \) by (unfold \( \exp{2\text{dist-def}} \)) simp 

with \( \exp{2\text{dist}} \ a \ b = 1 \) have \( \text{cross-ratio} \ ?p \ ?q \ a \ b = 1 \) by simp 

ultimately have \( a = b \) by (rule \( \text{cross-ratio-1-equal} \)) 

with \( a \neq b \) show False ..

qed

8.11.1 A formula for a cross ratio involving a perpendicular foot

lemma described-perp-foot-cross-ratio-formula:
assumes \( a \neq b \) and \( c \in \text{hyp2} \) and \( \text{are-endpoints-in-S} \ p \ q \ a \ b \) and \( \text{proj2-incident} \ p \ l \) and \( \text{proj2-incident} \ q \ l \) and \( \text{M-perp} \ l \ m \) and \( \text{proj2-incident} \ d \ l \) and \( \text{proj2-incident} \ d \ m \) and \( \text{proj2-incident} \ c \ m \) shows \( \text{cross-ratio} \ p \ q \ d \ a = (\cosh\text{-dist} \ b \ c \ast \sqrt{\text{cross-ratio} \ p \ q \ a \ b} - \cosh\text{-dist} \ a \ c) / (\cosh\text{-dist} \ a \ c \ast \sqrt{\text{cross-ratio} \ p \ q \ a \ b} - \cosh\text{-dist} \ b \ c) \)

(is \( ?pqda = (?bc \ast sqrt ?pqab - ?ac) / (?ac \ast ?pqab - ?bc \ast sqrt ?pqab) \))

proof –

let \( ?da = cosp\text{-dist} \ d \ a \) 

let \( ?db = cosp\text{-dist} \ d \ b \) 

let \( ?dc = cosp\text{-dist} \ d \ c \) 

let \( ?pqdb = cosp\text{-ratio} \ p \ q \ d \ b \)

from \( \text{are-endpoints-in-S} \ p \ q \ a \ b \) 

have \( p \neq q \) and \( p \in S \) and \( q \in S \) and \( a \in \text{hyp2} \) and \( b \in \text{hyp2} \) and \( \text{proj2-set-Col} \ \{p,q,a,b\} \) by (unfold \( \text{are-endpoints-in-S-def} \)) simp-all 

from \( \text{proj2-set-Col} \ \{p,q,a,b\} \) 

obtain \( l' \) where \( \text{proj2-incident} \ p \ l' \) and \( \text{proj2-incident} \ q \ l' \) and \( \text{proj2-incident} \ a \ l' \) and \( \text{proj2-incident} \ b \ l' \) by (unfold \( \text{proj2-set-Col-def} \)) auto 

from \( p \neq q \) and \( \text{proj2-incident} \ p \ l' \) and \( \text{proj2-incident} \ q \ l' \) and \( \text{proj2-incident} \ a \ l' \) and \( \text{proj2-incident} \ b \ l' \) and \( \text{proj2-incident-unique} \) 

have \( l' = l \) by fast 

with \( \text{proj2-incident} \ a \ l' \) and \( \text{proj2-incident} \ b \ l' \) 

have \( \text{proj2-incident} \ a \ l \) and \( \text{proj2-incident} \ b \ l \) by simp-all 

from \( \text{M-perp} \ l \ m \) and \( \{a \in \text{hyp2} \} \) and \( \text{proj2-incident} \ a \ l \) and \( \{c \in \text{hyp2} \} \) and \( \text{proj2-incident} \ c \ m \) and \( \text{proj2-incident} \ d \ l \) and \( \text{proj2-incident} \ d \ m \) 

have \( d \in \text{hyp2} \) by (rule \( \text{M-perp-hyp2} \)) 

with \( \{a \in \text{hyp2} \} \) and \( \{b \in \text{hyp2} \} \) and \( \{c \in \text{hyp2} \} \) 

have \( ?bc > 0 \) and \( ?da > 0 \) and \( ?ac > 0 \) 

by (simp-all add: \( \text{cosh-dist-positive} \))

from \( \text{proj2-incident} \ p \ l \) and \( \text{proj2-incident} \ q \ l \) and \( \text{proj2-incident} \ d \ l \) and \( \text{proj2-incident} \ a \ l \) and \( \text{proj2-incident} \ b \ l \)
have \( \text{proj2-set-Col} \{ p, q, d, a \} \) and \( \text{proj2-set-Col} \{ p, q, d, b \} \) and \( \text{proj2-set-Col} \{ p, q, a, b \} \) by (unfold \text{proj2-set-Col-def}) (simp-all add: exI [of \(-\)] ) with \( p \neq q \) and \( p \in S \) and \( q \in S \) and \( d \in \text{hyp2} \) and \( (a \in \text{hyp2} \) and \( b \in \text{hyp2} \) have \( \text{are-endpoints-in-S} p q d a \) and \( \text{are-endpoints-in-S} p q d b \) and \( \text{are-endpoints-in-S} p q a b \) by (unfold \text{are-endpoints-in-S-def}) simp-all hence \( ?pqda > 0 \) and \( ?pqdb > 0 \) and \( ?pqab > 0 \) by (simp-all add: \text{cross-ratio-S-S-hyp2-hyp2-positive}) from \( \text{proj2-incident} p b \) and \( \text{proj2-incident} q b \) and \( \text{proj2-incident} a b \) have \( \text{proj2-Col} p q a \) by (rule \text{proj2-incident-Col}) from \( a \in \text{hyp2} \) and \( b \in \text{hyp2} \) and \( p \in S \) and \( q \in S \) have \( a \neq p \) and \( a \neq q \) and \( b \neq p \) by (simp-all add: \text{hyp2-S-not-equal}) from \( \text{proj2-Col} p q a \) and \( p \neq q \) and \( a \neq p \) and \( a \neq q \) have \( ?pqda = ?pqdb \) by (rule \text{cross-ratio-product} [\text{symmetric}]) from \( \langle M-\text{perp} l m \rangle \) and \( a \in \text{hyp2} \) and \( b \in \text{hyp2} \) and \( c \in \text{hyp2} \) and \( d \in \text{hyp2} \) and \( \text{proj2-incident} a b \) and \( \text{proj2-incident} b b \) and \( \text{proj2-incident} b d \) and \( \text{proj2-incident} d b \) and \( \text{cosh-dist-perp-divide} \ [\text{of} \ l m - d c] \) have \( ?dc = ?ac \) / \( ?da \) and \( ?dc = ?bc \) / \( ?db \) by fast+ hence \( ?ac / ?da = ?bc / ?db \) by simp with \( ?bc > 0 \) and \( ?da > 0 \) have \( ?ac / ?bc = ?da / ?db \) by (simp add: \text{field-simps}) also from \( \langle \text{are-endpoints-in-S} p q d a \rangle \) and \( \text{are-endpoints-in-S} p q d b \) have \( \ldots \) \( = 2 * (\sqrt{?pqda} + 1 / (\sqrt{?pqda})) \) / \( (2 * (\sqrt{?pqdb} + 1 / (\sqrt{?pqdb}))) \) by (simp add: \text{cosh-dist-general}) also have \( \ldots = (\sqrt{?pqda} + 1 / (\sqrt{?pqda})) / (\sqrt{?pqdb} + 1 / (\sqrt{?pqdb})) \) by (simp only: \text{mult-divide-mult-cancel-left-if}) simp also have \( \ldots = \sqrt{?pqdb} * (\sqrt{?pqdb} + 1 / (\sqrt{?pqda})) / (\sqrt{?pqdb} * (\sqrt{?pqdb} + 1 / (\sqrt{?pqdb})) \) by simp also from \( ?pqdb > 0 \) have \( \ldots = (\sqrt{?pqdb} * ?pqda + \sqrt{?pqdb} / ?pqda) / (\sqrt{?pqdb} + 1) \) by (simp add: real-sqrt-mult [symmetric] real-sqrt-divide algebra-simps) also from \( ?pqdb = ?pqda * ?pqab \) and \( ?pqda > 0 \) and \( \text{real-sqrt-pow2} \) have \( \ldots = (?pqda * \sqrt{?pqab} + \sqrt{?pqab}) / (?pqda * ?pqab + 1) \) by (simp add: real-sqrt-mult power2-eq-square) finally have \( ?ac / ?bc = (?pqda * \sqrt{?pqab} + \sqrt{?pqab}) / (?pqda * ?pqab + 1) \) .
from \( \langle ?pqda > 0 \rangle \) and \( \langle ?pqab > 0 \rangle \)

have \( ?pqda * ?pqab + 1 > 0 \) by (simp add: add-pos-pos)

with \( \langle ?bc > 0 \rangle \)

and \( \langle ?ac / ?bc = ( ?pqda * sqrt ?pqab + sqrt ?pqab ) / ( ?pqda * ?pqab + 1 ) \rangle \)

have \( ?ac * ( ?pqda * ?pqab + 1 ) = ?bc * ( ?pqda * sqrt ?pqab + sqrt ?pqab ) \)

by (simp add: field-simps)


by (simp add: algebra-simps)

from \( \langle \text{proj2-set-Col} \{ p,q,a,b \} \rangle \)

and \( \langle p \neq q \rangle \) and \( \langle a \neq p \rangle \) and \( \langle a \neq q \rangle \)

and \( \langle b \neq p \rangle \)

have cross-ratio-correct \( p \ q \ a \ b \) by (unfold cross-ratio-correct-def) simp

have \( ?ac * ?pqab - ?bc * sqrt ?pqab \neq 0 \)

proof

assume \( ?ac * ?pqab - ?bc * sqrt ?pqab = 0 \)


have \( ?bc * sqrt ?pqab - ?ac = 0 \) by simp

with \( ?ac * ?pqab - ?bc * sqrt ?pqab = 0 \) and \( \langle ?ac > 0 \rangle \)

have \( ?pqab = 1 \) by simp

with \( \langle \text{cross-ratio-correct} \ p \ q \ a \ b \rangle \)

have \( a = b \) by (rule cross-ratio-1-equal)

with \( \langle a \neq b \rangle \) show False ..

qed


by (simp add: field-simps)

qed

lemma perp-foot-cross-ratio-formula:

assumes \( a \in \text{hyp2} \) and \( b \in \text{hyp2} \) and \( c \in \text{hyp2} \) and \( a \neq b \)

shows cross-ratio \( ( \text{endpoint-in-S} \ a \ b ) \) \( ( \text{endpoint-in-S} \ b \ a ) \)

\( ( \text{perp-foot} \ c \ ( \text{proj2-line-through} \ a \ b ) ) \ a \)

\( = ( \text{cosh-dist} \ b \ c * sqrt \ ( \text{exp-2dist} \ a \ b ) - \text{cosh-dist} \ a \ c ) \)

\( / ( \text{cosh-dist} \ a \ c * \text{exp-2dist} \ a \ b - \text{cosh-dist} \ b \ c * sqrt \ ( \text{exp-2dist} \ a \ b ) ) \)

(is cross-ratio \( ?p \ ?q \ ?d \ a \)


proof

from \( \langle a \neq b \rangle \) and \( \langle a \in \text{hyp2} \rangle \) and \( \langle b \in \text{hyp2} \rangle \)

have \( \text{are-endpoints-in-S} \ ?p \ ?q \ a \ b \)

by (rule endpoints-in-S-are-endpoints-in-S)

let \( \ell = \text{proj2-line-through} \ a \ b \)

have \( \text{proj2-incident} \ a \ ?l \) and \( \text{proj2-incident} \ b \ ?l \)

by (rule proj2-line-through-incident)

with \( \langle a \neq b \rangle \) and \( \langle a \in \text{hyp2} \rangle \) and \( \langle b \in \text{hyp2} \rangle \)

have \( \text{proj2-incident} \ ?p \ ?l \) and \( \text{proj2-incident} \ ?q \ ?l \)

by (simp-all add: endpoint-in-S-incident)
let \(?m = \text{drop-perp} c \ ?l\)

have \(\text{M-perp} \ ?l \ ?m\) by (rule drop-perp-perp)

have proj2-incident \(?d \ ?l\) and proj2-incident \(?d \ ?m\)
  by (rule perp-foot-incident)

have proj2-incident \(?m\) by (rule drop-perp-incident)

with \(\{a \neq b\}\) and \(\{c \in \text{hyp2}\}\) and \(\{\text{are-endpoints-in-S} ?p ?q a b\}\)

have proj2-incident \(?p ?l\) and proj2-incident \(?q ?l\) and \(\{\text{M-perp} ?l ?m\}\)

have proj2-incident \(?p ?l\) and proj2-incident \(?q ?l\) and \(\{\text{M-perp} ?l ?m\}\)

have proj2-incident \(?d ?l\) and proj2-incident \(?d ?m\) by (rule perp-foot-incident)+

have proj2-incident \(?d ?l\) and proj2-incident \(?d ?m\)
  by (rule perp-foot-incident)

have \(\text{cross-ratio} ?p ?q ?d a \) = \((?bc * \sqrt{\text{cross-ratio} ?p ?q a b} - ?ac) / (?ac * (\text{cross-ratio} ?p ?q a b) - ?bc * \sqrt{\text{cross-ratio} ?p ?q a b}))\)
  by (rule described-perp-foot-cross-ratio-formula)

with \(\{a \neq b\}\)

show \(\text{cross-ratio} ?p ?q ?d a \) = \((?bc * \sqrt{?pqab} - ?ac) / (?ac * ?pqab - ?bc * \sqrt{?pqab})\)
  by (unfold exp-2dist-def) simp

qed

8.12 The Klein–Beltrami model satisfies axiom 5

lemma statement 69:
  assumes \(a b \equiv_K a’ b’\) and \(b c \equiv_K b’ c’\) and \(a c \equiv_K a’ c’\)
  shows \(\exists J. \text{is-K2-isometry J} \wedge \text{hyp2-cltn2} a J = a’ \wedge \text{hyp2-cltn2} b J = b’ \wedge \text{hyp2-cltn2} c J = c’\)

proof cases
  assume \(a = b\)
  with \(\{a b \equiv_K a’ b’\}\) have \(a’ = b’\) by (simp add: hyp2.A3-reversed)
  with \(\{a = b\}\) and \(\{b c \equiv_K b’ c’\}\)
  show \(\exists J. \text{is-K2-isometry J} \wedge \text{hyp2-cltn2} a J = a’ \wedge \text{hyp2-cltn2} b J = b’ \wedge \text{hyp2-cltn2} c J = c’\)
    by (unfold real-hyp2-C-def) simp

next
  assume \(a \neq b\)
  with \(\{a b \equiv_K a’ b’\}\)
  have \(a’ \neq b’\) by (auto simp add: hyp2.A3’)

let \(?pa = \text{Rep-hyp2} a\)
  and \(?pb = \text{Rep-hyp2} b\)
  and \(?pc = \text{Rep-hyp2} c\)
  and \(?pa’ = \text{Rep-hyp2} a’\)
  and \(?pb’ = \text{Rep-hyp2} b’\)
  and \(?pc’ = \text{Rep-hyp2} c’\)

define \(pp \ pq \ pp’ \ pq’ \ l’\)
  where \(pp = \text{endpoint-in-S} ?pa ?pb\)
  and \(pq = \text{endpoint-in-S} ?pb ?pa\)
  and \(l = \text{proj2-line-through} ?pa ?pb\)
and \( pp' = \text{endpoint-in-S} \ ?pa' \ ?pb' \)
and \( pq' = \text{endpoint-in-S} \ ?pb' \ ?pa' \)
and \( l' = \text{proj2-line-through} \ ?pa' \ ?pb' \)
define \( pd \ ps \ m \ pd' \ ps' \ m' \)
where \( pd = \text{perp-foot} \ ?pc \ l \)
and \( ps = \text{perp-up} \ ?pc \ l \)
and \( m = \text{drop-perp} \ ?pc \ l \)
and \( pd' = \text{perp-foot} \ ?pc' \ l' \)
and \( ps' = \text{perp-up} \ ?pc' \ l' \)
and \( m' = \text{drop-perp} \ ?pc' \ l' \)

have \( pp \in S \) and \( pp' \in S \) and \( pq \in S \) and \( pq' \in S \)
unfolding \( pp\text{-def} \) and \( pp'\text{-def} \) and \( pq\text{-def} \) and \( pq'\text{-def} \)
by \((\text{simp-all add: Rep-hyp2 endpoint-in-S})\)

from \( \langle a \neq b \rangle \) and \( \langle a' \neq b' \rangle \)
have \( ?pa \neq ?pb \) and \( ?pa' \neq ?pb' \) by \((\text{unfold Rep-hyp2-inject})\)

moreover
have \( \text{proj2-incident} \ ?pa \ l \) and \( \text{proj2-incident} \ ?pb \ l \)
and \( \text{proj2-incident} \ ?pa' \ l' \) and \( \text{proj2-incident} \ ?pb' \ l' \)
by \((\text{unfold l-def l'-def}) \) (rule \( \text{proj2-line-through-incident} \))
ultimately have \( \text{proj2-incident} \ ?pa \ l \) and \( \text{proj2-incident} \ ?pb \ l \)
and \( \text{proj2-incident} \ ?pa' \ l' \) and \( \text{proj2-incident} \ ?pb' \ l' \)
unfolding \( pp\text{-def} \) and \( pp'\text{-def} \) and \( pq\text{-def} \) and \( pq'\text{-def} \)
by \((\text{simp-all add: Rep-hyp2 endpoint-in-S-incident})\)

from \( \langle pp \in S \rangle \) and \( \langle pp' \in S \rangle \) and \( \langle \text{proj2-incident} \ ?pa \ l \rangle \)
and \( \langle \text{proj2-incident} \ ?pa' \ l' \rangle \) and \( \langle \text{proj2-incident} \ ?pb \ l \rangle \)
and \( \langle \text{proj2-incident} \ ?pb' \ l' \rangle \)

have \( \text{right-angle} \ ?pp \ ?pd \ ?ps \) and \( \text{right-angle} \ ?pp' \ ?pd' \ ?ps' \)
unfolding \( pd\text{-def} \) and \( ps\text{-def} \) and \( pd'\text{-def} \) and \( ps'\text{-def} \)
by \((\text{simp-all add: Rep-hyp2 \ perp-foot-up-right-angle \ [of} \ ?pp \ ?pc \ ?pa \ l]) \)

with \( \text{right-angle-to-right-angle} \ [\text{of} \ ?pp \ ?pd \ ?pp' \ ?pd' \ ?ps \ ?ps'] \)

obtain \( J \) where \( \text{is-K2-isometry} \ ?J \) and \( \text{apply-cltn2} \ ?pd J = ?pp' \)
and \( \text{apply-cltn2} \ ?ps J = ?ps' \)
by \( \text{auto} \)

let \( ?paJ = \text{apply-cltn2} \ ?pa J \)
and \( ?pbJ = \text{apply-cltn2} \ ?pb J \)
and \( ?pcJ = \text{apply-cltn2} \ ?pc J \)
and \( ?pdJ = \text{apply-cltn2} \ ?pd J \)
and \( ?ppJ = \text{apply-cltn2} \ ?pp J \)
and \( ?pqJ = \text{apply-cltn2} \ ?pq J \)
and \( ?psJ = \text{apply-cltn2} \ ?ps J \)
and \( ?lJ = \text{apply-cltn2-line} \ ?l J \)
and \( ?mJ = \text{apply-cltn2-line} \ ?m J \)
have proj2-incident pd l and proj2-incident pd' l'
and proj2-incident pd m and proj2-incident pd' m'
by (unfold pd-def pd'-def m-def m'-def) (rule perp-foot-incident)+

from (proj2-incident pp l) and (proj2-incident pq l)
and (proj2-incident pd l) and (proj2-incident ?pa l)
and (proj2-incident ?pb l)
have proj2-set-Col {pp, pq, pd, ?pa} and proj2-set-Col {pp, pq, ?pa, ?pb}
by (unfold pd-def proj2-set-Col-def) (simp-all add: exI [of - l])

from (?pa \neq ?pb) and (?pa' \neq ?pb')
have pp \neq pq and pp' \neq pq'
unfolding pp-def and pq-def and pp'-def and pq'-def
by (simp-all add: Rep-hyp2 endpoint-in-S-swap)

from (proj2-incident ?pa l) and (proj2-incident ?pa' l')
have pd \in hyp2 and pd' \in hyp2
unfolding pd-def and pd'-def
by (simp-all add: Rep-hyp2 perp-foot-hyp2 [of ?pa l ?pc]
perp-foot-hyp2 [of ?pa' l' ?pc'])

from (proj2-incident ?pa l) and (proj2-incident ?pa' l')
have ps \in S and ps' \in S
unfolding ps-def and ps'-def
by (simp-all add: Rep-hyp2 perp-up-in-S [of ?pc ?pa l]
perp-up-in-S [of ?pc' ?pa' l'])

from (pd \in hyp2) and (pp \in S) and (ps \in S)
have pd \neq pp and ?pa \neq pp and ?pb \neq pp and pd \neq ps
by (simp-all add: Rep-hyp2 hyp2-S-not-equal)

from (is-K2-isometry J) and (pq \in S)
have ?pqJ \in S by (unfold is-K2-isometry-def) simp

from (pd \neq pp) and (proj2-incident pd l) and (proj2-incident pp l)
and (proj2-incident pd' l') and (proj2-incident pp' l')
have ?lj = l'
unfolding (?pdJ = pd' [symmetric] and ?ppJ = pp' [symmetric]
by (rule apply-cltn2-line-unique)
from (proj2-incident pq l) and (proj2-incident ?pa l)
and (proj2-incident ?pb l)
have proj2-incident ?pqJ l' and proj2-incident ?paJ l'
and proj2-incident ?pbJ l'
by (unfold (?lj = l' [symmetric]) simp-all)

from (?pa' \neq ?pb') and (?pqJ \in S) and (proj2-incident ?pa' l')
and (proj2-incident ?pb' l') and (proj2-incident ?pqJ l')
have ?pqJ = pp' \lor ?pqJ = pq'
unfolding pp'-def and pq'-def

212
have endpoints-in-S pp
are-endpoints-in-S pp

have cross-ratio pp' pq' pd' ?pa'
unfolding pp'-def and pq'-def and pd'-def and l'-def
by (simp add: Rep-hyp2 perp-foot-cross-ratio-formula)
also from assms
have ... = cross-ratio pp pq pd ?pa
unfolding pp-def and pq-def and pd-def and l-def
by (simp add: Rep-hyp2 perp-foot-cross-ratio-formula)
also from (proj2-set-Col {pp,pq,pd,?pa}) and (pp ≠ pq) and (pd ≠ pp)
and (?pa ≠ pp)
have ... = cross-ratio ?pp ?pq ?pd ?paJ by (simp add: cross-ratio-cltn2)
also from (?pp = pp' and (?pqJ = pq') and (?pdJ = pd')
have ... = cross-ratio pp' pq' pd' ?paJ by simp
finally
have cross-ratio pp' pq' pd' ?paJ = cross-ratio pp' pq' pd' ?pa' by simp

from (is-K2-isometry J)
have ?paJ ∈ hyp2 and ?pbJ ∈ hyp2 and ?pcJ ∈ hyp2
by (rule apply-cltn2-Rep-hyp2)+

from (proj2-incident pp' l') and (proj2-incident pq' l')
and (proj2-incident pd' l') and (proj2-incident ?paJ l')
and (proj2-incident ?pa' l') and (proj2-incident ?pbJ l')
and (proj2-incident ?pb' l')
have proj2-set-Col {pp',pq',pd',?paJ} and proj2-set-Col {pp',pq',pd',?pa'}
and proj2-set-Col {pp',pq',?pa',?pbJ}
and proj2-set-Col {pp',pq',?pa',?pb'}
by (unfold proj2-set-Col-def) (simp-all add: exI [of - l'])
with (?pp' ≠ pp') and (?pp' ∈ S) and (?pq' ∈ S) and (?pd' ∈ hyp2)
and (?paJ ∈ hyp2) and (?pbJ ∈ hyp2)
have arc-endpoints-in-S pp' pq' pd' ?paJ
and arc-endpoints-in-S pp' pq' pd' ?pa'
and arc-endpoints-in-S pp' pq' ?pa' ?pbJ
and arc-endpoints-in-S pp' pq' ?pa' ?pb'
by (unfold arc-endpoints-in-S-def) (simp-all add: Rep-hyp2)

213
hence cross-ratio-correct pp' pq' pd' ?paJ
and cross-ratio-correct pp' pq' pd' ?pa'
and cross-ratio-correct pp' pq' ?pa' ?pbJ
and cross-ratio-correct pp' pq' ?pa' ?pb'
by (simp-all add: are-ends-points-in-S-cross-ratio-correct)

from (cross-ratio-correct pp' pq' pd' ?paJ)
and (cross-ratio-correct pp' pq' pd' ?pa')
and (cross-ratio pp' pq' pd' ?paJ = cross-ratio pp' pq' pd' ?pa')
have ?paJ = ?pa' by (simp add: cross-ratio-unique)
with (?ppJ = pp'' and ?pqJ = pq'')
also from (proj2-set-Col {pp,pq,?pa,?pb}) and ?pp ≠ ?pq and (?pa ≠ ?pp)
and (?pb ≠ pp)
have ... = cross-ratio pp pq ?pa ?pb by (rule cross-ratio-cltn2)
also from (a ≠ b) and (a b ≡K a' b')
have ... = cross-ratio pp' pq' ?pa' ?pb'
unfolding pp-def pq-def pp''-def pq''-def
by (rule real-hyp2-C-cross-ratio-endpoints-in-S)
finally have cross-ratio pp' pq' ?pa' ?pbJ = cross-ratio pp' pq' ?pa' ?pb'.
with (cross-ratio-correct pp' pq' ?pa' ?pbJ)
and (cross-ratio-correct pp' pq' ?pa' ?pb'J)
have ?pbJ = ?pb' by (rule cross-ratio-unique)

let ?cc = cart2-pt ?pc
and ?cd = cart2-pt pd
and ?cs = cart2-pt ps
and ?cc' = cart2-pt ?pc'
and ?cd' = cart2-pt pd'
and ?cs' = cart2-pt ps'
and ?ccJ = cart2-pt ?pcJ
and ?cdJ = cart2-pt ?pdJ
and ?csJ = cart2-pt ?psJ

from (proj2-incident ?pa l) and (proj2-incident ?pa' l')
unfolding pd-def and ps-def and pd'-def and ps'-def

from (pd ∈ hyp2) and (ps ∈ S) and (is-K2-isometry J)
and (B?R ?cd ?cc)

also from (is-K2-isometry J)
have ... = cosh-dist ?pa ?pc by (simp add: Rep-hyp2 K2-isometry-cosh-dist)
also from (a c ≡K a' c')

214
have \( \ldots = \cosh\text{-dist } \?pa\' \?pc\' \) by (rule real-hyp2-C-cosh-dist)


have M-perp l' m' by (unfold m'-def) (rule drop-perp-perp)

have proj2-incident ?pm m and proj2-incident ?pc' m'
  by (unfold m-def m'-def) (rule drop-perp-incident)

from (proj2-incident ?pa b) and (proj2-incident ?pa' l')
have proj2-incident ps m and proj2-incident ps' m'
  unfolding ps-def and m-def and ps'-def and m'-def
  by (simp-all add: Rep-hyp2 perp-up-incident [of ?pc ?pa l]
    perp-up-incident [of ?pc' ?pa' l'])

with (pd \( \neq \) ps) and (proj2-incident pd m) and (proj2-incident pd' m')
have ?mJ = m'
  unfolding ?pdJ = pd' [symmetric] and ?psJ = ps' [symmetric]
  by (simp add: apply-cltn2-line-unique)

from (proj2-incident ?pc m)
have proj2-incident ?pcJ m' by (unfold \( \exists \)mJ = m' [symmetric]) simp

with (M-perp l' m') and Rep-hyp2 [of a'] and (pd' \( \in \) hyp2) and (\( \forall \)pcJ \( \in \) hyp2)
  and Rep-hyp2 [of c] and (proj2-incident ?pa' l')
  and (proj2-incident pd' l') and (proj2-incident pd' m')
  and (proj2-incident ?pc' m')
have cosh-dist pd' ?pcJ = cosh-dist ?pa' ?pcJ / cosh-dist pd' ?pa'
  and cosh-dist pd' ?pc' = cosh-dist ?pa' ?pc' / cosh-dist pd' ?pa'
  by (simp-all add: cosh-dist-perp-divide)

have cosh-dist pd' ?pcJ = cosh-dist pd' ?pc' by simp

with (pd' \( \in \) hyp2) and (\( \forall \)pcJ \( \in \) hyp2) and (\( \forall \)pc' \( \in \) hyp2) and (ps' \( \in \) S)
  and (\( \forall \)B'R \( \exists \)cd' \( \in \) hyp2) and (\( \forall \)pc' \( \in \) hyp2)
  and (\( \exists \)B'R \( \exists \)cd' \( \exists \)cs')
have ?pcJ = ?pc' by (rule cosh-dist-unique)

with (?pdJ = ?pa' and (?pdJ = ?pa'))
have hyp2-cltn2 b J = a' and hyp2-cltn2 b J = b' and hyp2-cltn2 c J = c'
  by (unfold hyp2-cltn2-def) (simp-all add: Rep-hyp2-inverse)

with (is-K2-isometry J)
show \( \exists \) J. is-K2-isometry J
  \& hyp2-cltn2 a J = a' \& hyp2-cltn2 b J = b' \& hyp2-cltn2 c J = c'
  by (simp add: exI [of - J])

qed

**Theorem hyp2-axiom5:**
\[ \forall a b c d a' b' c' d'. \]

\[ a \neq b \land B\text{K} \; a b c \land B\text{K} \; a' b' c' \land a b \equiv \text{K} a' b' \land b c \equiv \text{K} b' c' \]
\[ \land a d \equiv \text{K} a' d' \land b d \equiv \text{K} b' d' \]
\[ \rightarrow c d \equiv \text{K} c' d' \]

**Proof**
standard

fix a b c d a' b' c' d'
assume a \( \neq \) b \& B\text{K} \; a b c \& B\text{K} \; a' b' c' \& a b \equiv \text{K} a' b' \& b c \equiv \text{K} b' c' \\
\land a d \equiv \text{K} a' d' \land b d \equiv \text{K} b' d'
hence \(a \neq b\) and \(B_K a b c\) and \(B_K a' b' c'\) and \(a b \equiv_K a' b'\) and \(b c \equiv_K b' c'\) and \(a d \equiv_K a' d'\) and \(b d \equiv_K b' d'\) by simp-all from \(\langle a b \equiv_K a' b' \rangle\) and \(\langle a d \equiv_K a' d' \rangle\) and \(\langle b c \equiv_K b' c' \rangle\) and \(\langle b d \equiv_K b' d' \rangle\) and statement69

obtain \(J\) where is-K2-isometry \(J\) and hyp2-cltn2 \(a J = a'\) and hyp2-cltn2 \(b J = b'\) and hyp2-cltn2 \(d J = d'\) by auto let ?aJ = hyp2-cltn2 \(a J\) and ?bJ = hyp2-cltn2 \(b J\) and ?cJ = hyp2-cltn2 \(c J\) and ?dJ = hyp2-cltn2 \(d J\) from \(\langle a \neq b \rangle\) and \(\langle a b \equiv_K a' b' \rangle\) have \(a' \neq b'\) by (auto simp add: hyp2.A3') from \(\langle is-K2-isometry J \rangle\) and \(\langle B_K a b c \rangle\) have \(B_K ?aJ ?bJ ?cJ\) by (rule real-hyp2-B-hyp2-cltn2) hence \(B_K a' b' ?cJ\) by (unfold ⟨?aJ = a'⟩ ⟨?bJ = b'⟩)

from \(\langle is-K2-isometry J \rangle\) have \(b c \equiv_K b' c'\) by (rule real-hyp2-C-hyp2-cltn2) hence \(b c \equiv_K b' c'\) by (rule hyp2-extend-segment-unique) unfolding ⟨?cJ = c'⟩ [symmetric] and ⟨?dJ = d'⟩ [symmetric] by (rule real-hyp2-C-hyp2-cltn2)

qed

interpretation hyp2: tarski-first5 real-hyp2-C real-hyp2-B using hyp2-axiom4 and hyp2-axiom5 by unfold-locales

8.13 The Klein–Beltrami model satisfies axioms 6, 7, and 11

theorem hyp2-axiom6: \(\forall a b. \ B_K a b a \rightarrow a = b\)

proof standard+

fix \(a b\)

let ?ca = cart2-pt (Rep-hyp2 \(a\)) and ?cb = cart2-pt (Rep-hyp2 \(b\))

assume \(B_K a b a\)

hence \(B_K ?ca ?cb ?ca\) by (unfold real-hyp2-B-def hyp2-rep-def)

hence \(?ca = ?cb\) by (rule real-euclid.A6') hence Rep-hyp2 \(a = Rep-hyp2 b\) by (simp add: Rep-hyp2 hyp2-S-cart2-inj)
thus \( a = b \) by (unfold \textit{Rep-hyp2-inject})

\( \text{qed} \)

\textbf{lemma} \textit{between-inverse}:
\textbf{assumes} \( B_{R_1} (\text{hyp2-rep} \ p) \ v \ (\text{hyp2-rep} \ q) \)
\textbf{shows} \( \text{hyp2-rep} \ (\text{hyp2-abs} \ v) = v \)
\textbf{proof} –
\textbf{let} \( ?u = \text{hyp2-rep} \ p \)
\textbf{let} \( ?w = \text{hyp2-rep} \ q \)
\textbf{have} \( \text{norm} \ ?u < 1 \ \text{and} \ \text{norm} \ ?w < 1 \) by (rule \textit{norm-hyp2-rep-lt-1})
\textbf{from} \( \langle B \ I \ R \ ?u \ v \ ?w \rangle \)
\textbf{obtain} \( l \) \textbf{where} \( l \geq 0 \ \text{and} \ l \leq 1 \ \text{and} \ v - ?u = l \ast_R (\ ?w - ?u) \)
\textbf{by} (unfold \textit{real-euclid-B-def}) auto
\textbf{from} \( \langle v - ?u = l \ast_R (\ ?w - ?u) \rangle \)
\textbf{have} \( \text{norm} \ v \leq \text{norm} \ (l \ast_R ?w) + \text{norm} \ ((1 - l) \ast_R ?u) \)
\textbf{by} \( \textit{simp only: norm-triangle-ineq} [\textit{of} \ l \ast_R \ ?w \ (1 - l) \ast_R \ ?u] \)
\textbf{with} \( \langle l \geq 0 \rangle \ \text{and} \ (l \leq 1) \)
\textbf{have} \( \text{norm} \ v \leq l \ast_R \text{norm} \ ?w + (1 - l) \ast_R \text{norm} \ ?u \) by simp

\textbf{have} \( \text{norm} \ v < 1 \)
\textbf{proof} cases
\textbf{assume} \( l = 0 \)
\textbf{with} \( \langle \text{norm} \ ?w < 1 \rangle \ \text{and} \ (l \geq 0) \) \textbf{have} \( l \ast_R \text{norm} \ ?w < l \) by simp

\textbf{with} \( \langle \text{norm} \ ?u < 1 \rangle \ \text{and} \ (l \leq 1) \)
\textbf{and} \( \textit{mult-mono} [\textit{of} \ 1 - l \ast_C l \ast_C \text{norm} \ ?u \ 1] \)
\textbf{have} \( (1 - l) \ast_R \text{norm} \ ?u \leq 1 - l \) by simp
\textbf{with} \( (l \ast_R \text{norm} \ ?w < l) \)
\textbf{have} \( l \ast_R \text{norm} \ ?w + (1 - l) \ast_R \text{norm} \ ?u < 1 \) by simp
\textbf{with} \( \langle \text{norm} \ v \leq l \ast_R \text{norm} \ ?w + (1 - l) \ast_R \text{norm} \ ?u \rangle \)
\textbf{show} \( \text{norm} \ v < 1 \) by simp
\textbf{qed}

\textbf{thus} \( \text{hyp2-rep} \ (\text{hyp2-abs} \ v) = v \) by (rule \textit{hyp2-rep-abs})
\textbf{qed}

\textbf{lemma} \textit{between-switch}:
\textbf{assumes} \( B_{R_1} (\text{hyp2-rep} \ p) \ v \ (\text{hyp2-rep} \ q) \)
\textbf{shows} \( B_{K_1} (\text{hyp2-abs} \ p) \ q \)
\textbf{proof} –
\textbf{from} \( \text{assms} \) \textbf{have} \( \text{hyp2-rep} \ (\text{hyp2-abs} \ v) = v \) by (rule \textit{between-inverse})
\textbf{with} \( \text{assms} \) \textbf{show} \( B_{K_1} (\text{hyp2-abs} \ p) \ q \) by (unfold \textit{real-hyp2-B-def}) simp
\textbf{qed}

\[ \]
theorem hyp2-axiom7:
\( \forall a \ b \ c \ p \ q. \ B_K a p c \land B_K b q c \rightarrow (\exists x. B_K p x b \land B_K q x a) \)

proof
fix a b c p q
let ?ca = hyp2-rep a
and ?cb = hyp2-rep b
and ?cc = hyp2-rep c
and ?cp = hyp2-rep p
and ?cq = hyp2-rep q
assume \( B_K a p c \) and \( B_K b q c \)
hence \( B_I R ?ca ?cp ?cc \) and \( B_I R ?cb ?cq ?cc \) by (unfold real-hyp2-B-def)
obtain cx where \( B_I R ?cp cx ?cb \) and \( B_I R ?cq cx ?ca \) by auto
hence \( B_K p (hyp2-abs cx) b \) and \( B_K q (hyp2-abs cx) a \) by (simp-all add: between-switch)
thus \( \exists x. B_K p x b \land B_K q x a \) by (simp add: exI [of - hyp2-abs cx])
qed

theorem hyp2-axiom11:
\( \forall X \ Y. (\exists a, \forall x, y, x \in X \land y \in Y \rightarrow B_K a x y) \)
\( \rightarrow (\exists b, \forall x, y, x \in X \land y \in Y \rightarrow B_K x b y) \)

proof (rule allI)+
fix X Y :: hyp2 set
show (\( \exists a, \forall x, y, x \in X \land y \in Y \rightarrow B_K a x y) \)
\( \rightarrow (\exists b, \forall x, y, x \in X \land y \in Y \rightarrow B_K x b y) \)
proof
cases
assume \( X = {} \lor Y = {} \)
thus (\( \exists a, \forall x, y, x \in X \land y \in Y \rightarrow B_K a x y) \)
\( \rightarrow (\exists b, \forall x, y, x \in X \land y \in Y \rightarrow B_K x b y) \) by auto
next
assume \( \neg (X = {} \lor Y = {}) \)
hence \( X \neq {} \) and \( Y \neq {} \) by simp-all
then obtain w and z where \( w \in X \and z \in Y \) by auto

show (\( \exists a, \forall x, y, x \in X \land y \in Y \rightarrow B_K a x y) \)
\( \rightarrow (\exists b, \forall x, y, x \in X \land y \in Y \rightarrow B_K x b y) \)
proof
assume \( \exists a, \forall x, y, x \in X \land y \in Y \rightarrow B_K a x y \)
then obtain a where \( \forall x, y, x \in X \land y \in Y \rightarrow B_K a x y \)

let \( ?cX = hyp2-rep ' X \)
and \( ?cY = hyp2-rep ' Y \)
and \( ?ca = hyp2-rep a \)
and \( ?cw = hyp2-rep w \)
and \( ?cz = hyp2-rep z \)

from \( \forall x, y, x \in X \land y \in Y \rightarrow B_K a x y \)
have \( \forall cx cy. cx \in ?cX \land cy \in ?cY \rightarrow B_R ?ca cx cy \)
by (unfold real-hyp2-B-def) auto
obtain cb where \( \forall \ cx \ cy. \ cx \in ?cX \land cy \in ?cY \rightarrow B_R cx cb cy \) by auto
with (\( w \in X \)) and (\( z \in Y \)) have \( B_R \ ?cw cb \ ?cz \) by simp
hence hyp2-rep (hyp2-abs cb) = cb (is hyp2-rep ?b = cb)
  by (rule between-inverse)
with (\( \forall \ cx \ cy. \ cx \in ?cX \land cy \in ?cY \rightarrow B_R cx cb cy \))
have \( \exists \ b. \ \forall \ x \ y. \ x \in X \land y \in Y \rightarrow B_K x \ b \ y \)
  by (unfold real-hyp2-B-def) simp
thus \( \exists \ b. \ \forall \ x \ y. \ x \in X \land y \in Y \rightarrow B_K x \ b \ y \) by (rule exI)
qed
qed
qed
interpretation tarski-absolute-space real-hyp2-C real-hyp2-B
using hyp2-axiom6 and hyp2-axiom7 and hyp2-axiom11
by unfold-locales

8.14 The Klein–Beltrami model satisfies the dimension-specific axioms

lemma hyp2-rep-abs-examples:
  shows hyp2-rep (hyp2-abs 0) = 0 (is hyp2-rep ?a = ?ca)
  and hyp2-rep (hyp2-abs (vector [1/2,0])) = vector [1/2,0]
  (is hyp2-rep ?b = ?cb)
  and hyp2-rep (hyp2-abs (vector [0,1/2])) = vector [0,1/2]
  (is hyp2-rep ?c = ?cc)
  and hyp2-rep (hyp2-abs (vector [1/4,1/4])) = vector [1/4,1/4]
  (is hyp2-rep ?d = ?cd)
  and hyp2-rep (hyp2-abs (vector [1/2,1/2])) = vector [1/2,1/2]
  (is hyp2-rep ?t = ?ct)
proof –
  have norm ?ca < 1 and norm ?cb < 1 and norm ?cc < 1 and norm ?cd < 1
  and norm ?ct < 1
    by (unfold norm-vec-def L2-set-def) (simp-all add: sum-2 power2-eq-square)
    by (simp-all add: hyp2-rep-obs)
qed

theorem hyp2-axiom8: \( \exists \ a \ b \ c. \ \lnot B_K a \ b \ c \land \lnot B_K b \ c \ a \land \lnot B_K c \ a \ b \)
proof –
  let ?ca = 0 :: real^2
  and ?cb = vector [1/2,0] :: real^2
  and ?cc = vector [0,1/2] :: real^2
  let ?a = hyp2-abs ?ca
  and ?b = hyp2-abs ?cb
  and ?c = hyp2-abs ?cc
from hyp2-rep-abs-examples and non-Col-example
have \( \neg (\text{hyp2}.\text{Col} \ ?a \ ?b \ ?c) \)
by \((\text{unfold hyp2}.\text{Col-def}\)\real-euclid.\Col-def \ \text{real-hyp2-B-def})\ simp
thus \( \exists \ ?a \ ?b \ ?c . \neg B_K \ ?a \ ?b \ ?c \wedge \neg B_K \ ?b \ ?c \ ?a \wedge \neg B_K \ ?c \ ?a \ ?b \)
unfolding hyp2.\Col-def
by simp (rule \exI)+
qed

theorem hyp2-axiom9:
\[ \forall \ ?p \ ?q \ ?a \ ?b \ ?c . \ ?p \neq \ ?q \wedge \ ?a \equiv_{K} \ ?p \ ?q \wedge \ ?b \equiv_{K} \ ?b \ ?q \wedge \ ?c \equiv_{K} \ ?c \ ?q \]
\[ \rightarrow B_K \ ?a \ ?b \ ?c \ \vee \ B_K \ ?b \ ?c \ ?a \ \vee \ B_K \ ?c \ ?a \ ?b \]
proof (rule allI)+
fix \ ?p \ ?q \ ?a \ ?b \ ?c 
show \ ?p \neq \ ?q \wedge \ ?a \equiv_{K} \ ?p \ ?q \wedge \ ?b \equiv_{K} \ ?b \ ?q \wedge \ ?c \equiv_{K} \ ?c \ ?q 
\rightarrow B_K \ ?a \ ?b \ ?c \ \vee \ B_K \ ?b \ ?c \ ?a \ \vee \ B_K \ ?c \ ?a \ ?b 
proof 
assume \ ?p \neq \ ?q \wedge \ ?a \equiv_{K} \ ?p \ ?q \wedge \ ?b \equiv_{K} \ ?b \ ?q \wedge \ ?c \equiv_{K} \ ?c \ ?q 
hence \ ?p \neq \ ?q \ \wedge \ ?a \equiv_{K} \ ?p \ ?q \ \wedge \ ?b \equiv_{K} \ ?b \ ?q \ \wedge \ ?c \equiv_{K} \ ?c \ ?q \ \text{by simp-all} 

let \ ?pp = \text{Rep-hyp2} \ ?p 
and \ ?pq = \text{Rep-hyp2} \ ?q 
and \ ?pa = \text{Rep-hyp2} \ ?a 
and \ ?pb = \text{Rep-hyp2} \ ?b 
and \ ?pc = \text{Rep-hyp2} \ ?c 
define \ l \ \text{where} \ l = \text{proj2-line-through} \ ?pp \ ?pq 
define \ m \ ?ps \ ?pt \ ?stpq 
where \ m = \text{drop-perp} \ ?pa \ ?l 
and \ ?ps = \text{endpoint-in-S} \ ?pp \ ?pq 
and \ ?pt = \text{endpoint-in-S} \ ?pq \ ?pp 
and \ ?stpq = \text{exp-2dist} \ ?pp \ ?pq 
from \ ?p \neq \ ?q \ \text{have} \ ?pp \neq \ ?pq \ \text{by simp add: Rep-hyp2-inject} 

from Rep-hyp2 
\text{have} \ ?stpq > 0 \ \text{by (unfold \ ?stpq-def)} \ (\text{simp add: exp-2dist-positive}) 
\text{hence} \ \sqrt{\ ?stpq} \cdot \sqrt{\ ?stpq} = \ ?stpq 
by (\text{simp add: real-sqrt-mult \ [symmetric]}) 
from Rep-hyp2 \ \text{and} \ (\ ?pp \neq \ ?pq) 
\text{have} \ ?stpq \neq 1 \ \text{by (unfold \ ?stpq-def)} \ (\text{auto simp add: exp-2dist-1-equal}) 

\text{have} \ ?p \ ?a \ \text{and} \ ?p \ ?b \ \text{and} \ ?p \ ?c \ \text{by simp-all add: Rep-hyp2 hyp2-S-z-non-zero} 

\text{have} \ \forall \ ?pd \in \ {?pa,?pb,?pc} . 
\text{cross-ratio} \ ?ps \ ?pt \ (\text{perp-foot} \ ?pd \ ?l) \ ?pp = 1 / (\sqrt{\ ?stpq}) 
proof 
fix \ ?pd 
\text{assume} \ ?pd \in \ {?pa,?pb,?pc} 
with Rep-hyp2 \ \text{have} \ ?pd \in \ hyp2 \ \text{by auto} 

220
define pe x
  where pe = perp-foot pd l
  and x = cosh-dist ?pp pd

from \langle pd \in \{?pa,?pb,?pc\}\rangle and \langle a p \equiv_K a q\rangle and \langle b p \equiv_K b q\rangle
  and \langle c p \equiv_K c q\rangle
have cosh-dist pd ?pp = cosh-dist pd ?pq
  by (auto simp add: real-hyp2-C-cosh-dist)
with \langle pd \in hyp2\rangle and Rep-hyp2
have x = cosh-dist ?pq pd by (unfold x-def) (simp add: cosh-dist-swap)
from Rep-hyp2 [of p] and \langle pd \in hyp2\rangle and cosh-dist-positive [of ?pp pd]
have x \neq 0 by (unfold x-def) simp
from Rep-hyp2 and \langle pd \in hyp2\rangle and \langle ?pp \neq ?pq\rangle
have cross-ratio ps pt pe ?pp = (cosh-dist pd ?pq \times \sqrt{stpq} - cosh-dist pd ?pp \times \sqrt{stpq}) / (cosh-dist pd ?pp \times \sqrt{stpq} - cosh-dist pd ?pq \times \sqrt{stpq})
  unfolding ps-def and pt-def and pe-def and l-def and stpq-def
  by (simp add: perp-foot-cross-ratio-formula)
also from x-def and \langle x = cosh-dist ?pq pd\rangle
have \ldots = (x \times \sqrt{stpq} - x) / (x \times \sqrt{stpq} - x \times \sqrt{stpq}) by simp
also from \langle \sqrt{stpq} \times \sqrt{stpq} = stpq\rangle
have \ldots = (x \times \sqrt{stpq} - x) / ((x \times \sqrt{stpq} - x) \times \sqrt{stpq})
  by (simp add: algebra-simps)
also from \langle x \neq 0\rangle and \langle \sqrt{stpq} \neq 1\rangle have \ldots = 1 / \sqrt{stpq} by simp
finally show cross-ratio ps pt pe ?pp = 1 / \sqrt{stpq}.
qed
hence cross-ratio ps pt (perp-foot ?pa l) ?pp = 1 / \sqrt{stpq} by simp
have \forall pd \in \{?pa,?pb,?pc\}. proj2-incident pd m
proof
  fix pd
  assume pd \in \{?pa,?pb,?pc\}
  with Rep-hyp2 have pd \in hyp2 by auto
  with Rep-hyp2 and \langle ?pp \neq ?pq\rangle and proj2-line-through-incident
  have cross-ratio-correct ps pt ?pp (perp-foot pd l)
    and cross-ratio-correct ps pt ?pp (perp-foot ?pa l)
    unfolding ps-def and pt-def and l-def
    by (simp-all add: endpoints-in-S-perp-foot-cross-ratio-correct)
from \langle pd \in \{?pa,?pb,?pc\}\rangle
  and \forall pd \in \{?pa,?pb,?pc\}.
cross-ratio ps pt (perp-foot pd l) ?pp = 1 / (\sqrt{stpq})
have cross-ratio ps pt (perp-foot pd l) ?pp = 1 / \sqrt{stpq} by auto
with \langle cross-ratio ps pt (perp-foot ?pa l) ?pp = 1 / \sqrt{stpq}\rangle
have cross-ratio ps pt (perp-foot pd l) ?pp
  = cross-ratio ps pt (perp-foot ?pa l) ?pp
by simp

cross-ratio ps pt ?pp \((\perp\text{-foot } pd l)\)
\(\equiv\) cross-ratio ps pt ?pp \((\perp\text{-foot } ?pa l)\)

with (cross-ratio-correct ps pt ?pp \((\perp\text{-foot } pd l)\))
and (cross-ratio-correct ps pt ?pp \((\perp\text{-foot } ?pa l)\))

have \(\perp\text{-foot } pd l = \perp\text{-foot } ?pa l\) by (rule cross-ratio-unique)

with \(\text{Rep-hyp2} [af p] \text{ and } pd \in \text{hyp2}\)
and proj2-line-through-incident [of ?pp \(pq\)]

have drop-perp pd l = \(m\) by (unfold m-def l-def) simp

with drop-perp-incident [of pd l] show proj2-incident pd m by simp

qed

hence proj2-set-Col \{?pa, ?pb, ?pc\}

by (unfold proj2-set-Col-def) (simp add: exI [of \(m\)])


with \(z\text{-non-zero } ?pa\) and \(z\text{-non-zero } ?pb\) and \(z\text{-non-zero } ?pc\)

have real-euclid.Col (hyp2-rep a) (hyp2-rep b) (hyp2-rep c)

by (unfold hyp2-rep-def) (simp add: proj2-Col-iff-euclid-cart2)

thus \(B_K a b c \lor B_K b c a \lor B_K c a b\)

by (unfold real-hyp2-B-def real-euclid.Col-def)

qed

interpretation hyp2: tarski-absolute real-hyp2-C real-hyp2-B
using hyp2-axiom8 and hyp2-axiom9
by unfold-locales

8.15 The Klein–Beltrami model violates the Euclidean axiom

theorem hyp2-axiom10-false:
shows \(\neg (\forall a b c d t. B_K a d t \land B_K b d c \land a \neq d
t\Rightarrow (\exists x y. B_K a b x \land B_K a c y \land B_K x t y))\)

proof
assume \(\forall a b c d t. B_K a d t \land B_K b d c \land a \neq d
t\Rightarrow (\exists x y. B_K a b x \land B_K a c y \land B_K x t y)\)

let \(?ca = 0 : : \text{real}^2\)
and \(?cb = \text{vector } [1/2,0] : : \text{real}^2\)
and \(?cc = \text{vector } [0,1/2] : : \text{real}^2\)
and \(?cd = \text{vector } [1/4,1/4] : : \text{real}^2\)
and \(?ct = \text{vector } [1/2,1/2] : : \text{real}^2\)

let \(?a = \text{hyp2-abs } ?ca\)
and \(?b = \text{hyp2-abs } ?cb\)
and \(?c = \text{hyp2-abs } ?cc\)
and \(?d = \text{hyp2-abs } ?cd\)
and \(?t = \text{hyp2-abs } ?ct\)
hence \(?cd = (1/2) * ?ct \) and \(?cd - ?cb = (1/2) * ?cc - ?cb\)
  by (unfold vector-def) (simp-all add: vec-eq_iff)

hence \(?R x = ?cd \) and \(?cb \) and \(?cc \)
  by (unfold real-euclid-B-def) (simp-all add: ex1 [of - 1/2])

hence \(? ?d ?t \) and \(?k \)
  by (unfold real-hyp2-B-def) (simp-all add: hyp2-rep-abs-examples)

have \(?a \neq ?d\)

proof
  assume \(?a = ?d\)
  hence hyp2-rep \(?a = hyp2-rep \) by simp
  hence \(?ca = \) by (simp add: hyp2-rep-abs-examples)
  thus False by (simp add: vec-eq_iff forall-2)

qed

with \(\langle \rangle \) and \(\langle \rangle \)
and \(\langle \rangle \)

obtain \(\langle \rangle \) where \(\langle \rangle \) \(\langle \rangle \) and \(\langle \rangle \)
by blast

let \(\langle \rangle \)
and \(\langle \rangle \)

from \(\langle \rangle \) \(\langle \rangle \) and \(\langle \rangle \) and \(\langle \rangle \)

have \(\langle \rangle \)
  by (unfold real-euclid-B-def) (simp-all add: hyp2-rep-abs-examples)

from \(\langle \rangle \) \(\langle \rangle \) and \(\langle \rangle \) and \(\langle \rangle \)

obtain \(\langle \rangle \) where \(\langle \rangle \)
  and \(\langle \rangle \)
  and \(\langle \rangle \)
  by (unfold real-euclid-B-def) fast

from \(\langle \rangle \)

have \(\langle \rangle \)
  and \(\langle \rangle \)
  by (auto simp add: vec-eq_iff forall-2)

with \(\langle \rangle \)
and \(\langle \rangle \)

have \(\langle \rangle \) \(\langle \rangle \) and \(\langle \rangle \)
by simp-all

hence \(\langle \rangle \) \(\langle \rangle \) by simp-all

from \(\langle \rangle \)

have \(\langle \rangle \)
  and \(\langle \rangle \)
  by simp add: algebra-simps

with \(\langle \rangle \)
  and \(\langle \rangle \)

have \(\langle \rangle \) \(\langle \rangle \) \(\langle \rangle \)
by simp-all

hence \(\langle \rangle \) \(\langle \rangle \)

have \(\langle \rangle \)
also have \(\langle \rangle \)
also have \(\langle \rangle \)
finally have \(\langle \rangle \)

with \(\langle \rangle \)
and mult-less-cancel-left [of \(\langle \rangle \)]
have $(1 - l) \ast \ ?cx\$1 \leq 1 - l$ by auto

with $(1 - l) \ast (\ ?cx\$1) = 1/2$, have $l \leq 1/2$ by simp

have $\ ?cy\$2 \leq |\ ?cy\$2|$ by simp

also have ... $\leq \text{norm} \ ?cy$ by (rule component-le-norm-cart)

also have ... $< 1$ by (rule norm-hyp2-rep-lt-1)

finally have $\ ?cy\$2 < 1$.

with $(l \geq 0)$ and mult-less-cancel-left [of $l \ ?cy\$2 1]

have $l \ast \ ?cy\$2 \leq l$ by auto

with $(l \ast (\ ?cy\$2)) = 1/2$, have $l \geq 1/2$ by simp

with $(l \leq 1/2)$, have $l = 1/2$ by simp

with $(l \ast (\ ?cy\$2)) = 1/2$, have $\ ?cy\$2 = 1$ by simp

with $(\ ?cy\$2 < 1)$ show False by simp

qed

declare theorem hyp2-not-tarski: $\neg (\text{tarski real-hyp2-C real-hyp2-B})$

defined using hyp2-axiom10-false

by (unfold tarski-def tarski-space-def tarski-space-axioms-def) simp

Therefore axiom 10 is independent.

declare end

References


Bolyai-Lobachevskian Geometry; Projective Geometry. North-Holland


of Tarski’s Euclidean axiom. Master’s thesis, Victoria University


handle/10063/2315.
