The independence of Tarski's Euclidean axiom

T. J. M. Makarios

May 26, 2024

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

theory *Metric*
imports *HOL−Analysis.Multivariate-Analysis*
begin

locale *semimetric* =
  fixes *dist* :: $'p \Rightarrow 'p \Rightarrow real$
  assumes *nonneg* [*simp*]: $dist\ x\ y \geq 0$
  and *eq-0* [*simp*]: $dist\ x\ y = 0 \longleftrightarrow x = y$
  and *symm*: $dist\ x\ y = dist\ y\ x$
begin

lemma *refl* [*simp*]: *dist x x* = *0*
⟨*proof*⟩
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*⟩

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*⟩

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*
⟨*proof*⟩

lemma *unordered-pair-equality*: {*p*, *q*} = {*q*, *p*}
⟨*proof*⟩

lemma *cosine-rule*:
fixes *a b c* :: *real* ^ (′*n*::*finite*)
shows $(\textit{norm-dist}\ a\ c)^2$ =
$(\textit{norm-dist}\ a\ b)^2 + (\textit{norm-dist}\ b\ c)^2 + 2 * ((a - b) \cdot (b - c))$
⟨*proof*⟩

lemma *scalar-equiv*: $r *s\ x = r *_R x$
⟨*proof*⟩

lemma *norm-dist-dot*: $(\textit{norm-dist}\ x\ y)^2 = (x - y) \cdot (x - y)$
⟨*proof*⟩

definition *dep2* :: ′*a*::*real-vector* ⇒ ′*a* ⇒ *bool* **where**
$\textit{dep2}\ u\ v \triangleq \exists\, w\ r\ s.\ u = r *_R w \wedge v = s *_R w$

lemma *real2-eq*:
  fixes $u\ v :: real\hat{\ }2$
  assumes $u\$1 = v\$1$ **and** $u\$2 = v\$2$
  shows $u = v$
  ⟨*proof*⟩

definition $rotate2 :: real\hat{\ }2 \Rightarrow real\hat{\ }2$ **where**
  $rotate2\ x \triangleq vector\ [-x\$2,\ x\$1]$

declare *vector-2* [*simp*]

lemma *rotate2* [*simp*]:
  $(rotate2\ x)\$1 = -x\$2$
  $(rotate2\ x)\$2 = x\$1$
  ⟨*proof*⟩

lemma *rotate2-rotate2* [*simp*]: $rotate2\ (rotate2\ x) = -x$
⟨*proof*⟩

lemma *rotate2-dot* [*simp*]: $(rotate2\ u) \cdot (rotate2\ v) = u \cdot v$
  ⟨*proof*⟩

lemma *rotate2-scaleR* [*simp*]: $rotate2\ (k *_R x) = k *_R (rotate2\ x)$
⟨*proof*⟩

lemma *rotate2-uminus* [*simp*]: $rotate2\ (-x) = -(rotate2\ x)$
⟨*proof*⟩

lemma *rotate2-eq* [*iff*]: $rotate2\ x = rotate2\ y \longleftrightarrow x = y$
⟨*proof*⟩

lemma *dot2-rearrange-1*:
  fixes $u\ x :: real\hat{\ }2$
  assumes $u \cdot x = 0$ **and** $x\$1 \neq 0$
  shows $u = (u\$2\ /\ x\$1) *_R (rotate2\ x)$ (**is** $u = ?u'$)
⟨*proof*⟩

lemma *dot2-rearrange-2*:
  fixes $u\ x :: real\hat{\ }2$
  assumes $u \cdot x = 0$ **and** $x\$2 \neq 0$
  shows $u = -(u\$1\ /\ x\$2) *_R (rotate2\ x)$ (**is** $u = ?u'$)
⟨*proof*⟩

lemma *dot2-rearrange*:
  fixes $u\ x :: real\hat{\ }2$
  assumes $u \cdot x = 0$ **and** $x \neq 0$
  shows $\exists\, k.\ u = k *_R (rotate2\ x)$
⟨*proof*⟩

lemma *real2-orthogonal-dep2*:
  fixes *u v x* :: *real^2*
  assumes $x \neq 0$ **and** $u \cdot x = 0$ **and** $v \cdot x = 0$
  shows *dep2 u v*
⟨*proof*⟩

lemma *dot-left-diff-distrib*:
  fixes *u v x* :: *real^'n*
  shows $(u - v) \cdot x = (u \cdot x) - (v \cdot x)$
⟨*proof*⟩

lemma *dot-right-diff-distrib*:
  fixes *u v x* :: *real^'n*
  shows $x \cdot (u - v) = (x \cdot u) - (x \cdot v)$
⟨*proof*⟩

lemma *am-gm2*:
  fixes *a b* :: *real*
  assumes $a \geq 0$ **and** $b \geq 0$
  shows *sqrt* $(a * b) \leq (a + b) / 2$
  and *sqrt* $(a * b) = (a + b) / 2 \longleftrightarrow a = b$
⟨*proof*⟩

lemma *refl-on-allrel*: *refl-on* A $(A \times A)$
  ⟨*proof*⟩

lemma *refl-on-restrict*:
  assumes *refl-on* A r
  shows *refl-on* $(A \cap B)$ $(r \cap B \times B)$
⟨*proof*⟩

lemma *sym-allrel*: *sym* $(A \times A)$
  ⟨*proof*⟩

lemma *sym-restrict*:
  assumes *sym r*
  shows *sym* $(r \cap A \times A)$
⟨*proof*⟩

lemma *trans-allrel*: *trans* $(A \times A)$
  ⟨*proof*⟩

lemma *equiv-Int*:
  assumes *equiv* A r **and** *equiv* B s
  shows *equiv* $(A \cap B)$ $(r \cap s)$
⟨*proof*⟩

lemma *equiv-allrel*: *equiv* A $(A \times A)$
  ⟨*proof*⟩

lemma *equiv-restrict*:
  assumes *equiv* A r
  shows *equiv* $(A \cap B)$ $(r \cap B \times B)$
⟨*proof*⟩

lemma *invertible-times-eq-zero*:
  fixes x :: *real*^'*n* **and** A :: *real*^'*n*^'*n*
  assumes *invertible* A **and** $A *v\ x = 0$
  shows $x = 0$
  ⟨*proof*⟩

lemma *times-invertible-eq-zero*:
  fixes x :: *real*^'*n* **and** A :: *real*^'*n*^'*n*
  assumes *invertible* A **and** $x\ v*\ A = 0$
  shows $x = 0$
  ⟨*proof*⟩

lemma *matrix-id-invertible*:
  invertible (*mat 1* :: ('*a*::*semiring-1*)^'*n*^'*n*)
  ⟨*proof*⟩

lemma *Image-refl-on-nonempty*:
  assumes *refl-on* A r **and** $x \in A$
  shows $x \in r``\{x\}$
⟨*proof*⟩

lemma *quotient-element-nonempty*:
  assumes *equiv* A r **and** $X \in A//r$
  shows $\exists\ x.\ x \in X$
  ⟨*proof*⟩

lemma *zero-3*: (*3*::*3*) = *0*
  ⟨*proof*⟩

lemma *card-suc-ge-insert*:
  fixes A **and** x
  shows *card* A + *1* ≥ *card* (*insert* x A)
  ⟨*proof*⟩

lemma *card-le-UNIV*:
  fixes A :: ('*n*::*finite*) *set*
  shows *card* A ≤ *CARD*('*n*)
  ⟨*proof*⟩

lemma *partition-Image-element*:
  assumes *equiv* A r **and** $X \in A//r$ **and** $x \in X$
  shows $r``\{x\} = X$
  ⟨*proof*⟩

lemma *card-insert-ge*: *card* (*insert x A*) ≥ *card A*
⟨*proof*⟩

lemma *choose-1*:
assumes *card S* = *1*
shows ∃ *x*. *S* = {*x*}
⟨*proof*⟩

lemma *choose-2*:
assumes *card S* = *2*
shows ∃ *x y*. *S* = {*x*,*y*}
⟨*proof*⟩

lemma *choose-3*:
assumes *card S* = *3*
shows ∃ *x y z*. *S* = {*x*,*y*,*z*}
⟨*proof*⟩

lemma *card-gt-0-diff-singleton*:
assumes *card S* > *0* **and** *x* ∈ *S*
shows *card* (*S* − {*x*}) = *card S* − *1*
⟨*proof*⟩

lemma *eq-3-or-of-3*:
fixes *j* :: *4*
shows *j* = *3* ∨ (∃ *j'*::*3*. *j* = *of-int* (*Rep-bit1 j'*))
⟨*proof*⟩

lemma *sgn-plus*:
fixes *x y* :: *'a*::*linordered-idom*
assumes *sgn x* = *sgn y*
shows *sgn* (*x* + *y*) = *sgn x*
⟨*proof*⟩

lemma *sgn-div*:
fixes *x y* :: *'a*::*linordered-field*
assumes *y* ≠ *0* **and** *sgn x* = *sgn y*
shows *x* / *y* > *0*
⟨*proof*⟩

lemma *abs-plus*:
fixes *x y* :: *'a*::*linordered-idom*
assumes *sgn x* = *sgn y*
shows |*x* + *y*| = |*x*| + |*y*|
⟨*proof*⟩

lemma *sgn-plus-abs*:
fixes *x y* :: *'a*::*linordered-idom*

assumes $|x| > |y|$
shows $sgn\ (x + y) = sgn\ x$
$\langle proof \rangle$

end

3 Tarski's geometry

theory *Tarski*
imports *Complex-Main Miscellany Metric*
begin

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 $C :: {}'p \Rightarrow {}'p \Rightarrow {}'p \Rightarrow {}'p \Rightarrow bool$ (- - ≡ - - [*99*,*99*,*99*,*99*] *50*)
assumes *A1*: $\forall a\ b.\ a\ b \equiv b\ a$
and *A2*: $\forall a\ b\ p\ q\ r\ s.\ a\ b \equiv p\ q \wedge a\ b \equiv r\ s \longrightarrow p\ q \equiv r\ s$
and *A3*: $\forall a\ b\ c.\ a\ b \equiv c\ c \longrightarrow a = b$

locale *tarski-first5* = *tarski-first3* +
fixes $B :: {}'p \Rightarrow {}'p \Rightarrow {}'p \Rightarrow bool$
assumes *A4*: $\forall q\ a\ b\ c.\ \exists x.\ B\ q\ a\ x \wedge a\ x \equiv b\ c$
and *A5*: $\forall a\ b\ c\ d\ a'\ b'\ c'\ d'.\ a \neq b \wedge B\ a\ b\ c \wedge B\ a'\ b'\ c'$
$\wedge\ a\ b \equiv a'\ b' \wedge b\ c \equiv b'\ c' \wedge a\ d \equiv a'\ d' \wedge b$
$d \equiv b'\ d'$
$\longrightarrow c\ d \equiv c'\ d'$

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

locale *tarski-absolute* = *tarski-absolute-space* +
assumes *A8*: $\exists a\ b\ c.\ \neg\ B\ a\ b\ c \wedge \neg\ B\ b\ c\ a \wedge \neg\ B\ c\ a\ b$
and *A9*: $\forall p\ q\ a\ b\ c.\ p \neq q \wedge a\ p \equiv a\ q \wedge b\ p \equiv b\ q \wedge c\ p \equiv c\ q$
$\longrightarrow B\ a\ b\ c \vee B\ b\ c\ a \vee B\ c\ a\ b$

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

locale *tarski* = *tarski-absolute* + *tarski-space*

3.2 Semimetric spaces satisfy the first three axioms

context *semimetric*
begin
definition *smC* :: $'p \Rightarrow 'p \Rightarrow 'p \Rightarrow 'p \Rightarrow bool$ (- - $\equiv_{sm}$ - - [*99,99,99,99*] *50*)
where [*simp*]: $a\ b \equiv_{sm} c\ d \triangleq dist\ a\ b = dist\ c\ d$
end

sublocale *semimetric* < *tarski-first3 smC*
⟨*proof*⟩

3.3 Some consequences of the first three axioms

context *tarski-first3*
begin
lemma *A1'*: $a\ b \equiv b\ a$
⟨*proof*⟩

lemma *A2'*: $[\![a\ b \equiv p\ q;\ a\ b \equiv r\ s]\!] \Longrightarrow p\ q \equiv r\ s$
⟨*proof*⟩

lemma *A3'*: $a\ b \equiv c\ c \Longrightarrow a = b$
⟨*proof*⟩

theorem *th2-1*: $a\ b \equiv a\ b$
⟨*proof*⟩

theorem *th2-2*: $a\ b \equiv c\ d \Longrightarrow c\ d \equiv a\ b$
⟨*proof*⟩

theorem *th2-3*: $[\![a\ b \equiv c\ d;\ c\ d \equiv e\ f]\!] \Longrightarrow a\ b \equiv e\ f$
⟨*proof*⟩

theorem *th2-4*: $a\ b \equiv c\ d \Longrightarrow b\ a \equiv c\ d$
⟨*proof*⟩

theorem *th2-5*: $a\ b \equiv c\ d \Longrightarrow a\ b \equiv d\ c$
⟨*proof*⟩

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

definition *segments* :: $'p\ set\ set$ **where**
$segments = \{X.\ is\text{-}segment\ X\}$

definition *SC* :: $'p\ set \Rightarrow 'p\ set \Rightarrow bool$ **where**
$SC\ X\ Y \triangleq \exists\, w\ x\ y\ z.\ X = \{w,\ x\} \land Y = \{y,\ z\} \land w\ x \equiv y\ z$

definition *SC-rel* :: $('p\ set \times 'p\ set)\ set$ **where**
$SC\text{-}rel = \{(X,\ Y) \mid X\ Y.\ SC\ X\ Y\}$

lemma *left-segment-congruence*:
assumes $\{a, b\} = \{p, q\}$ **and** $p\ q \equiv c\ d$
shows $a\ b \equiv c\ d$
⟨*proof*⟩

lemma *right-segment-congruence*:
assumes $\{c, d\} = \{p, q\}$ **and** $a\ b \equiv p\ q$
shows $a\ b \equiv c\ d$
⟨*proof*⟩

lemma *C-SC-equiv*: $a\ b \equiv c\ d = SC\ \{a, b\}\ \{c, d\}$
⟨*proof*⟩

lemmas *SC-refl* = *th2-1* [*simplified*]

lemma *SC-rel-refl*: *refl-on segments SC-rel*
⟨*proof*⟩

lemma *SC-sym*:
assumes *SC X Y*
shows *SC Y X*
⟨*proof*⟩

lemma *SC-sym′*: *SC X Y* = *SC Y X*
⟨*proof*⟩

lemma *SC-rel-sym*: *sym SC-rel*
⟨*proof*⟩

lemma *SC-trans*:
assumes *SC X Y* **and** *SC Y Z*
shows *SC X Z*
⟨*proof*⟩

lemma *SC-rel-trans*: *trans SC-rel*
⟨*proof*⟩

lemma *A3-reversed*:
assumes $a\ a \equiv b\ c$
shows $b = c$
⟨*proof*⟩

lemma *equiv-segments-SC-rel*: *equiv segments SC-rel*
⟨*proof*⟩

end

3.4 Some consequences of the first five axioms

context *tarski-first5*
begin
  lemma *A4'*: $\exists x.\ B\ q\ a\ x \wedge a\ x \equiv b\ c$
    ⟨*proof*⟩

  theorem *th2-8*: $a\ a \equiv b\ b$
  ⟨*proof*⟩

  definition *OFS* :: $['p,'p,'p,'p,'p,'p,'p,'p] \Rightarrow$ *bool* **where**
   $OFS\ a\ b\ c\ d\ a'\ b'\ c'\ d' \triangleq$
     $B\ a\ b\ c \wedge B\ a'\ b'\ c' \wedge a\ b \equiv a'\ b' \wedge b\ c \equiv b'\ c' \wedge a\ d \equiv a'\ d' \wedge b\ d \equiv b'\ d'$

  lemma *A5'*: $[\![OFS\ a\ b\ c\ d\ a'\ b'\ c'\ d';\ a \neq b]\!] \Longrightarrow c\ d \equiv c'\ d'$
  ⟨*proof*⟩

  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*⟩

  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*⟩

  theorem *th2-12*:
    assumes $q \neq a$
    shows $\exists!x.\ B\ q\ a\ x \wedge a\ x \equiv b\ c$
    ⟨*proof*⟩
end

3.5 Simple theorems about betweenness

theorem (**in** *tarski-first5*) *th3-1*: $B\ a\ b\ b$
⟨*proof*⟩

context *tarski-absolute-space*
begin
  lemma *A6'*:
    assumes $B\ a\ b\ a$
    shows $a = b$
  ⟨*proof*⟩

lemma *A7′*:
assumes $B\ a\ p\ c$ **and** $B\ b\ q\ c$
shows $\exists x.\ B\ p\ x\ b \wedge B\ q\ x\ a$
⟨*proof*⟩

lemma *A11′*:
assumes $\forall\ x\ y.\ x \in X \wedge y \in Y \longrightarrow B\ a\ x\ y$
shows $\exists\ b.\ \forall\ x\ y.\ x \in X \wedge y \in Y \longrightarrow B\ x\ b\ y$
⟨*proof*⟩

theorem *th3-2*:
assumes $B\ a\ b\ c$
shows $B\ c\ b\ a$
⟨*proof*⟩

theorem *th3-4*:
assumes $B\ a\ b\ c$ **and** $B\ b\ a\ c$
shows $a = b$
⟨*proof*⟩

theorem *th3-5-1*:
assumes $B\ a\ b\ d$ **and** $B\ b\ c\ d$
shows $B\ a\ b\ c$
⟨*proof*⟩

theorem *th3-6-1*:
assumes $B\ a\ b\ c$ **and** $B\ a\ c\ d$
shows $B\ b\ c\ d$
⟨*proof*⟩

theorem *th3-7-1*:
assumes $b \neq c$ **and** $B\ a\ b\ c$ **and** $B\ b\ c\ d$
shows $B\ a\ c\ d$
⟨*proof*⟩

theorem *th3-7-2*:
assumes $b \neq c$ **and** $B\ a\ b\ c$ **and** $B\ b\ c\ d$
shows $B\ a\ b\ d$
⟨*proof*⟩
end

3.6 Simple theorems about congruence and betweenness

definition (**in** *tarski-first5*) *Col* :: $'p \Rightarrow 'p \Rightarrow 'p \Rightarrow bool$ **where**
$Col\ a\ b\ c \triangleq B\ a\ b\ c \vee B\ b\ c\ a \vee B\ c\ a\ b$

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*
  (- - $\equiv_{\mathbb{R}}$ - - [*99*,*99*,*99*,*99*] *50*) **where**
    real-euclid-C ≜ *norm-metric.smC*

definition *real-euclid-B* :: [*real*^('*n*::*finite*), *real*^('*n*), *real*^('*n*)] ⇒ *bool*
  ($B_{\mathbb{R}}$ - - - [*99*,*99*,*99*] *50*) **where**
    $B_{\mathbb{R}}$ *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*⟩

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

lemma *rearrange-real-euclid-B*:
  fixes *w y z* :: *real*^('*n*) **and** *h*
  shows *y* − *w* = *h* $*_R$ (*z* − *w*) ⟷ *y* = *h* $*_R$ *z* + (*1* − *h*) $*_R$ *w*
⟨*proof*⟩

interpretation *real-euclid*: *tarski-absolute-space real-euclid-C real-euclid-B*
⟨*proof*⟩

4.3 Real Euclidean space satisfies the Euclidean axiom

lemma *rearrange-real-euclid-B-2*:
  fixes *a b c* :: *real*^('*n*::*finite*)
  assumes *l* ≠ *0*
  shows *b* − *a* = *l* $*_R$ (*c* − *a*) ⟷ *c* = (*1*/*l*) $*_R$ *b* + (*1* − *1*/*l*) $*_R$ *a*
⟨*proof*⟩

interpretation *real-euclid*: *tarski-space real-euclid-C real-euclid-B*
⟨*proof*⟩

4.4 The real Euclidean plane

lemma *Col-dep2*:
  real-euclid.Col a b c ⟷ *dep2* (*b* − *a*) (*c* − *a*)
⟨*proof*⟩

lemma *non-Col-example*:
  ¬(*real-euclid.Col 0* (*vector* [*1*/*2*,*0*] :: *real*^*2*) (*vector* [*0*,*1*/*2*]))
  (**is** ¬ (*real-euclid.Col ?a ?b ?c*))

⟨*proof*⟩

interpretation *real-euclid*:
tarski *real-euclid-C*::([*real*^*2*, *real*^*2*, *real*^*2*, *real*^*2*] ⇒ *bool*) *real-euclid-B*
⟨*proof*⟩

4.5 Special cases of theorems of Tarski's geometry

lemma *real-euclid-B-disjunction*:
assumes $l \geq 0$ **and** $b - a = l *_R (c - a)$
shows $B_{\mathbb{R}}\ a\ b\ c \vee B_{\mathbb{R}}\ a\ c\ b$
⟨*proof*⟩

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** $B_{\mathbb{R}}\ a\ b\ c$ **and** $B_{\mathbb{R}}\ a\ b\ d$
shows $B_{\mathbb{R}}\ a\ c\ d \vee B_{\mathbb{R}}\ a\ d\ c$
⟨*proof*⟩

theorem *real-euclid-th5-3*:
assumes $B_{\mathbb{R}}\ a\ b\ d$ **and** $B_{\mathbb{R}}\ a\ c\ d$
shows $B_{\mathbb{R}}\ a\ b\ c \vee B_{\mathbb{R}}\ a\ c\ b$
⟨*proof*⟩

end

5 Linear algebra

theory *Linear-Algebra2*
imports *Miscellany*
begin

lemma *exhaust-4*:
fixes *x* :: *4*
shows $x = 1 \vee x = 2 \vee x = 3 \vee x = 4$
⟨*proof*⟩

lemma *forall-4*: (∀ *i*::*4*. *P i*) ⟷ *P 1* ∧ *P 2* ∧ *P 3* ∧ *P 4*
⟨*proof*⟩

lemma *UNIV-4*: (*UNIV*::(*4 set*)) = {*1*, *2*, *3*, *4*}
⟨*proof*⟩

lemma *vector-4*:
fixes *w* :: *'a*::*zero*
shows (*vector* [*w*, *x*, *y*, *z*] :: *'a*^*4*)$*1* = *w*
and (*vector* [*w*, *x*, *y*, *z*] :: *'a*^*4*)$*2* = *x*

and $(vector\ [w, x, y, z] :: 'a^{\frown}4)\$3 = y$
and $(vector\ [w, x, y, z] :: 'a^{\frown}4)\$4 = z$
⟨*proof*⟩

definition
is-basis :: $(real^{\frown\prime}n)$ *set* ⇒ *bool* **where**
is-basis $S \triangleq$ *independent* $S \wedge$ *span* $S = UNIV$

lemma *card-finite*:
assumes *card* $S = CARD('n::finite)$
shows *finite* S
⟨*proof*⟩

lemma *independent-is-basis*:
fixes $B :: (real^{\frown\prime}n)$ *set*
shows *independent* $B \wedge$ *card* $B = CARD('n) \longleftrightarrow$ *is-basis* B
⟨*proof*⟩

lemma *basis-finite*:
fixes $B :: (real^{\frown\prime}n)$ *set*
assumes *is-basis* B
shows *finite* B
⟨*proof*⟩

lemma *basis-expand*:
assumes *is-basis* B
shows $\exists\, c.\ v = (\sum w{\in}B.\ (c\ w) *_R w)$
⟨*proof*⟩

lemma *not-span-independent-insert*:
fixes $v :: ('a::real\text{-}vector)^{\frown\prime}n$
assumes *independent* S **and** $v \notin$ *span* S
shows *independent* (*insert* v S)
⟨*proof*⟩

lemma *orthogonal-sum*:
fixes $v :: real^{\frown\prime}n$
assumes $\bigwedge w.\ w{\in}S \Longrightarrow$ *orthogonal* v w
shows *orthogonal* v $(\sum w{\in}S.\ c\ w *s\ w)$
⟨*proof*⟩

lemma *orthogonal-self-eq-0*:
fixes $v :: ('a::real\text{-}inner)^{\frown\prime}n$
assumes *orthogonal* v v
shows $v = 0$
⟨*proof*⟩

lemma *orthogonal-in-span-eq-0*:
fixes $v :: real^{\frown\prime}n$

assumes $v \in span\ S$ **and** $\bigwedge w.\ w{\in}S \Longrightarrow orthogonal\ v\ w$
shows $v = 0$
⟨*proof*⟩

lemma *orthogonal-independent*:
fixes $v :: real^{\frown\prime}n$
assumes *independent S* **and** $v \neq 0$ **and** $\bigwedge w.\ w{\in}S \Longrightarrow orthogonal\ v\ w$
shows *independent* (*insert v S*)
⟨*proof*⟩

lemma *dot-scaleR-mult*:
shows $(k *_R a) \cdot b = k * (a \cdot b)$ **and** $a \cdot (k *_R b) = k * (a \cdot b)$
⟨*proof*⟩

lemma *dependent-explicit-finite*:
fixes $S :: (('a{::}\{real\text{-}vector,field\})^{\frown\prime}n)\ set$
assumes *finite S*
shows $dependent\ S \longleftrightarrow (\exists\ u.\ (\exists\ v{\in}S.\ u\ v \neq 0) \wedge (\sum\ v{\in}S.\ u\ v *_R v) = 0)$
⟨*proof*⟩

lemma *dependent-explicit-2*:
fixes $v\ w :: ('a{::}\{field,real\text{-}vector\})^{\frown\prime}n$
assumes $v \neq w$
shows $dependent\ \{v,\ w\} \longleftrightarrow (\exists\ i\ j.\ (i \neq 0 \vee j \neq 0) \wedge i *_R v + j *_R w = 0)$
⟨*proof*⟩

5.1 Matrices

lemma *zero-not-invertible*:
$\neg\ (invertible\ (0{::}real^{\frown\prime}n^{\frown\prime}n))$
⟨*proof*⟩

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

lemma *vector-matrix-row*:
fixes $x :: ('a{::}comm\text{-}semiring\text{-}1)^{\frown\prime}m$ **and** $A :: ('a^{\frown\prime}n^{\frown\prime}m)$
shows $x\ v{*}\ A = (\sum\ i{\in}UNIV.\ (x\$i) *s\ (A\$i))$
⟨*proof*⟩

lemma *matrix-inv*:
assumes *invertible M*
shows $matrix\text{-}inv\ M ** M = mat\ 1$
and $M ** matrix\text{-}inv\ M = mat\ 1$
⟨*proof*⟩

lemma *matrix-inv-invertible*:
assumes *invertible M*
shows *invertible* (*matrix-inv M*)
⟨*proof*⟩

lemma *invertible-times-non-zero*:
  fixes M :: $real^{\frown\prime}n^{\frown\prime}n$
  assumes *invertible* M **and** $v \neq 0$
  shows $M *v\ v \neq 0$
  ⟨*proof*⟩

lemma *matrix-right-invertible-ker*:
  fixes M :: $real^{\frown}('m::finite)^{\frown\prime}n$
  shows $(\exists\ M'.\ M ** M' = mat\ 1) \longleftrightarrow (\forall\ x.\ x\ v{*}\ M = 0 \longrightarrow x = 0)$
  ⟨*proof*⟩

lemma *left-invertible-iff-invertible*:
  fixes M :: $real^{\frown\prime}n^{\frown\prime}n$
  shows $(\exists\ N.\ N ** M = mat\ 1) \longleftrightarrow$ *invertible* M
  ⟨*proof*⟩

lemma *right-invertible-iff-invertible*:
  fixes M :: $real^{\frown\prime}n^{\frown\prime}n$
  shows $(\exists\ N.\ M ** N = mat\ 1) \longleftrightarrow$ *invertible* M
  ⟨*proof*⟩

definition *symmatrix* :: $'a^{\frown\prime}n^{\frown\prime}n \Rightarrow bool$ **where**
  symmatrix $M \triangleq$ *transpose* $M = M$

lemma *symmatrix-preserve*:
  fixes $M\ N$ :: $('a::comm\text{-}semiring\text{-}1)^{\frown\prime}n^{\frown\prime}n$
  assumes *symmatrix* M
  shows *symmatrix* $(N ** M **$ *transpose* $N)$
⟨*proof*⟩

lemma *non-zero-mult-invertible-non-zero*:
  fixes M :: $real^{\frown\prime}n^{\frown\prime}n$
  assumes $v \neq 0$ **and** *invertible* M
  shows $v\ v{*}\ M \neq 0$
  ⟨*proof*⟩

end

6 Right group actions

theory *Action*
  imports *HOL−Algebra.Group*
begin

locale *action* = *group* +
  fixes *act* :: $'b \Rightarrow 'a \Rightarrow 'b$ (**infixl** $<o$ *69*)
  assumes *id-act* [*simp*]: $b <o\ \mathbf{1} = b$
  and *act-act'*:

$g \in carrier\ G \wedge h \in carrier\ G \longrightarrow (b <o\ g) <o\ h = b <o\ (g \otimes h)$
begin

lemma *act-act*:
  assumes $g \in carrier\ G$ **and** $h \in carrier\ G$
  shows $(b <o\ g) <o\ h = b <o\ (g \otimes h)$
⟨*proof*⟩

lemma *act-act-inv* [*simp*]:
  assumes $g \in carrier\ G$
  shows $b <o\ g <o\ inv\ g = b$
⟨*proof*⟩

lemma *act-inv-act* [*simp*]:
  assumes $g \in carrier\ G$
  shows $b <o\ inv\ g <o\ g = b$
  ⟨*proof*⟩

lemma *act-inv-iff*:
  assumes $g \in carrier\ G$
  shows $b <o\ inv\ g = c \longleftrightarrow b = c <o\ g$
⟨*proof*⟩

end

end

7 Projective geometry

theory *Projective*
  imports *Linear-Algebra2*
  Euclid-Tarski
  Action
begin

7.1 Proportionality on non-zero vectors

context *vector-space*
begin

  definition *proportionality* :: $('b \times 'b)\ set$ **where**
    $proportionality \triangleq \{(x, y).\ x \neq 0 \wedge y \neq 0 \wedge (\exists k.\ x = scale\ k\ y)\}$

  definition *non-zero-vectors* :: $'b\ set$ **where**
    $non\text{-}zero\text{-}vectors \triangleq \{x.\ x \neq 0\}$

  lemma *proportionality-refl-on*: *refl-on local.non-zero-vectors local.proportionality*
  ⟨*proof*⟩

lemma *proportionality-sym*: *sym local.proportionality*
⟨*proof*⟩

lemma *proportionality-trans*: *trans local.proportionality*
⟨*proof*⟩

theorem *proportionality-equiv*: *equiv local.non-zero-vectors local.proportionality*
⟨*proof*⟩

end

definition *invertible-proportionality* ::
((*real*^(′*n*::*finite*)^′*n*) × (*real*^′*n*^′*n*)) *set* **where**
invertible-proportionality ≜
real-vector.proportionality ∩ (*Collect invertible* × *Collect invertible*)

lemma *invertible-proportionality-equiv*:
equiv (*Collect invertible* :: (*real*^(′*n*::*finite*)^′*n*) *set*)
invertible-proportionality
(**is** *equiv ?invs* -)
⟨*proof*⟩

7.2 Points of the real projective plane

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

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*⟩

lemma *proj2-rep-non-zero*: *proj2-rep x* ≠ *0*
⟨*proof*⟩

lemma *proj2-rep-abs*:
fixes *v* :: *real*^*3*
assumes *v* ∈ *real-vector.non-zero-vectors*
shows (*v*, *proj2-rep* (*proj2-abs v*)) ∈ *real-vector.proportionality*
⟨*proof*⟩

lemma *proj2-abs-rep*: *proj2-abs* (*proj2-rep x*) = *x*
⟨*proof*⟩

lemma *proj2-abs-mult*:

assumes $c \neq 0$
shows *proj2-abs* $(c *_R v) =$ *proj2-abs* v
⟨*proof*⟩

lemma *proj2-abs-mult-rep*:
assumes $c \neq 0$
shows *proj2-abs* $(c *_R$ *proj2-rep* $x) = x$
⟨*proof*⟩

lemma *proj2-rep-inj*: *inj proj2-rep*
⟨*proof*⟩

lemma *proj2-rep-abs2*:
assumes $v \neq 0$
shows $\exists\ k.\ k \neq 0 \wedge$ *proj2-rep* (*proj2-abs* v) $= k *_R v$
⟨*proof*⟩

lemma *proj2-abs-abs-mult*:
assumes *proj2-abs* $v =$ *proj2-abs* w **and** $w \neq 0$
shows $\exists\ c.\ v = c *_R w$
⟨*proof*⟩

lemma *dependent-proj2-abs*:
assumes $p \neq 0$ **and** $q \neq 0$ **and** $i \neq 0 \vee j \neq 0$ **and** $i *_R p + j *_R q = 0$
shows *proj2-abs* $p =$ *proj2-abs* q
⟨*proof*⟩

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 $i \neq 0 \vee j \neq 0$
shows $v = w$
⟨*proof*⟩

lemma *proj2-rep-independent*:
assumes $p \neq q$
shows *independent* {*proj2-rep* p, *proj2-rep* q}
⟨*proof*⟩

7.3 Lines of the real projective plane

definition *proj2-Col* :: [*proj2*, *proj2*, *proj2*] ⇒ *bool* **where**
proj2-Col $p\ q\ r \triangleq$
$(\exists\ i\ j\ k.\ i *_R$ *proj2-rep* $p + j *_R$ *proj2-rep* $q + k *_R$ *proj2-rep* $r = 0$
$\wedge\ (i{\neq}0 \vee j{\neq}0 \vee k{\neq}0))$

lemma *proj2-Col-abs*:
assumes $p \neq 0$ **and** $q \neq 0$ **and** $r \neq 0$ **and** $i \neq 0 \vee j \neq 0 \vee k \neq 0$
and $i *_R p + j *_R q + k *_R r = 0$

shows *proj2-Col* (*proj2-abs p*) (*proj2-abs q*) (*proj2-abs r*)
(**is** *proj2-Col ?pp ?pq ?pr*)
⟨*proof*⟩

lemma *proj2-Col-permute*:
assumes *proj2-Col a b c*
shows *proj2-Col a c b*
and *proj2-Col b a c*
⟨*proof*⟩

lemma *proj2-Col-coincide*: *proj2-Col a a c*
⟨*proof*⟩

lemma *proj2-Col-iff*:
assumes $a \neq r$
shows *proj2-Col a r t* ⟷
$t = a \vee (\exists\ i.\ t = \mathit{proj2\text{-}abs}\ (i *_R (\mathit{proj2\text{-}rep}\ a) + (\mathit{proj2\text{-}rep}\ r)))$
⟨*proof*⟩

definition *proj2-Col-coeff* :: *proj2* ⇒ *proj2* ⇒ *proj2* ⇒ *real* **where**
$\mathit{proj2\text{-}Col\text{-}coeff}\ a\ r\ t \triangleq \epsilon\ i.\ t = \mathit{proj2\text{-}abs}\ (i *_R \mathit{proj2\text{-}rep}\ a + \mathit{proj2\text{-}rep}\ r)$

lemma *proj2-Col-coeff*:
assumes *proj2-Col a r t* **and** $a \neq r$ **and** $t \neq a$
shows $t = \mathit{proj2\text{-}abs}\ ((\mathit{proj2\text{-}Col\text{-}coeff}\ a\ r\ t) *_R \mathit{proj2\text{-}rep}\ a + \mathit{proj2\text{-}rep}\ r)$
⟨*proof*⟩

lemma *proj2-Col-coeff-unique'*:
assumes $a \neq 0$ **and** $r \neq 0$ **and** *proj2-abs a* ≠ *proj2-abs r*
and $\mathit{proj2\text{-}abs}\ (i *_R a + r) = \mathit{proj2\text{-}abs}\ (j *_R a + r)$
shows $i = j$
⟨*proof*⟩

lemma *proj2-Col-coeff-unique*:
assumes $a \neq r$
and $\mathit{proj2\text{-}abs}\ (i *_R \mathit{proj2\text{-}rep}\ a + \mathit{proj2\text{-}rep}\ r)$
$= \mathit{proj2\text{-}abs}\ (j *_R \mathit{proj2\text{-}rep}\ a + \mathit{proj2\text{-}rep}\ r)$
shows $i = j$
⟨*proof*⟩

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*
⟨*proof*⟩

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

⟨*proof*⟩

lemma *L2P-inj* [*simp*]:
assumes *L2P l* = *L2P m*
shows *l* = *m*
⟨*proof*⟩

lemma *P2L-to-L2P*: *P2L p* = *l* ⟷ *p* = *L2P l*
⟨*proof*⟩

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* ≠ *0*
shows ∃ *k*. *k* ≠ *0* ∧ *proj2-line-rep* (*proj2-line-abs v*) = *k* $*_R$ *v*
⟨*proof*⟩

lemma *proj2-line-abs-rep* [*simp*]: *proj2-line-abs* (*proj2-line-rep l*) = *l*
⟨*proof*⟩

lemma *proj2-line-rep-non-zero*: *proj2-line-rep l* ≠ *0*
⟨*proof*⟩

lemma *proj2-line-rep-dependent*:
assumes *i* $*_R$ *proj2-line-rep l* + *j* $*_R$ *proj2-line-rep m* = *0*
and *i* ≠ *0* ∨ *j* ≠ *0*
shows *l* = *m*
⟨*proof*⟩

lemma *proj2-line-abs-mult*:
assumes *k* ≠ *0*
shows *proj2-line-abs* (*k* $*_R$ *v*) = *proj2-line-abs v*
⟨*proof*⟩

lemma *proj2-line-abs-abs-mult*:
assumes *proj2-line-abs v* = *proj2-line-abs w* **and** *w* ≠ *0*
shows ∃ *k*. *v* = *k* $*_R$ *w*
⟨*proof*⟩

definition *proj2-incident* :: *proj2* ⇒ *proj2-line* ⇒ *bool* **where**
proj2-incident p l ≜ (*proj2-rep p*) · (*proj2-line-rep l*) = *0*

lemma *proj2-points-define-line*:
shows ∃ *l*. *proj2-incident p l* ∧ *proj2-incident q l*
⟨*proof*⟩

definition *proj2-line-through* :: *proj2* ⇒ *proj2* ⇒ *proj2-line* **where**
proj2-line-through p q ≜ ϵ *l*. *proj2-incident p l* ∧ *proj2-incident q l*

lemma *proj2-line-through-incident*:
shows *proj2-incident p* (*proj2-line-through p q*)
and *proj2-incident q* (*proj2-line-through p q*)
⟨*proof*⟩

lemma *proj2-line-through-unique*:
assumes *p* ≠ *q* **and** *proj2-incident p l* **and** *proj2-incident q l*
shows *l* = *proj2-line-through p q*
⟨*proof*⟩

lemma *proj2-incident-unique*:
assumes *proj2-incident p l*
and *proj2-incident q l*
and *proj2-incident p m*
and *proj2-incident q m*
shows *p* = *q* ∨ *l* = *m*
⟨*proof*⟩

lemma *proj2-lines-define-point*: ∃ *p*. *proj2-incident p l* ∧ *proj2-incident p m*
⟨*proof*⟩

definition *proj2-intersection* :: *proj2-line* ⇒ *proj2-line* ⇒ *proj2* **where**
proj2-intersection l m ≜ *L2P* (*proj2-line-through* (*L2P l*) (*L2P m*))

lemma *proj2-incident-switch*:
assumes *proj2-incident p l*
shows *proj2-incident* (*L2P l*) (*P2L p*)
⟨*proof*⟩

lemma *proj2-intersection-incident*:
shows *proj2-incident* (*proj2-intersection l m*) *l*
and *proj2-incident* (*proj2-intersection l m*) *m*
⟨*proof*⟩

lemma *proj2-intersection-unique*:
assumes *l* ≠ *m* **and** *proj2-incident p l* **and** *proj2-incident p m*
shows *p* = *proj2-intersection l m*
⟨*proof*⟩

lemma *proj2-not-self-incident*:
¬ (*proj2-incident p* (*P2L p*))
⟨*proof*⟩

lemma *proj2-another-point-on-line*:
∃ *q*. *q* ≠ *p* ∧ *proj2-incident q l*

⟨*proof*⟩

lemma *proj2-another-line-through-point*:
  ∃ *m*. *m* ≠ *l* ∧ *proj2-incident p m*
⟨*proof*⟩

lemma *proj2-incident-abs*:
  assumes *v* ≠ *0* **and** *w* ≠ *0*
  shows *proj2-incident* (*proj2-abs v*) (*proj2-line-abs w*) ⟷ *v* · *w* = *0*
⟨*proof*⟩

lemma *proj2-incident-left-abs*:
  assumes *v* ≠ *0*
  shows *proj2-incident* (*proj2-abs v*) *l* ⟷ *v* · (*proj2-line-rep l*) = *0*
⟨*proof*⟩

lemma *proj2-incident-right-abs*:
  assumes *v* ≠ *0*
  shows *proj2-incident p* (*proj2-line-abs v*) ⟷ (*proj2-rep p*) · *v* = *0*
⟨*proof*⟩

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*
  ⟨*proof*⟩

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*] :: *real^3^3*)
  (**is** - ⟷ ¬ *invertible* (*vector* [*?u*, *?v*, *?w*]))
⟨*proof*⟩

lemma *not-invertible-iff-proj2-set-Col*:
  ¬ *invertible* (*vector* [*proj2-rep p*, *proj2-rep q*, *proj2-rep r*] :: *real^3^3*)
  ⟷ *proj2-set-Col* {*p*,*q*,*r*}
  (**is** ¬ *invertible ?M* ⟷ -)
⟨*proof*⟩

lemma *proj2-Col-iff-set-Col*:
  proj2-Col p q r ⟷ *proj2-set-Col* {*p*,*q*,*r*}
  ⟨*proof*⟩

lemma *proj2-incident-Col*:

assumes *proj2-incident p l* **and** *proj2-incident q l* **and** *proj2-incident r l*
shows *proj2-Col p q r*
⟨*proof*⟩

lemma *proj2-incident-iff-Col*:
assumes $p \neq q$ **and** *proj2-incident p l* **and** *proj2-incident q l*
shows *proj2-incident r l* $\longleftrightarrow$ *proj2-Col p q r*
⟨*proof*⟩

lemma *proj2-incident-iff*:
assumes $p \neq q$ **and** *proj2-incident p l* **and** *proj2-incident q l*
shows *proj2-incident r l*
$\longleftrightarrow r = p \vee (\exists\ k.\ r = \textit{proj2-abs}\ (k *_R \textit{proj2-rep}\ p + \textit{proj2-rep}\ q))$
⟨*proof*⟩

lemma *not-proj2-set-Col-iff-span*:
assumes *card S* = *3*
shows ¬ *proj2-set-Col S* $\longleftrightarrow$ *span* (*proj2-rep* ‘ *S*) = *UNIV*
⟨*proof*⟩

lemma *proj2-no-3-Col-span*:
assumes *proj2-no-3-Col S* **and** $p \in S$
shows *span* (*proj2-rep* ‘ $(S - \{p\})$) = *UNIV*
⟨*proof*⟩

lemma *fourth-proj2-no-3-Col*:
assumes ¬ *proj2-Col p q r*
shows ∃ *s. proj2-no-3-Col* $\{s,r,p,q\}$
⟨*proof*⟩

lemma *proj2-set-Col-expand*:
assumes *proj2-set-Col S* **and** $\{p,q,r\} \subseteq S$ **and** $p \neq q$ **and** $r \neq p$
shows $\exists\ k.\ r = \textit{proj2-abs}\ (k *_R \textit{proj2-rep}\ p + \textit{proj2-rep}\ q)$
⟨*proof*⟩

7.4 Collineations of the real projective plane

typedef *cltn2* =
(*Collect invertible* :: (*real^3^3*) *set*)//*invertible-proportionality*
⟨*proof*⟩

definition *cltn2-rep* :: *cltn2* ⇒ *real^3^3* **where**
cltn2-rep A ≜ ϵ *B. B* ∈ *Rep-cltn2 A*

definition *cltn2-abs* :: *real^3^3* ⇒ *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*⟩

lemma *cltn2-rep-invertible*: *invertible* (*cltn2-rep A*)
⟨*proof*⟩

lemma *cltn2-rep-abs*:
fixes *A* :: *real^3^3*
assumes *invertible A*
shows (*A*, *cltn2-rep* (*cltn2-abs A*)) ∈ *invertible-proportionality*
⟨*proof*⟩

lemma *cltn2-rep-abs2*:
assumes *invertible A*
shows ∃ *k*. *k* ≠ *0* ∧ *cltn2-rep* (*cltn2-abs A*) = $k *_R A$
⟨*proof*⟩

lemma *cltn2-abs-rep*: *cltn2-abs* (*cltn2-rep A*) = *A*
⟨*proof*⟩

lemma *cltn2-abs-mult*:
assumes *k* ≠ *0* **and** *invertible A*
shows *cltn2-abs* ($k *_R A$) = *cltn2-abs A*
⟨*proof*⟩

lemma *cltn2-abs-mult-rep*:
assumes *k* ≠ *0*
shows *cltn2-abs* ($k *_R$ *cltn2-rep A*) = *A*
⟨*proof*⟩

lemma *apply-cltn2-abs*:
assumes *x* ≠ *0* **and** *invertible A*
shows *apply-cltn2* (*proj2-abs x*) (*cltn2-abs A*) = *proj2-abs* (*x v∗ A*)
⟨*proof*⟩

lemma *apply-cltn2-left-abs*:
assumes *v* ≠ *0*
shows *apply-cltn2* (*proj2-abs v*) *C* = *proj2-abs* (*v v∗ cltn2-rep C*)
⟨*proof*⟩

lemma *apply-cltn2-right-abs*:
assumes *invertible M*
shows *apply-cltn2 p* (*cltn2-abs M*) = *proj2-abs* (*proj2-rep p v∗ M*)
⟨*proof*⟩

lemma *non-zero-mult-rep-non-zero*:
  assumes $v \neq 0$
  shows $v\ v{*}\ cltn2\text{-}rep\ C \neq 0$
  ⟨*proof*⟩

lemma *rep-mult-rep-non-zero*: $proj2\text{-}rep\ p\ v{*}\ cltn2\text{-}rep\ A \neq 0$
  ⟨*proof*⟩

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

7.4.1 As a group

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

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

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

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

lemma *cltn2-compose-left-abs*:
  assumes *invertible M*
  shows $cltn2\text{-}compose\ (cltn2\text{-}abs\ M)\ A = cltn2\text{-}abs\ (M\ {*}{*}\ cltn2\text{-}rep\ A)$
⟨*proof*⟩

lemma *cltn2-compose-right-abs*:
  assumes *invertible M*
  shows $cltn2\text{-}compose\ A\ (cltn2\text{-}abs\ M) = cltn2\text{-}abs\ (cltn2\text{-}rep\ A\ {*}{*}\ M)$
⟨*proof*⟩

lemma *cltn2-abs-rep-abs-mult*:
  assumes *invertible M* **and** *invertible N*
  shows $cltn2\text{-}abs\ (cltn2\text{-}rep\ (cltn2\text{-}abs\ M)\ {*}{*}\ N) = cltn2\text{-}abs\ (M\ {*}{*}\ N)$
⟨*proof*⟩

lemma *cltn2-assoc*:
  $cltn2\text{-}compose\ (cltn2\text{-}compose\ A\ B)\ C = cltn2\text{-}compose\ A\ (cltn2\text{-}compose\ B\ C)$
⟨*proof*⟩

lemma *cltn2-left-id*: $cltn2\text{-}compose\ cltn2\text{-}id\ A = A$
⟨*proof*⟩

lemma *cltn2-left-inverse*: *cltn2-compose* (*cltn2-inverse* *A*) *A* = *cltn2-id*
⟨*proof*⟩

lemma *cltn2-left-inverse-ex*:
∃ *B*. *cltn2-compose* *B* *A* = *cltn2-id*
⟨*proof*⟩

interpretation *cltn2*:
group (|*carrier* = *UNIV*, *mult* = *cltn2-compose*, *one* = *cltn2-id*|)
⟨*proof*⟩

lemma *cltn2-inverse-inv* [*simp*]:
$inv_{(|carrier\ =\ UNIV,\ mult\ =\ cltn2\text{-}compose,\ one\ =\ cltn2\text{-}id|)}\ A$
= *cltn2-inverse* *A*
⟨*proof*⟩

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* [*simp*]: *apply-cltn2* *p* *cltn2-id* = *p*
⟨*proof*⟩

lemma *apply-cltn2-compose*:
apply-cltn2 (*apply-cltn2* *p* *A*) *B* = *apply-cltn2* *p* (*cltn2-compose* *A* *B*)
⟨*proof*⟩

interpretation *cltn2*:
action (|*carrier* = *UNIV*, *mult* = *cltn2-compose*, *one* = *cltn2-id*|) *apply-cltn2*
⟨*proof*⟩

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-transpose* (*cltn2-inverse* *A*)))

lemma *cltn2-transpose-abs*:
assumes *invertible* *M*
shows *cltn2-transpose* (*cltn2-abs* *M*) = *cltn2-abs* (*transpose* *M*)
⟨*proof*⟩

lemma *cltn2-transpose-compose*:
cltn2-transpose (*cltn2-compose* *A* *B*)
= *cltn2-compose* (*cltn2-transpose* *B*) (*cltn2-transpose* *A*)
⟨*proof*⟩

lemma *cltn2-transpose-transpose*: *cltn2-transpose* (*cltn2-transpose A*) = *A*
⟨*proof*⟩

lemma *cltn2-transpose-id* [*simp*]: *cltn2-transpose cltn2-id* = *cltn2-id*
⟨*proof*⟩

lemma *apply-cltn2-line-id* [*simp*]: *apply-cltn2-line l cltn2-id* = *l*
⟨*proof*⟩

lemma *apply-cltn2-line-compose*:
apply-cltn2-line (*apply-cltn2-line l A*) *B*
= *apply-cltn2-line l* (*cltn2-compose A B*)
⟨*proof*⟩

interpretation *cltn2-line*:
action
(|*carrier* = *UNIV*, *mult* = *cltn2-compose*, *one* = *cltn2-id*|)
apply-cltn2-line
⟨*proof*⟩

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*⟩

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

lemma *apply-cltn2-incident*:
proj2-incident p (*apply-cltn2-line l A*)
⟷ *proj2-incident* (*apply-cltn2 p* (*cltn2-inverse A*)) *l*
⟨*proof*⟩

lemma *apply-cltn2-preserve-incident* [*iff*]:
proj2-incident (*apply-cltn2 p A*) (*apply-cltn2-line l A*)
⟷ *proj2-incident p l*
⟨*proof*⟩

lemma *apply-cltn2-preserve-set-Col*:
assumes *proj2-set-Col S*
shows *proj2-set-Col* {*apply-cltn2 p C* | *p*. *p* ∈ *S*}
⟨*proof*⟩

lemma *apply-cltn2-injective*:
assumes *apply-cltn2 p C* = *apply-cltn2 q C*
shows *p* = *q*

⟨*proof*⟩

lemma *apply-cltn2-line-injective*:
assumes *apply-cltn2-line l C = apply-cltn2-line m C*
shows *l = m*
⟨*proof*⟩

lemma *apply-cltn2-line-unique*:
assumes *p ≠ q* **and** *proj2-incident p l* **and** *proj2-incident q l*
and *proj2-incident (apply-cltn2 p C) m*
and *proj2-incident (apply-cltn2 q C) m*
shows *apply-cltn2-line l C = m*
⟨*proof*⟩

lemma *apply-cltn2-unique*:
assumes *l ≠ m* **and** *proj2-incident p l* **and** *proj2-incident p m*
and *proj2-incident q (apply-cltn2-line l C)*
and *proj2-incident q (apply-cltn2-line m C)*
shows *apply-cltn2 p C = q*
⟨*proof*⟩

7.4.3 Parts of some Statements from [1]

All theorems with names beginning with *statement* are based on corresponding theorems in [1].

lemma *statement52-existence*:
fixes *a* :: *proj2^3* **and** *a3* :: *proj2*
assumes *proj2-no-3-Col (insert a3 (range (($) a)))*
shows ∃ *A. apply-cltn2 (proj2-abs (vector [1,1,1])) A = a3* ∧
(∀ *j. apply-cltn2 (proj2-abs (axis j 1)) A = a$j*)
⟨*proof*⟩

lemma *statement53-existence*:
fixes *p* :: *proj2^4^2*
assumes ∀ *i. proj2-no-3-Col (range (($) (p$i)))*
shows ∃ *C.* ∀ *j. apply-cltn2 (p0j) C = p1j*
⟨*proof*⟩

lemma *apply-cltn2-linear*:
assumes $j *_R v + k *_R w \neq 0$
shows $j *_R (v\ v{*}\ cltn2\text{-}rep\ C) + k *_R (w\ v{*}\ cltn2\text{-}rep\ C) \neq 0$
(**is** *?u ≠ 0*)
and $apply\text{-}cltn2\ (proj2\text{-}abs\ (j *_R v + k *_R w))\ C$
$= proj2\text{-}abs\ (j *_R (v\ v{*}\ cltn2\text{-}rep\ C) + k *_R (w\ v{*}\ cltn2\text{-}rep\ C))$
⟨*proof*⟩

lemma *apply-cltn2-imp-mult*:
assumes *apply-cltn2 p C = q*
shows $\exists\ k.\ k \neq 0 \wedge proj2\text{-}rep\ p\ v{*}\ cltn2\text{-}rep\ C = k *_R proj2\text{-}rep\ q$

⟨*proof*⟩

lemma *statement55*:
  assumes $p \neq q$
  and *apply-cltn2 p C* = *q*
  and *apply-cltn2 q C* = *p*
  and *proj2-incident p l*
  and *proj2-incident q l*
  and *proj2-incident r l*
  shows *apply-cltn2* (*apply-cltn2 r C*) *C* = *r*
⟨*proof*⟩

7.5 Cross ratios

definition *cross-ratio* :: *proj2* ⇒ *proj2* ⇒ *proj2* ⇒ *proj2* ⇒ *real* **where**
  cross-ratio p q r s ≜ *proj2-Col-coeff p q s* / *proj2-Col-coeff p q r*

definition *cross-ratio-correct* :: *proj2* ⇒ *proj2* ⇒ *proj2* ⇒ *proj2* ⇒ *bool* **where**
  cross-ratio-correct p q r s ≜
  proj2-set-Col {*p*,*q*,*r*,*s*} ∧ $p \neq q$ ∧ $r \neq p$ ∧ $s \neq p$ ∧ $r \neq q$

lemma *proj2-Col-coeff-abs*:
  assumes $p \neq q$ **and** $j \neq 0$
  shows *proj2-Col-coeff p q* (*proj2-abs* ($i *_R$ *proj2-rep p* + $j *_R$ *proj2-rep q*))
  = *i*/*j*
  (**is** *proj2-Col-coeff p q ?r* = *i*/*j*)
⟨*proof*⟩

lemma *proj2-set-Col-coeff*:
  assumes *proj2-set-Col S* **and** {*p*,*q*,*r*} ⊆ *S* **and** $p \neq q$ **and** $r \neq p$
  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*⟩

lemma *cross-ratio-abs*:
  fixes *u v* :: *real^3* **and** *i j k l* :: *real*
  assumes $u \neq 0$ **and** $v \neq 0$ **and** *proj2-abs u* ≠ *proj2-abs v*
  and $j \neq 0$ **and** $l \neq 0$
  shows *cross-ratio* (*proj2-abs u*) (*proj2-abs v*)
  (*proj2-abs* ($i *_R u + j *_R v$))
  (*proj2-abs* ($k *_R u + l *_R v$))
  = *j* ∗ *k* / (*i* ∗ *l*)
  (**is** *cross-ratio ?p ?q ?r ?s* = -)
⟨*proof*⟩

lemma *cross-ratio-abs2*:
  assumes $p \neq q$
  shows *cross-ratio p q*
  (*proj2-abs* ($i *_R$ *proj2-rep p* + *proj2-rep q*))

(*proj2-abs* (*j* $*_R$ *proj2-rep p* + *proj2-rep q*))
= *j*/*i*
(**is** *cross-ratio p q ?r ?s* = -)
⟨*proof*⟩

lemma *cross-ratio-correct-cltn2*:
assumes *cross-ratio-correct p q r s*
shows *cross-ratio-correct* (*apply-cltn2 p C*) (*apply-cltn2 q C*)
(*apply-cltn2 r C*) (*apply-cltn2 s C*)
(**is** *cross-ratio-correct ?pC ?qC ?rC ?sC*)
⟨*proof*⟩

lemma *cross-ratio-cltn2*:
assumes *proj2-set-Col* {*p*,*q*,*r*,*s*} **and** *p* ≠ *q* **and** *r* ≠ *p* **and** *s* ≠ *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*⟩

lemma *cross-ratio-unique*:
assumes *cross-ratio-correct p q r s* **and** *cross-ratio-correct p q r t*
and *cross-ratio p q r s* = *cross-ratio p q r t*
shows *s* = *t*
⟨*proof*⟩

lemma *cltn2-three-point-line*:
assumes *p* ≠ *q* **and** *r* ≠ *p* **and** *r* ≠ *q*
and *proj2-incident p l* **and** *proj2-incident q l* **and** *proj2-incident r l*
and *apply-cltn2 p C* = *p* **and** *apply-cltn2 q C* = *q* **and** *apply-cltn2 r C* = *r*
and *proj2-incident s l*
shows *apply-cltn2 s C* = *s* (**is** *?sC* = *s*)
⟨*proof*⟩

lemma *cross-ratio-equal-cltn2*:
assumes *cross-ratio-correct p q r s*
and *cross-ratio-correct* (*apply-cltn2 p C*) (*apply-cltn2 q C*)
(*apply-cltn2 r C*) *t*
(**is** *cross-ratio-correct ?pC ?qC ?rC t*)
and *cross-ratio* (*apply-cltn2 p C*) (*apply-cltn2 q C*) (*apply-cltn2 r C*) *t*
= *cross-ratio p q r s*
shows *t* = *apply-cltn2 s C* (**is** *t* = *?sC*)
⟨*proof*⟩

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*⟩

lemma *cross-ratio-product*:
  assumes *proj2-Col p q s* **and** $p \neq q$ **and** $s \neq p$ **and** $s \neq q$
  shows *cross-ratio p q r s* $*$ *cross-ratio p q s t* = *cross-ratio p q r t*
⟨*proof*⟩

lemma *cross-ratio-equal-1*:
  assumes *proj2-Col p q r* **and** $p \neq q$ **and** $r \neq p$ **and** $r \neq q$
  shows *cross-ratio p q r r* = *1*
⟨*proof*⟩

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*⟩

lemma *cross-ratio-swap-34*:
  shows *cross-ratio p q s r* = *1* / (*cross-ratio p q r s*)
  ⟨*proof*⟩

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*⟩

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* / (*cross-ratio p q r s*)
⟨*proof*⟩

7.6 Cartesian subspace of the real projective plane

definition *vector2-append1* :: *real^2* ⇒ *real^3* **where**
  vector2-append1 v = *vector* [*v$1*, *v$2*, *1*]

lemma *vector2-append1-non-zero*: *vector2-append1 v* ≠ *0*
⟨*proof*⟩

definition *proj2-pt* :: *real^2* ⇒ *proj2* **where**
  proj2-pt v ≜ *proj2-abs* (*vector2-append1 v*)

lemma *proj2-pt-scalar*:
  ∃ *c*. *c* ≠ *0* ∧ *proj2-rep* (*proj2-pt v*) = *c* $*_R$ *vector2-append1 v*
  ⟨*proof*⟩

abbreviation *z-non-zero* :: *proj2* ⇒ *bool* **where**
  z-non-zero p ≜ (*proj2-rep p*)*$3* ≠ *0*

definition *cart2-pt* :: *proj2* ⇒ *real^2* **where**
  cart2-pt p ≜

vector [(*proj2-rep p*)\$*1* / (*proj2-rep p*)\$*3*, (*proj2-rep p*)\$*2* / (*proj2-rep p*)\$*3*]

definition *cart2-append1* :: *proj2* ⇒ *real*^*3* **where**
cart2-append1 p ≜ (*1* / ((*proj2-rep p*)\$*3*)) $*_R$ *proj2-rep p*

lemma *cart2-append1-z*:
assumes *z-non-zero p*
shows (*cart2-append1 p*)\$*3* = *1*
⟨*proof*⟩

lemma *cart2-append1-non-zero*:
assumes *z-non-zero p*
shows *cart2-append1 p* ≠ *0*
⟨*proof*⟩

lemma *proj2-rep-cart2-append1*:
assumes *z-non-zero p*
shows *proj2-rep p* = ((*proj2-rep p*)\$*3*) $*_R$ *cart2-append1 p*
⟨*proof*⟩

lemma *proj2-abs-cart2-append1*:
assumes *z-non-zero p*
shows *proj2-abs* (*cart2-append1 p*) = *p*
⟨*proof*⟩

lemma *cart2-append1-inj*:
assumes *z-non-zero p* **and** *cart2-append1 p* = *cart2-append1 q*
shows *p* = *q*
⟨*proof*⟩

lemma *cart2-append1*:
assumes *z-non-zero p*
shows *vector2-append1* (*cart2-pt p*) = *cart2-append1 p*
⟨*proof*⟩

lemma *cart2-proj2*: *cart2-pt* (*proj2-pt v*) = *v*
⟨*proof*⟩

lemma *z-non-zero-proj2-pt*: *z-non-zero* (*proj2-pt v*)
⟨*proof*⟩

lemma *cart2-append1-proj2*: *cart2-append1* (*proj2-pt v*) = *vector2-append1 v*
⟨*proof*⟩

lemma *proj2-pt-inj*: *inj proj2-pt*
⟨*proof*⟩

lemma *proj2-cart2*:
assumes *z-non-zero p*

shows *proj2-pt* (*cart2-pt* *p*) = *p*
⟨*proof*⟩

lemma *cart2-injective*:
assumes *z-non-zero* *p* **and** *z-non-zero* *q* **and** *cart2-pt* *p* = *cart2-pt* *q*
shows *p* = *q*
⟨*proof*⟩

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*⟩

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* ⟷ *real-euclid.Col* (*cart2-pt* *p*) (*cart2-pt* *q*) (*cart2-pt* *r*)
(**is** - ⟷ *real-euclid.Col* *?a* *?b* *?c*)
⟨*proof*⟩

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*⟩

lemma *euclid-B-cart2-common-line*:
assumes *z-non-zero* *p* **and** *z-non-zero* *q* **and** *z-non-zero* *r*
and $B_{\mathbb{R}}$ (*cart2-pt* *p*) (*cart2-pt* *q*) (*cart2-pt* *r*)
(**is** $B_{\mathbb{R}}$ *?cp* *?cq* *?cr*)
shows ∃ *l*. *proj2-incident* *p* *l* ∧ *proj2-incident* *q* *l* ∧ *proj2-incident* *r* *l*
⟨*proof*⟩

lemma *cart2-append1-between*:
assumes *z-non-zero* *p* **and** *z-non-zero* *q* **and** *z-non-zero* *r*
shows $B_{\mathbb{R}}$ (*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*⟩

lemma *cart2-append1-between-right-strict*:
assumes *z-non-zero* *p* **and** *z-non-zero* *q* **and** *z-non-zero* *r*
and $B_{\mathbb{R}}$ (*cart2-pt* *p*) (*cart2-pt* *q*) (*cart2-pt* *r*) **and** $q \neq r$
shows ∃ *k*≥*0*. *k* < *1*
∧ *cart2-append1* *q* = *k* $*_R$ *cart2-append1* *r* + (*1* − *k*) $*_R$ *cart2-append1* *p*
⟨*proof*⟩

lemma *cart2-append1-between-strict*:
assumes *z-non-zero p* **and** *z-non-zero q* **and** *z-non-zero r*
and $B_{\mathbb{R}}$ (*cart2-pt p*) (*cart2-pt q*) (*cart2-pt r*) **and** $q \neq p$ **and** $q \neq r$
shows $\exists\ k>0.\ k < 1$
$\wedge$ *cart2-append1 q* = *k* $*_R$ *cart2-append1 r* + (*1* − *k*) $*_R$ *cart2-append1 p*
⟨*proof*⟩

end

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^3^3* **where**
M ≜ *vector* [
vector [*1*, *0*, *0*],
vector [*0*, *1*, *0*],
vector [*0*, *0*, −*1*]]

lemma *M-symmatrix*: *symmatrix M*
⟨*proof*⟩

lemma *M-self-inverse*: *M* ∗∗ *M* = *mat 1*
⟨*proof*⟩

lemma *M-invertible*: *invertible M*
⟨*proof*⟩

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*⟩

lemma *pole-abs*:
assumes *v* ≠ *0*
shows *pole* (*proj2-line-abs v*) = *proj2-abs* (*M* ∗*v* *v*)
⟨*proof*⟩

lemma *polar-rep-non-zero*: $M * v$ *proj2-rep* $p \neq 0$
⟨*proof*⟩

lemma *pole-polar*: *pole* (*polar* p) = p
⟨*proof*⟩

lemma *pole-rep-non-zero*: $M * v$ *proj2-line-rep* $l \neq 0$
⟨*proof*⟩

lemma *polar-pole*: *polar* (*pole* l) = l
⟨*proof*⟩

lemma *polar-inj*:
assumes *polar* p = *polar* q
shows $p = q$
⟨*proof*⟩

definition *conic-sgn* :: *proj2* ⇒ *real* **where**
conic-sgn $p \triangleq$ *sgn* (*proj2-rep* p · ($M * v$ *proj2-rep* p))

lemma *conic-sgn-abs*:
assumes $v \neq 0$
shows *conic-sgn* (*proj2-abs* v) = *sgn* (v · ($M * v\ v$))
⟨*proof*⟩

lemma *sgn-conic-sgn*: *sgn* (*conic-sgn* p) = *conic-sgn* p
⟨*proof*⟩

definition S :: *proj2 set* **where**
$S \triangleq$ {p. *conic-sgn* $p = 0$}

definition *K2* :: *proj2 set* **where**
K2 $\triangleq$ {p. *conic-sgn* $p < 0$}

lemma *S-K2-empty*: $S \cap K2 = \{\}$
⟨*proof*⟩

lemma *K2-abs*:
assumes $v \neq 0$
shows *proj2-abs* $v \in K2 \longleftrightarrow v \cdot (M * v\ v) < 0$
⟨*proof*⟩

definition *K2-centre* = *proj2-abs* (*vector* [0,0,1])

lemma *K2-centre-non-zero*: *vector* [0,0,1] ≠ (0 :: *real*^3)
⟨*proof*⟩

lemma *K2-centre-in-K2*: *K2-centre* ∈ *K2*

⟨proof⟩

lemma *K2-imp-M-neg*:
assumes $v \neq 0$ **and** *proj2-abs* $v \in K2$
shows $v \cdot (M *v\ v) < 0$
⟨proof⟩

lemma *M-neg-imp-z-squared-big*:
assumes $v \cdot (M *v\ v) < 0$
shows $(v\$3)^2 > (v\$1)^2 + (v\$2)^2$
⟨proof⟩

lemma *M-neg-imp-z-non-zero*:
assumes $v \cdot (M *v\ v) < 0$
shows $v\$3 \neq 0$
⟨proof⟩

lemma *M-neg-imp-K2*:
assumes $v \cdot (M *v\ v) < 0$
shows *proj2-abs* $v \in K2$
⟨proof⟩

lemma *M-reverse*: $a \cdot (M *v\ b) = b \cdot (M *v\ a)$
⟨proof⟩

lemma *S-abs*:
assumes $v \neq 0$
shows *proj2-abs* $v \in S \longleftrightarrow v \cdot (M *v\ v) = 0$
⟨proof⟩

lemma *S-alt-def*: $p \in S \longleftrightarrow$ *proj2-rep* $p \cdot (M *v$ *proj2-rep* $p) = 0$
⟨proof⟩

lemma *incident-polar*:
proj2-incident p (*polar* q) $\longleftrightarrow$ *proj2-rep* $p \cdot (M *v$ *proj2-rep* $q) = 0$
⟨proof⟩

lemma *incident-own-polar-in-S*: *proj2-incident* p (*polar* p) $\longleftrightarrow p \in S$
⟨proof⟩

lemma *incident-polar-swap*:
assumes *proj2-incident* p (*polar* q)
shows *proj2-incident* q (*polar* p)
⟨proof⟩

lemma *incident-pole-polar*:
assumes *proj2-incident* $p\ l$
shows *proj2-incident* (*pole* l) (*polar* p)
⟨proof⟩

definition *z-zero* :: *proj2-line* **where**
z-zero $\triangleq$ *proj2-line-abs* (*vector* [*0*,*0*,*1*])

lemma *z-zero*:
assumes (*proj2-rep p*)$*3* = *0*
shows *proj2-incident p z-zero*
⟨*proof*⟩

lemma *z-zero-conic-sgn-1*:
assumes *proj2-incident p z-zero*
shows *conic-sgn p* = *1*
⟨*proof*⟩

lemma *conic-sgn-not-1-z-non-zero*:
assumes *conic-sgn p* ≠ *1*
shows *z-non-zero p*
⟨*proof*⟩

lemma *z-zero-not-in-S*:
assumes *proj2-incident p z-zero*
shows *p* ∉ *S*
⟨*proof*⟩

lemma *line-incident-point-not-in-S*: ∃ *p*. *p* ∉ *S* ∧ *proj2-incident p l*
⟨*proof*⟩

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*⟩

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*⟩

lemma *apply-cltn2-abs-in-S*:
assumes *v* ≠ *0*
shows *apply-cltn2* (*proj2-abs v*) *C* ∈ *S*
⟷ *v* · (*cltn2-rep C* ∗∗ *M* ∗∗ *transpose* (*cltn2-rep C*) ∗*v v*) = *0*
⟨*proof*⟩

lemma *apply-cltn2-in-S*:
apply-cltn2 p C ∈ *S*
⟷ *proj2-rep p* · (*cltn2-rep C* ∗∗ *M* ∗∗ *transpose* (*cltn2-rep C*) ∗*v proj2-rep p*)
= *0*

⟨*proof*⟩

lemma *norm-M*: (*vector2-append1 v*) · (*M* ∗*v vector2-append1 v*) = (*norm v*)^2 − *1*
⟨*proof*⟩

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*,*1*,*0*])

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*,*1*,*0*] ≠ (*0* :: *real*^*3*)
  and *vector* [*1*,*0*,*0*] ≠ (*0* :: *real*^*3*)
  ⟨*proof*⟩

lemma *east-west-distinct*: *east* ≠ *west*
⟨*proof*⟩

lemma *north-south-distinct*: *north* ≠ *south*
⟨*proof*⟩

lemma *north-not-east-or-west*: *north* ∉ {*east*, *west*}
⟨*proof*⟩

lemma *compass-in-S*:
  shows *east* ∈ *S* **and** *west* ∈ *S* **and** *north* ∈ *S* **and** *south* ∈ *S*
  ⟨*proof*⟩

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*⟩

lemma *east-west-tangents-distinct*: *polar east* ≠ *polar west*
⟨*proof*⟩

lemma *east-west-tangents-incident-far-north*:
  shows *proj2-incident far-north* (*polar east*)
  and *proj2-incident far-north* (*polar west*)

⟨*proof*⟩

lemma *east-west-tangents-far-north*:
proj2-intersection (*polar east*) (*polar west*) = *far-north*
⟨*proof*⟩

instantiation *proj2* :: *zero*
begin
definition *proj2-zero-def*: *0* = *proj2-pt 0*
instance ⟨*proof*⟩
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*⟩

lemma *east-west-on-equator*:
shows *proj2-incident east equator* **and** *proj2-incident west equator*
⟨*proof*⟩

lemma *north-far-north-distinct*: *north* ≠ *far-north*
⟨*proof*⟩

lemma *north-south-far-north-on-meridian*:
shows *proj2-incident north meridian* **and** *proj2-incident south meridian*
and *proj2-incident far-north meridian*
⟨*proof*⟩

lemma *K2-centre-on-equator-meridian*:
shows *proj2-incident K2-centre equator*
and *proj2-incident K2-centre meridian*
⟨*proof*⟩

lemma *on-equator-meridian-is-K2-centre*:
assumes *proj2-incident a equator* **and** *proj2-incident a meridian*
shows *a* = *K2-centre*
⟨*proof*⟩

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*
⟨*proof*⟩

lemma *compass-reflect-invertible*:
shows *invertible rep-equator-reflect* **and** *invertible rep-meridian-reflect*
⟨*proof*⟩

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*⟩

lemma *on-equator-rep*:
assumes *z-non-zero a* **and** *proj2-incident a equator*
shows ∃ *x*. *a* = *proj2-abs* (*vector* [*x*,*0*,*1*])
⟨*proof*⟩

lemma *on-meridian-rep*:
assumes *z-non-zero a* **and** *proj2-incident a meridian*
shows ∃ *y*. *a* = *proj2-abs* (*vector* [*0*,*y*,*1*])
⟨*proof*⟩

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

abbreviation *hyp2* == *K2*

typedef *hyp2* = *K2*
⟨*proof*⟩

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*⟩

lemma *norm-eq-1-iff-in-S*:
shows *norm v = 1 ⟷ proj2-pt v ∈ S*
⟨*proof*⟩

lemma *norm-le-1-iff-in-hyp2-S*:
norm v ≤ 1 ⟷ proj2-pt v ∈ hyp2 ∪ S
⟨*proof*⟩

lemma *proj2-pt-hyp2-rep*: *proj2-pt (hyp2-rep p) = Rep-hyp2 p*
⟨*proof*⟩

lemma *hyp2-rep-abs*:
assumes *norm v < 1*
shows *hyp2-rep (hyp2-abs v) = v*
⟨*proof*⟩

lemma *hyp2-abs-rep*: *hyp2-abs (hyp2-rep p) = p*
⟨*proof*⟩

lemma *norm-hyp2-rep-lt-1*: *norm (hyp2-rep p) < 1*
⟨*proof*⟩

lemma *hyp2-S-z-non-zero*:
assumes *p ∈ hyp2 ∪ S*
shows *z-non-zero p*
⟨*proof*⟩

lemma *hyp2-S-not-equal*:
assumes *a ∈ hyp2* **and** *p ∈ S*
shows *a ≠ p*
⟨*proof*⟩

lemma *hyp2-S-cart2-inj*:
assumes *p ∈ hyp2 ∪ S* **and** *q ∈ hyp2 ∪ S* **and** *cart2-pt p = cart2-pt q*
shows *p = q*
⟨*proof*⟩

lemma *on-equator-in-hyp2-rep*:
assumes *a ∈ hyp2* **and** *proj2-incident a equator*
shows *∃ x. |x| < 1 ∧ a = proj2-abs (vector [x,0,1])*
⟨*proof*⟩

lemma *on-meridian-in-hyp2-rep*:

assumes *a* ∈ *hyp2* **and** *proj2-incident a meridian*
shows ∃ *y*. $|y| < 1$ ∧ *a* = *proj2-abs* (*vector* [*0*,*y*,*1*])
⟨*proof*⟩

definition *hyp2-cltn2* :: *hyp2* ⇒ *cltn2* ⇒ *hyp2* **where**
hyp2-cltn2 p A ≜ *Abs-hyp2* (*apply-cltn2* (*Rep-hyp2 p*) *A*)

definition *is-K2-isometry* :: *cltn2* ⇒ *bool* **where**
is-K2-isometry J ≜ (∀ *p*. *apply-cltn2 p J* ∈ *S* ⟷ *p* ∈ *S*)

lemma *cltn2-id-is-K2-isometry*: *is-K2-isometry cltn2-id*
⟨*proof*⟩

lemma *J-M-J-transpose-K2-isometry*:
assumes $k \neq 0$
and *repJ* ** *M* ** *transpose repJ* = $k *_R$ *M* (**is** *?N* = -)
shows *is-K2-isometry* (*cltn2-abs repJ*) (**is** *is-K2-isometry ?J*)
⟨*proof*⟩

lemma *equator-reflect-K2-isometry*:
shows *is-K2-isometry equator-reflect*
⟨*proof*⟩

lemma *meridian-reflect-K2-isometry*:
shows *is-K2-isometry meridian-reflect*
⟨*proof*⟩

lemma *cltn2-compose-is-K2-isometry*:
assumes *is-K2-isometry H* **and** *is-K2-isometry J*
shows *is-K2-isometry* (*cltn2-compose H J*)
⟨*proof*⟩

lemma *cltn2-inverse-is-K2-isometry*:
assumes *is-K2-isometry J*
shows *is-K2-isometry* (*cltn2-inverse J*)
⟨*proof*⟩

interpretation *K2-isometry-subgroup*: *subgroup*
Collect is-K2-isometry
(|*carrier* = *UNIV*, *mult* = *cltn2-compose*, *one* = *cltn2-id*|)
⟨*proof*⟩

interpretation *K2-isometry*: *group*
(|*carrier* = *Collect is-K2-isometry*, *mult* = *cltn2-compose*, *one* = *cltn2-id*|)
⟨*proof*⟩

lemma *K2-isometry-inverse-inv* [*simp*]:
assumes *is-K2-isometry J*
shows $inv_{(|carrier\ =\ Collect\ is\text{-}K2\text{-}isometry,\ mult\ =\ cltn2\text{-}compose,\ one\ =\ cltn2\text{-}id|)}$

J
  = *cltn2-inverse J*
  ⟨*proof*⟩

definition *real-hyp2-C* :: [*hyp2*, *hyp2*, *hyp2*, *hyp2*] ⇒ *bool*
  (- - $\equiv_K$ - - [*99*,*99*,*99*,*99*] *50*) **where**
  p q $\equiv_K$ *r s* ≜
    (∃ *A*. *is-K2-isometry A* ∧ *hyp2-cltn2 p A* = *r* ∧ *hyp2-cltn2 q A* = *s*)

definition *real-hyp2-B* :: [*hyp2*, *hyp2*, *hyp2*] ⇒ *bool*
(B_K - - - [*99*,*99*,*99*] *50*) **where**
  B_K *p q r* ≜ $B_{\mathbb{R}}$ (*hyp2-rep p*) (*hyp2-rep q*) (*hyp2-rep r*)

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

lemma *collinear-quadratic*:
  assumes *t* = *i* $*_R$ *a* + *r*
  shows *t* · (*M* ∗*v t*) =
  (*a* · (*M* ∗*v a*)) ∗ i^2 + *2* ∗ (*a* · (*M* ∗*v r*)) ∗ *i* + *r* · (*M* ∗*v r*)
⟨*proof*⟩

lemma *S-quadratic′*:
  assumes *p* ≠ *0* **and** *q* ≠ *0* **and** *proj2-abs p* ≠ *proj2-abs q*
  shows *proj2-abs* (*k* $*_R$ *p* + *q*) ∈ *S*
  ⟷ *p* · (*M* ∗*v p*) ∗ k^2 + *p* · (*M* ∗*v q*) ∗ *2* ∗ *k* + *q* · (*M* ∗*v q*) = *0*
⟨*proof*⟩

lemma *S-quadratic*:
  assumes *p* ≠ *q* **and** *r* = *proj2-abs* (*k* $*_R$ *proj2-rep p* + *proj2-rep q*)
  shows *r* ∈ *S*
  ⟷ *proj2-rep p* · (*M* ∗*v proj2-rep p*) ∗ k^2
      + *proj2-rep p* · (*M* ∗*v proj2-rep q*) ∗ *2* ∗ *k*
      + *proj2-rep q* · (*M* ∗*v proj2-rep q*)
    = *0*
⟨*proof*⟩

definition *quarter-discrim* :: *real^3* ⇒ *real^3* ⇒ *real* **where**
  quarter-discrim p q ≜ $(p \cdot (M *v\ q))^2$ − *p* · (*M* ∗*v p*) ∗ (*q* · (*M* ∗*v q*))

lemma *quarter-discrim-invariant*:
  assumes *t* = *i* $*_R$ *a* + *r*
  shows *quarter-discrim a t* = *quarter-discrim a r*
⟨*proof*⟩

lemma *quarter-discrim-positive*:
  assumes *p* ≠ *0* **and** *q* ≠ *0* **and** *proj2-abs p* ≠ *proj2-abs q* (**is** *?pp* ≠ *?pq*)
  and *proj2-abs p* ∈ *K2*
  shows *quarter-discrim p q* > *0*
⟨*proof*⟩

lemma *quarter-discrim-self-zero*:
  assumes *proj2-abs a = proj2-abs b*
  shows *quarter-discrim a b = 0*
⟨*proof*⟩

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 ∈ K2*
  shows *S-intersection-coeff1 p q ≠ S-intersection-coeff2 p q*
⟨*proof*⟩

lemma *S-intersections-distinct*:
  assumes *p ≠ 0* **and** *q ≠ 0* **and** *proj2-abs p ≠ proj2-abs q* (**is** *?pp ≠ ?pq*)
  and *proj2-abs p ∈ K2*
  shows *S-intersection1 p q ≠ S-intersection2 p q*
⟨*proof*⟩

lemma *S-intersections-in-S*:
  assumes *p ≠ 0* **and** *q ≠ 0* **and** *proj2-abs p ≠ proj2-abs q* (**is** *?pp ≠ ?pq*)
  and *proj2-abs p ∈ K2*
  shows *S-intersection1 p q ∈ S* **and** *S-intersection2 p q ∈ S*

⟨*proof*⟩

lemma *S-intersections-Col*:
assumes $p \neq 0$ **and** $q \neq 0$
shows *proj2-Col* (*proj2-abs p*) (*proj2-abs q*) (*S-intersection1 p q*)
(**is** *proj2-Col ?pp ?pq ?pr*)
and *proj2-Col* (*proj2-abs p*) (*proj2-abs q*) (*S-intersection2 p q*)
(**is** *proj2-Col ?pp ?pq ?ps*)
⟨*proof*⟩

lemma *S-intersections-incident*:
assumes $p \neq 0$ **and** $q \neq 0$ **and** *proj2-abs p* $\neq$ *proj2-abs q* (**is** *?pp* $\neq$ *?pq*)
and *proj2-incident* (*proj2-abs p*) *l* **and** *proj2-incident* (*proj2-abs q*) *l*
shows *proj2-incident* (*S-intersection1 p q*) *l* (**is** *proj2-incident ?pr l*)
and *proj2-incident* (*S-intersection2 p q*) *l* (**is** *proj2-incident ?ps l*)
⟨*proof*⟩

lemma *K2-line-intersect-twice*:
assumes $a \in K2$ **and** $a \neq r$
shows $\exists$ *s u*. $s \neq u \land s \in S \land u \in S \land$ *proj2-Col a r s* $\land$ *proj2-Col a r u*
⟨*proof*⟩

lemma *point-in-S-polar-is-tangent*:
assumes $p \in S$ **and** $q \in S$ **and** *proj2-incident q* (*polar p*)
shows $q = p$
⟨*proof*⟩

lemma *line-through-K2-intersect-S-twice*:
assumes $p \in K2$ **and** *proj2-incident p l*
shows $\exists$ *q r*. $q \neq r \land q \in S \land r \in S \land$ *proj2-incident q l* $\land$ *proj2-incident r l*
⟨*proof*⟩

lemma *line-through-K2-intersect-S-again*:
assumes $p \in K2$ **and** *proj2-incident p l*
shows $\exists$ *r*. $r \neq q \land r \in S \land$ *proj2-incident r l*
⟨*proof*⟩

lemma *line-through-K2-intersect-S*:
assumes $p \in K2$ **and** *proj2-incident p l*
shows $\exists$ *r*. $r \in S \land$ *proj2-incident r l*
⟨*proof*⟩

lemma *line-intersect-S-at-most-twice*:
$\exists$ *p q*. $\forall$ *r*$\in$*S*. *proj2-incident r l* $\longrightarrow r = p \lor r = q$
⟨*proof*⟩

lemma *card-line-intersect-S*:
assumes $T \subseteq S$ **and** *proj2-set-Col T*
shows *card* $T \leq 2$

⟨*proof*⟩

lemma *line-S-two-intersections-only*:
assumes $p \neq q$ **and** $p \in S$ **and** $q \in S$ **and** $r \in S$
and *proj2-incident p l* **and** *proj2-incident q l* **and** *proj2-incident r l*
shows $r = p \lor r = q$
⟨*proof*⟩

lemma *line-through-K2-intersect-S-exactly-twice*:
assumes $p \in K2$ **and** *proj2-incident p l*
shows $\exists\ q\ r.\ q \neq r \land q \in S \land r \in S \land$ *proj2-incident q l* $\land$ *proj2-incident r l*
$\land$ ($\forall\ s{\in}S.$ *proj2-incident s l* $\longrightarrow s = q \lor s = r$)
⟨*proof*⟩

lemma *tangent-not-through-K2*:
assumes $p \in S$ **and** $q \in K2$
shows $\neg$ *proj2-incident q* (*polar p*)
⟨*proof*⟩

lemma *outside-exists-line-not-intersect-S*:
assumes *conic-sgn p* = *1*
shows $\exists\ l.$ *proj2-incident p l* $\land$ ($\forall\ q.$ *proj2-incident q l* $\longrightarrow q \notin S$)
⟨*proof*⟩

lemma *lines-through-intersect-S-twice-in-K2*:
assumes $\forall\ l.$ *proj2-incident p l*
$\longrightarrow$ ($\exists\ q\ r.\ q \neq r \land q \in S \land r \in S \land$ *proj2-incident q l* $\land$ *proj2-incident r l*)
shows $p \in K2$
⟨*proof*⟩

lemma *line-through-hyp2-pole-not-in-hyp2*:
assumes $a \in hyp2$ **and** *proj2-incident a l*
shows *pole l* $\notin$ *hyp2*
⟨*proof*⟩

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*⟩

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*⟩

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*⟩

lemma *cart2-append1-apply-cltn2*:
assumes $p \in hyp2 \cup S$ **and** *is-K2-isometry J*
shows $\exists\ k.\ k \neq 0$
$\wedge$ *cart2-append1 p v* cltn2-rep J* $=$ $k *_R$ *cart2-append1* (*apply-cltn2 p J*)
⟨*proof*⟩

8.5 The K-isometries form a group action

lemma *hyp2-cltn2-id* [*simp*]: *hyp2-cltn2 p cltn2-id* $=$ *p*
⟨*proof*⟩

lemma *apply-cltn2-Rep-hyp2*:
assumes *is-K2-isometry J*
shows *apply-cltn2* (*Rep-hyp2 p*) *J* $\in$ *hyp2*
⟨*proof*⟩

lemma *Rep-hyp2-cltn2*:
assumes *is-K2-isometry J*
shows *Rep-hyp2* (*hyp2-cltn2 p J*) $=$ *apply-cltn2* (*Rep-hyp2 p*) *J*
⟨*proof*⟩

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*⟩

interpretation *K2-isometry*: *action*
(|*carrier* = *Collect is-K2-isometry*, *mult* = *cltn2-compose*, *one* = *cltn2-id*|)
hyp2-cltn2
⟨*proof*⟩

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

lemma *three-in-S-tangent-intersection-no-3-Col*:
assumes $p \in S$ **and** $q \in S$ **and** $r \in S$
and $p \neq q$ **and** $r \notin \{p,q\}$
shows *proj2-no-3-Col* {*proj2-intersection* (*polar p*) (*polar q*),*r*,*p*,*q*}
(**is** *proj2-no-3-Col* {*?s*,*r*,*p*,*q*})
⟨*proof*⟩

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*. *is-K2-isometry J*
$\wedge$ *apply-cltn2 east J* $=$ *p*
$\wedge$ *apply-cltn2 west J* $=$ *q*
$\wedge$ *apply-cltn2 north J* $=$ *r*
$\wedge$ *apply-cltn2 far-north J* $=$ *proj2-intersection* (*polar p*) (*polar q*)

⟨*proof*⟩

lemma *statement66-existence*:
assumes $a1 \in K2$ **and** $a2 \in K2$ **and** $p1 \in S$ **and** $p2 \in S$
shows $\exists\ J.\ is\text{-}K2\text{-}isometry\ J \wedge apply\text{-}cltn2\ a1\ J = a2 \wedge apply\text{-}cltn2\ p1\ J = p2$
⟨*proof*⟩

lemma *K2-isometry-swap*:
assumes $a \in hyp2$ **and** $b \in hyp2$
shows $\exists\ J.\ is\text{-}K2\text{-}isometry\ J \wedge apply\text{-}cltn2\ a\ J = b \wedge apply\text{-}cltn2\ b\ J = a$
⟨*proof*⟩

theorem *hyp2-axiom1*: $\forall\ a\ b.\ a\ b \equiv_K\ b\ a$
⟨*proof*⟩

theorem *hyp2-axiom2*: $\forall\ a\ b\ p\ q\ r\ s.\ a\ b \equiv_K\ p\ q \wedge a\ b \equiv_K\ r\ s \longrightarrow p\ q \equiv_K\ r\ s$
⟨*proof*⟩

theorem *hyp2-axiom3*: $\forall\ a\ b\ c.\ a\ b \equiv_K\ c\ c \longrightarrow a = b$
⟨*proof*⟩

interpretation *hyp2*: *tarski-first3 real-hyp2-C*
⟨*proof*⟩

8.7 Some lemmas about betweenness

lemma *S-at-edge*:
assumes $p \in S$ **and** $q \in hyp2 \cup S$ **and** $r \in hyp2 \cup S$ **and** *proj2-Col p q r*
shows $B_{\mathbb{R}}$ (*cart2-pt p*) (*cart2-pt q*) (*cart2-pt r*)
$\vee$ $B_{\mathbb{R}}$ (*cart2-pt p*) (*cart2-pt r*) (*cart2-pt q*)
(**is** $B_{\mathbb{R}}$ *?cp ?cq ?cr* $\vee$ -)
⟨*proof*⟩

lemma *hyp2-in-middle*:
assumes $p \in S$ **and** $q \in S$ **and** $r \in hyp2 \cup S$ **and** *proj2-Col p q r*
and $p \neq q$
shows $B_{\mathbb{R}}$ (*cart2-pt p*) (*cart2-pt r*) (*cart2-pt q*) (**is** $B_{\mathbb{R}}$ *?cp ?cr ?cq*)
⟨*proof*⟩

lemma *hyp2-incident-in-middle*:
assumes $p \neq q$ **and** $p \in S$ **and** $q \in S$ **and** $a \in hyp2 \cup S$
and *proj2-incident p l* **and** *proj2-incident q l* **and** *proj2-incident a l*
shows $B_{\mathbb{R}}$ (*cart2-pt p*) (*cart2-pt a*) (*cart2-pt q*)
⟨*proof*⟩

lemma *extend-to-S*:
assumes $p \in hyp2 \cup S$ **and** $q \in hyp2 \cup S$
shows $\exists\ r \in S.\ B_{\mathbb{R}}$ (*cart2-pt p*) (*cart2-pt q*) (*cart2-pt r*)
(**is** $\exists\ r \in S.\ B_{\mathbb{R}}$ *?cp ?cq* (*cart2-pt r*))

⟨*proof*⟩

definition *endpoint-in-S* :: *proj2* ⇒ *proj2* ⇒ *proj2* **where**
endpoint-in-S a b
$\triangleq$ ϵ *p*. *p*∈*S* ∧ $B_{\mathbb{R}}$ (*cart2-pt a*) (*cart2-pt b*) (*cart2-pt p*)

lemma *endpoint-in-S*:
assumes *a* ∈ *hyp2* ∪ *S* **and** *b* ∈ *hyp2* ∪ *S*
shows *endpoint-in-S a b* ∈ *S* (**is** *?p* ∈ *S*)
and $B_{\mathbb{R}}$ (*cart2-pt a*) (*cart2-pt b*) (*cart2-pt* (*endpoint-in-S a b*))
(**is** $B_{\mathbb{R}}$ *?ca ?cb ?cp*)
⟨*proof*⟩

lemma *endpoint-in-S-swap*:
assumes *a* ≠ *b* **and** *a* ∈ *hyp2* ∪ *S* **and** *b* ∈ *hyp2* ∪ *S*
shows *endpoint-in-S a b* ≠ *endpoint-in-S b a* (**is** *?p* ≠ *?q*)
⟨*proof*⟩

lemma *endpoint-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*⟩

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*⟩

lemma *endpoint-in-S-unique*:
assumes *a* ≠ *b* **and** *a* ∈ *hyp2* ∪ *S* **and** *b* ∈ *hyp2* ∪ *S* **and** *p* ∈ *S*
and $B_{\mathbb{R}}$ (*cart2-pt a*) (*cart2-pt b*) (*cart2-pt p*) (**is** $B_{\mathbb{R}}$ *?ca ?cb ?cp*)
shows *p* = *endpoint-in-S a b* (**is** *p* = *?q*)
⟨*proof*⟩

lemma *between-hyp2-S*:
assumes *p* ∈ *hyp2* ∪ *S* **and** *r* ∈ *hyp2* ∪ *S* **and** $k \geq 0$ **and** $k \leq 1$
shows *proj2-pt* ($k *_R$ (*cart2-pt r*) + ($1 - k$) $*_R$ (*cart2-pt p*)) ∈ *hyp2* ∪ *S*
(**is** *proj2-pt ?cq* ∈ -)
⟨*proof*⟩

8.8 The Klein–Beltrami model satisfies axiom 4

definition *expansion-factor* :: *proj2* ⇒ *cltn2* ⇒ *real* **where**
expansion-factor p J $\triangleq$ (*cart2-append1 p v∗ cltn2-rep J*)$*3*

lemma *expansion-factor*:

assumes $p \in hyp2 \cup S$ **and** *is-K2-isometry J*
shows *expansion-factor p J* $\neq 0$
and *cart2-append1 p v∗ cltn2-rep J*
$=$ *expansion-factor p J* $*_R$ *cart2-append1* (*apply-cltn2 p J*)
⟨*proof*⟩

lemma *expansion-factor-linear-apply-cltn2*:
assumes $p \in hyp2 \cup S$ **and** $q \in hyp2 \cup S$ **and** $r \in hyp2 \cup S$
and *is-K2-isometry J*
and *cart2-pt r* $= k *_R$ *cart2-pt p* $+ (1 - k) *_R$ *cart2-pt q*
shows *expansion-factor r J* $*_R$ *cart2-append1* (*apply-cltn2 r J*)
$= (k *$ *expansion-factor p J*$) *_R$ *cart2-append1* (*apply-cltn2 p J*)
$+ ((1 - k) *$ *expansion-factor q J*$) *_R$ *cart2-append1* (*apply-cltn2 q J*)
(**is** *?er* $*_R$ - = ($k *$ *?ep*) $*_R$ - + (($1 - k$) $*$ *?eq*) $*_R$ -)
⟨*proof*⟩

lemma *expansion-factor-linear*:
assumes $p \in hyp2 \cup S$ **and** $q \in hyp2 \cup S$ **and** $r \in hyp2 \cup S$
and *is-K2-isometry J*
and *cart2-pt r* $= k *_R$ *cart2-pt p* $+ (1 - k) *_R$ *cart2-pt q*
shows *expansion-factor r J*
$= k *$ *expansion-factor p J* $+ (1 - k) *$ *expansion-factor q J*
(**is** *?er* $= k *$ *?ep* $+ (1 - k) *$ *?eq*)
⟨*proof*⟩

lemma *expansion-factor-sgn-invariant*:
assumes $p \in hyp2 \cup S$ **and** $q \in hyp2 \cup S$ **and** *is-K2-isometry J*
shows *sgn* (*expansion-factor p J*) = *sgn* (*expansion-factor q J*)
(**is** *sgn ?ep* = *sgn ?eq*)
⟨*proof*⟩

lemma *statement-63*:
assumes $p \in hyp2 \cup S$ **and** $q \in hyp2 \cup S$ **and** $r \in hyp2 \cup S$
and *is-K2-isometry J* **and** $B_{\mathbb{R}}$ (*cart2-pt p*) (*cart2-pt q*) (*cart2-pt r*)
shows $B_{\mathbb{R}}$
(*cart2-pt* (*apply-cltn2 p J*))
(*cart2-pt* (*apply-cltn2 q J*))
(*cart2-pt* (*apply-cltn2 r J*))
⟨*proof*⟩

theorem *hyp2-axiom4*: $\forall\ q\ a\ b\ c.\ \exists\ x.\ B_K\ q\ a\ x \wedge a\ x \equiv_K b\ c$
⟨*proof*⟩

8.9 More betweenness theorems

lemma *hyp2-S-points-fix-line*:
assumes $a \in hyp2$ **and** $p \in S$ **and** *is-K2-isometry J*
and *apply-cltn2 a J* $= a$ (**is** *?aJ* $= a$)
and *apply-cltn2 p J* $= p$ (**is** *?pJ* $= p$)

and *proj2-incident a l* **and** *proj2-incident p l* **and** *proj2-incident b l*
shows *apply-cltn2 b J* = *b* (**is** *?bJ* = *b*)
⟨*proof*⟩

lemma *K2-isometry-endpoint-in-S*:
assumes $a \neq b$ **and** $a \in$ *hyp2* $\cup$ *S* **and** $b \in$ *hyp2* $\cup$ *S* **and** *is-K2-isometry J*
shows *apply-cltn2* (*endpoint-in-S a b*) *J*
= *endpoint-in-S* (*apply-cltn2 a J*) (*apply-cltn2 b J*)
(**is** *?pJ* = *endpoint-in-S ?aJ ?bJ*)
⟨*proof*⟩

lemma *between-endpoint-in-S*:
assumes $a \neq b$ **and** $b \neq c$
and $a \in$ *hyp2* $\cup$ *S* **and** $b \in$ *hyp2* $\cup$ *S* **and** $c \in$ *hyp2* $\cup$ *S*
and $B_{\mathbb{R}}$ (*cart2-pt a*) (*cart2-pt b*) (*cart2-pt c*) (**is** $B_{\mathbb{R}}$ *?ca ?cb ?cc*)
shows *endpoint-in-S a b* = *endpoint-in-S b c* (**is** *?p* = *?q*)
⟨*proof*⟩

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*⟩

lemma *line-S-match-intersections*:
assumes $p \neq q$ **and** $r \neq s$ **and** $p \in S$ **and** $q \in S$ **and** $r \in S$ **and** $s \in S$
and *proj2-set-Col* {*p*,*q*,*r*,*s*}
shows $(p = r \land q = s) \lor (q = r \land p = s)$
⟨*proof*⟩

definition *are-endpoints-in-S* :: [*proj2*, *proj2*, *proj2*, *proj2*] ⇒ *bool* **where**
are-endpoints-in-S p q a b
≜ $p \neq q \land p \in S \land q \in S \land a \in$ *hyp2* $\land\ b \in$ *hyp2* $\land$ *proj2-set-Col* {*p*,*q*,*a*,*b*}

lemma *are-endpoints-in-S′*:
assumes $p \neq q$ **and** $a \neq b$ **and** $p \in S$ **and** $q \in S$ **and** $a \in$ *hyp2* $\cup$ *S*
and $b \in$ *hyp2* $\cup$ *S* **and** *proj2-set-Col* {*p*,*q*,*a*,*b*}
shows (*p* = *endpoint-in-S a b* ∧ *q* = *endpoint-in-S b a*)
∨ (*q* = *endpoint-in-S a b* ∧ *p* = *endpoint-in-S b a*)
(**is** (*p* = *?r* ∧ *q* = *?s*) ∨ (*q* = *?r* ∧ *p* = *?s*))
⟨*proof*⟩

lemma *are-endpoints-in-S*:
assumes $a \neq b$ **and** *are-endpoints-in-S p q a b*
shows (*p* = *endpoint-in-S a b* ∧ *q* = *endpoint-in-S b a*)
∨ (*q* = *endpoint-in-S a b* ∧ *p* = *endpoint-in-S b a*)
⟨*proof*⟩

lemma *S-intersections-endpoints-in-S*:
assumes $a \neq 0$ **and** $b \neq 0$ **and** *proj2-abs a* ≠ *proj2-abs b* (**is** *?pa* ≠ *?pb*)

and *proj2-abs a* ∈ *hyp2* **and** *proj2-abs b* ∈ *hyp2* ∪ *S*
shows (*S-intersection1 a b* = *endpoint-in-S ?pa ?pb*
∧ *S-intersection2 a b* = *endpoint-in-S ?pb ?pa*)
∨ (*S-intersection2 a b* = *endpoint-in-S ?pa ?pb*
∧ *S-intersection1 a b* = *endpoint-in-S ?pb ?pa*)
(**is** (*?pp* = *?pr* ∧ *?pq* = *?ps*) ∨ (*?pq* = *?pr* ∧ *?pp* = *?ps*))
⟨*proof*⟩

lemma *between-endpoints-in-S*:
assumes *a* ≠ *b* **and** *a* ∈ *hyp2* ∪ *S* **and** *b* ∈ *hyp2* ∪ *S*
shows $B_{\mathbb{R}}$
(*cart2-pt* (*endpoint-in-S a b*)) (*cart2-pt a*) (*cart2-pt* (*endpoint-in-S b a*))
(**is** $B_{\mathbb{R}}$ *?cp ?ca ?cq*)
⟨*proof*⟩

lemma *S-hyp2-S-cart2-append1*:
assumes *p* ≠ *q* **and** *p* ∈ *S* **and** *q* ∈ *S* **and** *a* ∈ *hyp2*
and *proj2-incident p l* **and** *proj2-incident q l* **and** *proj2-incident a l*
shows ∃ *k*. *k* > *0* ∧ *k* < *1*
∧ *cart2-append1 a* = *k* $*_R$ *cart2-append1 q* + (*1* − *k*) $*_R$ *cart2-append1 p*
⟨*proof*⟩

lemma *are-endpoints-in-S-swap-34*:
assumes *are-endpoints-in-S p q a b*
shows *are-endpoints-in-S p q b a*
⟨*proof*⟩

lemma *proj2-set-Col-endpoints-in-S*:
assumes *a* ≠ *b* **and** *a* ∈ *hyp2* ∪ *S* **and** *b* ∈ *hyp2* ∪ *S*
shows *proj2-set-Col* {*endpoint-in-S a b*, *endpoint-in-S b a*, *a*, *b*}
(**is** *proj2-set-Col* {*?p*,*?q*,*a*,*b*})
⟨*proof*⟩

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*⟩

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*⟩

lemma *endpoint-in-S-S-strict-hyp2-distinct*:
assumes *p* ∈ *S* **and** *a* ∈ *hyp2*
shows *endpoint-in-S p a* ≠ *p*
⟨*proof*⟩

lemma *end-and-opposite-are-endpoints-in-S*:
assumes $a \in hyp2$ **and** $b \in hyp2$ **and** $p \in S$
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*
(**is** *are-endpoints-in-S p ?q a b*)
⟨*proof*⟩

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*⟩

lemma *real-hyp2-C-hyp2-cltn2*:
assumes *is-K2-isometry J*
shows *a b* $\equiv_K$ (*hyp2-cltn2 a J*) (*hyp2-cltn2 b J*) (**is** *a b* $\equiv_K$ *?aJ ?bJ*)
⟨*proof*⟩

8.10 Perpendicularity

definition *M-perp* :: *proj2-line* ⇒ *proj2-line* ⇒ *bool* **where**
M-perp l m ≜ *proj2-incident* (*pole l*) *m*

lemma *M-perp-sym*:
assumes *M-perp l m*
shows *M-perp m l*
⟨*proof*⟩

lemma *M-perp-to-compass*:
assumes *M-perp l m* **and** $a \in hyp2$ **and** *proj2-incident a l*
and $b \in hyp2$ **and** *proj2-incident b m*
shows ∃ *J*. *is-K2-isometry J*
∧ *apply-cltn2-line equator J* = *l* ∧ *apply-cltn2-line meridian J* = *m*
⟨*proof*⟩

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*)
⟨*proof*⟩

lemma *drop-perp-perp*: *M-perp l* (*drop-perp p l*)
⟨*proof*⟩

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*)
⟨*proof*⟩

lemma *M-perp-hyp2*:
assumes *M-perp l m* **and** $a \in$ *hyp2* **and** *proj2-incident a l* **and** $b \in$ *hyp2*
and *proj2-incident b m* **and** *proj2-incident c l* **and** *proj2-incident c m*
shows $c \in$ *hyp2*
⟨*proof*⟩

lemma *perp-foot-hyp2*:
assumes $a \in$ *hyp2* **and** *proj2-incident a l* **and** $b \in$ *hyp2*
shows *perp-foot b l* $\in$ *hyp2*
⟨*proof*⟩

definition *perp-up* :: *proj2* ⇒ *proj2-line* ⇒ *proj2* **where**
perp-up a l
≜ *if proj2-incident a l then* ϵ *p. p* $\in S \wedge$ *proj2-incident p* (*drop-perp a l*)
else endpoint-in-S (*perp-foot a l*) *a*

lemma *perp-up-degenerate-in-S-incident*:
assumes $a \in$ *hyp2* **and** *proj2-incident a l*
shows *perp-up a l* $\in S$ (**is** *?p* $\in S$)
and *proj2-incident* (*perp-up a l*) (*drop-perp a l*)
⟨*proof*⟩

lemma *perp-up-non-degenerate-in-S-at-end*:
assumes $a \in$ *hyp2* **and** $b \in$ *hyp2* **and** *proj2-incident b l*
and ¬ *proj2-incident a l*
shows *perp-up a l* $\in S$
and $B_{\mathbb{R}}$ (*cart2-pt* (*perp-foot a l*)) (*cart2-pt a*) (*cart2-pt* (*perp-up a l*))
⟨*proof*⟩

lemma *perp-up-in-S*:
assumes $a \in$ *hyp2* **and** $b \in$ *hyp2* **and** *proj2-incident b l*
shows *perp-up a l* $\in S$
⟨*proof*⟩

lemma *perp-up-incident*:
assumes $a \in$ *hyp2* **and** $b \in$ *hyp2* **and** *proj2-incident b l*
shows *proj2-incident* (*perp-up a l*) (*drop-perp a l*)
(**is** *proj2-incident ?p ?m*)
⟨*proof*⟩

lemma *drop-perp-same-line-pole-in-S*:
assumes *drop-perp p l* = *l*
shows *pole l* $\in S$
⟨*proof*⟩

lemma *hyp2-drop-perp-not-same-line*:

assumes *a* ∈ *hyp2*
shows *drop-perp a l* ≠ *l*
⟨*proof*⟩

lemma *hyp2-incident-perp-foot-same-point*:
assumes *a* ∈ *hyp2* **and** *proj2-incident a l*
shows *perp-foot a l* = *a*
⟨*proof*⟩

lemma *perp-up-at-end*:
assumes *a* ∈ *hyp2* **and** *b* ∈ *hyp2* **and** *proj2-incident b l*
shows $B_{\mathbb{R}}$ (*cart2-pt* (*perp-foot a l*)) (*cart2-pt a*) (*cart2-pt* (*perp-up a l*))
⟨*proof*⟩

definition *perp-down* :: *proj2* ⇒ *proj2-line* ⇒ *proj2* **where**
perp-down a l ≜ *endpoint-in-S* (*perp-up a l*) *a*

lemma *perp-down-in-S*:
assumes *a* ∈ *hyp2* **and** *b* ∈ *hyp2* **and** *proj2-incident b l*
shows *perp-down a l* ∈ *S*
⟨*proof*⟩

lemma *perp-down-incident*:
assumes *a* ∈ *hyp2* **and** *b* ∈ *hyp2* **and** *proj2-incident b l*
shows *proj2-incident* (*perp-down a l*) (*drop-perp a l*)
⟨*proof*⟩

lemma *perp-up-down-distinct*:
assumes *a* ∈ *hyp2* **and** *b* ∈ *hyp2* **and** *proj2-incident b l*
shows *perp-up a l* ≠ *perp-down a l*
⟨*proof*⟩

lemma *perp-up-down-foot-are-endpoints-in-S*:
assumes *a* ∈ *hyp2* **and** *b* ∈ *hyp2* **and** *proj2-incident b l*
shows *are-endpoints-in-S* (*perp-up a l*) (*perp-down a l*) (*perp-foot a l*) *a*
⟨*proof*⟩

lemma *perp-foot-opposite-endpoint-in-S*:
assumes *a* ∈ *hyp2* **and** *b* ∈ *hyp2* **and** *c* ∈ *hyp2* **and** *a* ≠ *b*
shows
endpoint-in-S (*endpoint-in-S a b*) (*perp-foot c* (*proj2-line-through a b*))
= *endpoint-in-S b a*
(**is** *endpoint-in-S ?p ?d* = *endpoint-in-S b a*)
⟨*proof*⟩

lemma *endpoints-in-S-perp-foot-are-endpoints-in-S*:
assumes *a* ∈ *hyp2* **and** *b* ∈ *hyp2* **and** *c* ∈ *hyp2* **and** *a* ≠ *b*
and *proj2-incident a l* **and** *proj2-incident b l*
shows *are-endpoints-in-S*

$(endpoint\text{-}in\text{-}S\ a\ b)\ (endpoint\text{-}in\text{-}S\ b\ a)\ a\ (perp\text{-}foot\ c\ l)$
⟨*proof*⟩

definition *right-angle* :: *proj2* ⇒ *proj2* ⇒ *proj2* ⇒ *bool* **where**
right-angle p a q
$\triangleq p \in S \wedge q \in S \wedge a \in hyp2$
$\wedge$ *M-perp* (*proj2-line-through p a*) (*proj2-line-through a q*)

lemma *perp-foot-up-right-angle*:
assumes $p \in S$ **and** $a \in hyp2$ **and** $b \in hyp2$ **and** *proj2-incident p l*
and *proj2-incident b l*
shows *right-angle p* (*perp-foot a l*) (*perp-up a l*)
⟨*proof*⟩

lemma *M-perp-unique*:
assumes $a \in hyp2$ **and** $b \in hyp2$ **and** *proj2-incident a l*
and *proj2-incident b m* **and** *proj2-incident b n* **and** *M-perp l m*
and *M-perp l n*
shows $m = n$
⟨*proof*⟩

lemma *perp-foot-eq-implies-drop-perp-eq*:
assumes $a \in hyp2$ **and** $b \in hyp2$ **and** *proj2-incident a l*
and *perp-foot b l* = *perp-foot c l*
shows *drop-perp b l* = *drop-perp c l*
⟨*proof*⟩

lemma *right-angle-to-compass*:
assumes *right-angle p a q*
shows ∃ *J*. *is-K2-isometry J* ∧ *apply-cltn2 p J* = *east*
∧ *apply-cltn2 a J* = *K2-centre* ∧ *apply-cltn2 q J* = *north*
⟨*proof*⟩

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*⟩

8.11 Functions of distance

definition *exp-2dist* :: *proj2* ⇒ *proj2* ⇒ *real* **where**
exp-2dist a b
≜ *if* $a = b$
then 1
else cross-ratio (*endpoint-in-S a b*) (*endpoint-in-S b a*) *a b*

definition *cosh-dist* :: *proj2* ⇒ *proj2* ⇒ *real* **where**
cosh-dist a b ≜ (*sqrt* (*exp-2dist a b*) + *sqrt* (*1* / (*exp-2dist a b*))) / *2*

lemma *exp-2dist-formula*:
assumes $a \neq 0$ **and** $b \neq 0$ **and** *proj2-abs a* ∈ *hyp2* (**is** *?pa* ∈ *hyp2*)
and *proj2-abs b* ∈ *hyp2* (**is** *?pb* ∈ *hyp2*)
shows *exp-2dist* (*proj2-abs a*) (*proj2-abs b*)
= (*a* · (*M* ∗*v b*) + *sqrt* (*quarter-discrim a b*))
/ (*a* · (*M* ∗*v b*) − *sqrt* (*quarter-discrim a b*))
∨ *exp-2dist* (*proj2-abs a*) (*proj2-abs b*)
= (*a* · (*M* ∗*v b*) − *sqrt* (*quarter-discrim a b*))
/ (*a* · (*M* ∗*v b*) + *sqrt* (*quarter-discrim a b*))
(**is** *?e2d* = (*?aMb* + *?sqd*) / (*?aMb* − *?sqd*)
∨ *?e2d* = (*?aMb* − *?sqd*) / (*?aMb* + *?sqd*))
⟨*proof*⟩

lemma *cosh-dist-formula*:
assumes $a \neq 0$ **and** $b \neq 0$ **and** *proj2-abs a* ∈ *hyp2* (**is** *?pa* ∈ *hyp2*)
and *proj2-abs b* ∈ *hyp2* (**is** *?pb* ∈ *hyp2*)
shows *cosh-dist* (*proj2-abs a*) (*proj2-abs b*)
= |*a* · (*M* ∗*v b*)| / *sqrt* (*a* · (*M* ∗*v a*) ∗ (*b* · (*M* ∗*v b*)))
(**is** *cosh-dist ?pa ?pb* = |*?aMb*| / *sqrt* (*?aMa* ∗ *?bMb*))
⟨*proof*⟩

lemma *cosh-dist-perp-special-case*:
assumes $|x| < 1$ **and** $|y| < 1$
shows *cosh-dist* (*proj2-abs* (*vector* [*x*,*0*,*1*])) (*proj2-abs* (*vector* [*0*,*y*,*1*]))
= (*cosh-dist K2-centre* (*proj2-abs* (*vector* [*x*,*0*,*1*])))
∗ (*cosh-dist K2-centre* (*proj2-abs* (*vector* [*0*,*y*,*1*])))
(**is** *cosh-dist ?pa ?pb* = (*cosh-dist ?po ?pa*) ∗ (*cosh-dist ?po ?pb*))
⟨*proof*⟩

lemma *K2-isometry-cross-ratio-endpoints-in-S*:
assumes *a* ∈ *hyp2* **and** *b* ∈ *hyp2* **and** *is-K2-isometry J* **and** *a* ≠ *b*
shows *cross-ratio* (*apply-cltn2* (*endpoint-in-S a b*) *J*)
(*apply-cltn2* (*endpoint-in-S b a*) *J*) (*apply-cltn2 a J*) (*apply-cltn2 b J*)
= *cross-ratio* (*endpoint-in-S a b*) (*endpoint-in-S b a*) *a b*
(**is** *cross-ratio ?pJ ?qJ ?aJ ?bJ* = *cross-ratio ?p ?q a b*)
⟨*proof*⟩

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*⟩

lemma *K2-isometry-cosh-dist*:
assumes *a* ∈ *hyp2* **and** *b* ∈ *hyp2* **and** *is-K2-isometry J*
shows *cosh-dist* (*apply-cltn2 a J*) (*apply-cltn2 b J*) = *cosh-dist a b*
⟨*proof*⟩

lemma *cosh-dist-perp*:
  assumes *M-perp l m* **and** $a \in hyp2$ **and** $b \in hyp2$ **and** $c \in hyp2$
  and *proj2-incident a l* **and** *proj2-incident b l*
  and *proj2-incident b m* **and** *proj2-incident c m*
  shows *cosh-dist a c* = *cosh-dist b a* ∗ *cosh-dist b c*
⟨*proof*⟩

lemma *are-endpoints-in-S-ordered-cross-ratio*:
  assumes *are-endpoints-in-S p q a b*
  and $B_{\mathbb{R}}$ (*cart2-pt a*) (*cart2-pt b*) (*cart2-pt p*) (**is** $B_{\mathbb{R}}$ *?ca ?cb ?cp*)
  shows *cross-ratio p q a b* ≥ *1*
⟨*proof*⟩

lemma *cross-ratio-S-S-hyp2-hyp2-positive*:
  assumes *are-endpoints-in-S p q a b*
  shows *cross-ratio p q a b* > *0*
⟨*proof*⟩

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*⟩

lemma *exp-2dist-positive*:
  assumes $a \in hyp2$ **and** $b \in hyp2$
  shows *exp-2dist a b* > *0*
⟨*proof*⟩

lemma *cosh-dist-at-least-1*:
  assumes $a \in hyp2$ **and** $b \in hyp2$
  shows *cosh-dist a b* ≥ *1*
⟨*proof*⟩

lemma *cosh-dist-positive*:
  assumes $a \in hyp2$ **and** $b \in hyp2$
  shows *cosh-dist a b* > *0*
⟨*proof*⟩

lemma *cosh-dist-perp-divide*:
  assumes *M-perp l m* **and** $a \in hyp2$ **and** $b \in hyp2$ **and** $c \in hyp2$
  and *proj2-incident a l* **and** *proj2-incident b l* **and** *proj2-incident b m*
  and *proj2-incident c m*
  shows *cosh-dist b c* = *cosh-dist a c* / *cosh-dist b a*
⟨*proof*⟩

lemma *real-hyp2-C-cross-ratio-endpoints-in-S*:
  assumes $a \neq b$ **and** $a\ b \equiv_K c\ d$
  shows *cross-ratio* (*endpoint-in-S* (*Rep-hyp2 a*) (*Rep-hyp2 b*))

($endpoint\text{-}in\text{-}S$ ($Rep\text{-}hyp2$ b) ($Rep\text{-}hyp2$ a)) ($Rep\text{-}hyp2$ a) ($Rep\text{-}hyp2$ b)
= $cross\text{-}ratio$ ($endpoint\text{-}in\text{-}S$ ($Rep\text{-}hyp2$ c) ($Rep\text{-}hyp2$ d))
($endpoint\text{-}in\text{-}S$ ($Rep\text{-}hyp2$ d) ($Rep\text{-}hyp2$ c)) ($Rep\text{-}hyp2$ c) ($Rep\text{-}hyp2$ d)
(**is** $cross\text{-}ratio\ ?p\ ?q\ ?a'\ ?b' = cross\text{-}ratio\ ?r\ ?s\ ?c'\ ?d'$)
⟨*proof*⟩

lemma *real-hyp2-C-exp-2dist*:
assumes $a\ b \equiv_K c\ d$
shows $exp\text{-}2dist$ ($Rep\text{-}hyp2$ a) ($Rep\text{-}hyp2$ b)
= $exp\text{-}2dist$ ($Rep\text{-}hyp2$ c) ($Rep\text{-}hyp2$ d)
(**is** $exp\text{-}2dist\ ?a'\ ?b' = exp\text{-}2dist\ ?c'\ ?d'$)
⟨*proof*⟩

lemma *real-hyp2-C-cosh-dist*:
assumes $a\ b \equiv_K c\ d$
shows $cosh\text{-}dist$ ($Rep\text{-}hyp2$ a) ($Rep\text{-}hyp2$ b)
= $cosh\text{-}dist$ ($Rep\text{-}hyp2$ c) ($Rep\text{-}hyp2$ d)
⟨*proof*⟩

lemma *cross-ratio-in-terms-of-cosh-dist*:
assumes $are\text{-}endpoints\text{-}in\text{-}S\ p\ q\ a\ b$
and $B_{\mathbb{R}}$ ($cart2\text{-}pt$ a) ($cart2\text{-}pt$ b) ($cart2\text{-}pt$ p)
shows $cross\text{-}ratio\ p\ q\ a\ b$
$= 2 * (cosh\text{-}dist\ a\ b)^2 + 2 * cosh\text{-}dist\ a\ b * sqrt\ ((cosh\text{-}dist\ a\ b)^2 - 1) - 1$
(**is** $?pqab = 2 * ?ab^2 + 2 * ?ab * sqrt\ (?ab^2 - 1) - 1$)
⟨*proof*⟩

lemma *are-endpoints-in-S-cross-ratio-correct*:
assumes $are\text{-}endpoints\text{-}in\text{-}S\ p\ q\ a\ b$
shows $cross\text{-}ratio\text{-}correct\ p\ q\ a\ b$
⟨*proof*⟩

lemma *endpoints-in-S-cross-ratio-correct*:
assumes $a \neq b$ **and** $a \in hyp2$ **and** $b \in hyp2$
shows $cross\text{-}ratio\text{-}correct$ ($endpoint\text{-}in\text{-}S$ a b) ($endpoint\text{-}in\text{-}S$ b a) a b
⟨*proof*⟩

lemma *endpoints-in-S-perp-foot-cross-ratio-correct*:
assumes $a \in hyp2$ **and** $b \in hyp2$ **and** $c \in hyp2$ **and** $a \neq b$
and $proj2\text{-}incident\ a\ l$ **and** $proj2\text{-}incident\ b\ l$
shows $cross\text{-}ratio\text{-}correct$
($endpoint\text{-}in\text{-}S$ a b) ($endpoint\text{-}in\text{-}S$ b a) a ($perp\text{-}foot$ c l)
(**is** $cross\text{-}ratio\text{-}correct\ ?p\ ?q\ a\ ?d$)
⟨*proof*⟩

lemma *cosh-dist-unique*:
assumes $a \in hyp2$ **and** $b \in hyp2$ **and** $c \in hyp2$ **and** $p \in S$
and $B_{\mathbb{R}}$ ($cart2\text{-}pt$ a) ($cart2\text{-}pt$ b) ($cart2\text{-}pt$ p) (**is** $B_{\mathbb{R}}\ ?ca\ ?cb\ ?cp$)
and $B_{\mathbb{R}}$ ($cart2\text{-}pt$ a) ($cart2\text{-}pt$ c) ($cart2\text{-}pt$ p) (**is** $B_{\mathbb{R}}\ ?ca\ ?cc\ ?cp$)

and *cosh-dist a b* = *cosh-dist a c* (**is** *?ab* = *?ac*)
shows $b = c$
⟨*proof*⟩

lemma *cosh-dist-swap*:
assumes $a \in$ *hyp2* **and** $b \in$ *hyp2*
shows *cosh-dist a b* = *cosh-dist b a*
⟨*proof*⟩

lemma *exp-2dist-1-equal*:
assumes $a \in$ *hyp2* **and** $b \in$ *hyp2* **and** *exp-2dist a b* = *1*
shows $a = b$
⟨*proof*⟩

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** $c \in$ *hyp2* **and** *are-endpoints-in-S p q a b*
and *proj2-incident p l* **and** *proj2-incident q l* **and** *M-perp l m*
and *proj2-incident d l* **and** *proj2-incident d m* **and** *proj2-incident c m*
shows *cross-ratio p q d a*
= (*cosh-dist b c* ∗ *sqrt* (*cross-ratio p q a b*) − *cosh-dist a c*)
/ (*cosh-dist a c* ∗ *cross-ratio p q a b*
− *cosh-dist b c* ∗ *sqrt* (*cross-ratio p q a b*))
(**is** *?pqda* = (*?bc* ∗ *sqrt ?pqab* − *?ac*) / (*?ac* ∗ *?pqab* − *?bc* ∗ *sqrt ?pqab*))
⟨*proof*⟩

lemma *perp-foot-cross-ratio-formula*:
assumes $a \in$ *hyp2* **and** $b \in$ *hyp2* **and** $c \in$ *hyp2* **and** $a \neq b$
shows *cross-ratio* (*endpoint-in-S a b*) (*endpoint-in-S b a*)
(*perp-foot c* (*proj2-line-through a b*)) *a*
= (*cosh-dist b c* ∗ *sqrt* (*exp-2dist a b*) − *cosh-dist a c*)
/ (*cosh-dist a c* ∗ *exp-2dist a b* − *cosh-dist b c* ∗ *sqrt* (*exp-2dist a b*))
(**is** *cross-ratio ?p ?q ?d a*
= (*?bc* ∗ *sqrt ?pqab* − *?ac*) / (*?ac* ∗ *?pqab* − *?bc* ∗ *sqrt ?pqab*))
⟨*proof*⟩

8.12 The Klein–Beltrami model satisfies axiom 5

lemma *statement69*:
assumes $a\ b \equiv_K a'\ b'$ **and** $b\ c \equiv_K b'\ c'$ **and** $a\ c \equiv_K a'\ c'$
shows ∃ *J*. *is-K2-isometry J*
∧ *hyp2-cltn2 a J* = a' ∧ *hyp2-cltn2 b J* = b' ∧ *hyp2-cltn2 c J* = c'
⟨*proof*⟩

theorem *hyp2-axiom5*:
$\forall\ a\ b\ c\ d\ a'\ b'\ c'\ d'.$
$a \neq b \land B_K\ a\ b\ c \land B_K\ a'\ b'\ c' \land a\ b \equiv_K a'\ b' \land b\ c \equiv_K b'\ c'$
$\land\ a\ d \equiv_K a'\ d' \land b\ d \equiv_K b'\ d'$
$\longrightarrow c\ d \equiv_K c'\ d'$

⟨*proof*⟩

interpretation *hyp2*: *tarski-first5 real-hyp2-C real-hyp2-B*
⟨*proof*⟩

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

theorem *hyp2-axiom6*: $\forall\ a\ b.\ B_K\ a\ b\ a \longrightarrow a = b$
⟨*proof*⟩

lemma *between-inverse*:
assumes $B_{\mathbb{R}}$ (*hyp2-rep p*) *v* (*hyp2-rep q*)
shows *hyp2-rep* (*hyp2-abs v*) = *v*
⟨*proof*⟩

lemma *between-switch*:
assumes $B_{\mathbb{R}}$ (*hyp2-rep p*) *v* (*hyp2-rep q*)
shows B_K *p* (*hyp2-abs v*) *q*
⟨*proof*⟩

theorem *hyp2-axiom7*:
$\forall\ a\ b\ c\ p\ q.\ B_K\ a\ p\ c \wedge B_K\ b\ q\ c \longrightarrow (\exists\ x.\ B_K\ p\ x\ b \wedge B_K\ q\ x\ a)$
⟨*proof*⟩

theorem *hyp2-axiom11*:
$\forall\ X\ Y.\ (\exists\ a.\ \forall\ x\ y.\ x \in X \wedge y \in Y \longrightarrow B_K\ a\ x\ y)$
$\longrightarrow (\exists\ b.\ \forall\ x\ y.\ x \in X \wedge y \in Y \longrightarrow B_K\ x\ b\ y)$
⟨*proof*⟩

interpretation *tarski-absolute-space real-hyp2-C real-hyp2-B*
⟨*proof*⟩

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*⟩

theorem *hyp2-axiom8*: $\exists\ a\ b\ c.\ \neg\ B_K\ a\ b\ c \wedge \neg\ B_K\ b\ c\ a \wedge \neg\ B_K\ c\ a\ b$
⟨*proof*⟩

theorem *hyp2-axiom9*:
$\forall\ p\ q\ a\ b\ c.\ p \neq q \wedge a\ p \equiv_K\ a\ q \wedge b\ p \equiv_K\ b\ q \wedge c\ p \equiv_K\ c\ q$
$\longrightarrow B_K\ a\ b\ c \vee B_K\ b\ c\ a \vee B_K\ c\ a\ b$
⟨*proof*⟩

interpretation *hyp2*: *tarski-absolute real-hyp2-C real-hyp2-B*
⟨*proof*⟩

8.15 The Klein–Beltrami model violates the Euclidean axiom

theorem *hyp2-axiom10-false*:
shows $\neg\ (\forall\ a\ b\ c\ d\ t.\ B_K\ a\ d\ t \wedge B_K\ b\ d\ c \wedge a \neq d$
$\longrightarrow (\exists\ x\ y.\ B_K\ a\ b\ x \wedge B_K\ a\ c\ y \wedge B_K\ x\ t\ y))$
⟨*proof*⟩

theorem *hyp2-not-tarski*: ¬ (*tarski real-hyp2-C real-hyp2-B*)
⟨*proof*⟩

Therefore axiom 10 is independent.

end

References

[1] K. Borsuk and W. Szmielew. *Foundations of Geometry: Euclidean and Bolyai-Lobachevskian Geometry; Projective Geometry.* North-Holland Publishing Company, 1960. Translated from Polish by Erwin Marquit.

[2] T. J. M. Makarios. A mechanical verification of the independence of Tarski's Euclidean axiom. Master's thesis, Victoria University of Wellington, New Zealand, 2012. http://researcharchive.vuw.ac.nz/handle/10063/2315.

[3] W. Schwabhäuser, W. Szmielew, and A. Tarski. *Metamathematische Methoden in der Geometrie.* Springer-Verlag, 1983.