Formalization of Gyrovector Spaces as Models of Hyperbolic Geometry and Special Relativity

Jelena Marković and Filip Marić

April 9, 2025

Abstract

In this paper, we present an Isabelle/HOL formalization of noncommutative and nonassociative algebraic structures known as *gyrogroups* and *gyrovector spaces*. These concepts were introduced by Abraham A. Ungar [1] and have deep connections to hyperbolic geometry and special relativity. Gyrovector spaces can be used to define models of hyperbolic geometry. Unlike other models, gyrovector spaces offer the advantage that all definitions exhibit remarkable syntactical similarities to standard Euclidean and Cartesian geometry (e.g., points on the line between a and b satisfy the parametric equation $a \oplus t \otimes (\ominus a \oplus b)$, for $t \in \mathbb{R}$, while the hyperbolic Pythagorean theorem is expressed as $a^2 \oplus b^2 = c^2$, where $\otimes$, $\oplus$, and $\ominus$ represent gyro operations).

We begin by formally defining gyrogroups and gyrovector spaces and proving their numerous properties. Next, we formalize Möbius and Einstein models of these abstract structures (formulated in the two-dimensional, complex plane), and then demonstrate that these are equivalent to the Poincaré and Klein-Beltrami models, satisfying Tarski's geometry axioms for hyperbolic geometry.

Contents

theory *GyroGroup*
  imports *Main*
begin

class *gyrogroupoid* =
  fixes *gyrozero* :: $'a$ (0_g)
  fixes *gyroplus* :: $'a \Rightarrow 'a \Rightarrow 'a$ (**infixl** $\oplus$ *100*)
begin
definition *gyroaut* :: $('a \Rightarrow 'a) \Rightarrow$ *bool* **where**
  gyroaut f $\longleftrightarrow$
    $(\forall\ a\ b.\ f\ (a \oplus b) = f\ a \oplus f\ b) \wedge$
    bij f
end

class *gyrogroup* = *gyrogroupoid* +
fixes *gyroinv* :: $'a \Rightarrow 'a$ ($\ominus$)
fixes *gyr* :: $'a \Rightarrow 'a \Rightarrow 'a \Rightarrow 'a$
assumes *gyro-left-id* [*simp*]: $\bigwedge a.\ 0_g \oplus a = a$
assumes *gyro-left-inv* [*simp*]: $\bigwedge a.\ \ominus a \oplus a = 0_g$
assumes *gyro-left-assoc*: $\bigwedge a\ b\ z.\ a \oplus (b \oplus z) = (a \oplus b) \oplus (gyr\ a\ b\ z)$
assumes *gyr-left-loop*: $\bigwedge a\ b.\ gyr\ a\ b = gyr\ (a \oplus b)\ b$
assumes *gyr-gyroaut*: $\bigwedge a\ b.\ gyroaut\ (gyr\ a\ b)$
begin

definition *gyrominus* :: $'a \Rightarrow 'a \Rightarrow 'a$ (**infixl** $\ominus_b$ *100*) **where**
$a \ominus_b b = a \oplus (\ominus\ b)$

end

context *gyrogroup*
begin

lemma *gyr-distrib* [*simp*]:
$gyr\ a\ b\ (x \oplus y) = gyr\ a\ b\ x \oplus gyr\ a\ b\ y$
⟨*proof*⟩

lemma *gyr-inj*:
assumes $gyr\ a\ b\ x = gyr\ a\ b\ y$
shows $x = y$
⟨*proof*⟩

Def 2.7, (2.2)

definition *cogyroplus* (**infixr** $\oplus_c$ *100*) **where**
$a \oplus_c b = a \oplus (gyr\ a\ (\ominus b)\ b)$

definition *cogyrominus* :: $'a \Rightarrow 'a \Rightarrow 'a$ (**infixl** $\ominus_{cb}$ *100*) **where**
$a \ominus_{cb} b = a \oplus_c (\ominus\ b)$

definition *cogyroinv* ($\ominus_c$) **where**
$\ominus_c\ a = 0_g \ominus_{cb} a$

Thm 2.8, (1)

lemma *gyro-left-cancel*:
assumes $a \oplus b = a \oplus c$
shows $b = c$
⟨*proof*⟩

Thm 2.8, (2)

definition *gyro-is-left-id* **where**
$gyro\text{-}is\text{-}left\text{-}id\ z \longleftrightarrow (\forall\ x.\ z \oplus x = x)$

lemma *gyro-is-left-id-0* [*simp*]:

shows *gyro-is-left-id* 0_g
⟨*proof*⟩

lemma *gyr-id-1'*:
assumes *gyro-is-left-id z*
shows *gyr z a* = *id*
⟨*proof*⟩

lemma *gyr-id-1* [*simp*]:
shows *gyr* 0_g *a* = *id*
⟨*proof*⟩

Thm 2.8, (3)

definition *gyro-is-left-inv* **where**
gyro-is-left-inv x a ⟷ $x \oplus a = 0_g$

definition *gyro-is-right-inv* **where**
gyro-is-right-inv x a ⟷ $a \oplus x = 0_g$

lemma *gyro-is-left-inv* [*simp*]:
shows *gyro-is-left-inv* $(\ominus a)$ *a*
⟨*proof*⟩

lemma *gyr-inv-1'*:
assumes *gyro-is-left-inv x a*
shows *gyr x a* = *id*
⟨*proof*⟩

lemma *gyr-inv-1* [*simp*]:
shows *gyr* $(\ominus a)$ *a* = *id*
⟨*proof*⟩

Thm 2.8, (4)

lemma *gyr-id* [*simp*]:
shows *gyr a a* = *id*
⟨*proof*⟩

Thm 2.8, (5)

lemma *gyro-right-id* [*simp*]:
shows $a \oplus 0_g = a$
⟨*proof*⟩

lemma *gyro-inv-id* [*simp*]: $\ominus\ 0_g = 0_g$
⟨*proof*⟩

Thm 2.8, (6)

lemma *gyro-left-id-unique*:
assumes *gyro-is-left-id z*
shows $z = 0_g$

⟨*proof*⟩

Thm 2.8, (7)

lemma *gyro-left-inv-right-inv*:
assumes *gyro-is-left-inv x a*
shows *gyro-is-right-inv x a*
⟨*proof*⟩

lemma *gyro-rigth-inv* [*simp*]:
shows $a \oplus (\ominus a) = 0_g$
⟨*proof*⟩

Thm 2.8, (8)

lemma
assumes *gyro-is-left-inv x a*
shows $x = \ominus a$
⟨*proof*⟩

Thm 2.8, (9)

lemma *gyro-left-cancel'*:
shows $\ominus\ a \oplus (a \oplus b) = b$
⟨*proof*⟩

Thm 2.8, (10)

lemma *gyr-def*:
shows $gyr\ a\ b\ x = \ominus\ (a \oplus b) \oplus (a \oplus (b \oplus x))$
⟨*proof*⟩

Thm 2.8, (11)

lemma *gyr-id-3*:
shows $gyr\ a\ b\ 0_g = 0_g$
⟨*proof*⟩

Thm 2.8, (12)

lemma *gyr-inv-3*:
shows $gyr\ a\ b\ (\ominus x) = \ominus\ (gyr\ a\ b\ x)$
⟨*proof*⟩

Thm 2.8, (13)

lemma *gyr-id-2* [*simp*]:
shows $gyr\ a\ 0_g = id$
⟨*proof*⟩

lemma *gyr-distrib-gyrominus*:
shows $gyr\ a\ b\ (c \ominus_b d) = gyr\ a\ b\ c \ominus_b gyr\ a\ b\ d$
⟨*proof*⟩

lemma *gyro-inv-idem* [*simp*]:
shows $\ominus\ (\ominus\ a) = a$

⟨*proof*⟩

lemma *gyr-inv-2* [*simp*]:
shows *gyr a* (⊖ *a*) = *id*
⟨*proof*⟩

(2.3.a)

lemma *cogyro-left-id*:
shows $0_g \oplus_c a = a$
⟨*proof*⟩

(2.3.b)

lemma *cogyro-rigth-id*:
shows $a \oplus_c 0_g = a$
⟨*proof*⟩

(2.4)

lemma *cogyrominus*:
shows $a \ominus_{cb} b = a \ominus_b$ *gyr a b b*
⟨*proof*⟩

(2.7)

lemma *cogyro-right-inv*:
shows $a \oplus_c (\ominus_c a) = 0_g$
⟨*proof*⟩

(2.6)

lemma *cogyro-left-inv*:
shows $(\ominus_c a) \oplus_c a = 0_g$
⟨*proof*⟩

(2.8)

lemma *cogyro-gyro-inv*:
shows $\ominus_c a = \ominus a$
⟨*proof*⟩

Thm 2.9, (2.9)

lemma *gyr-nested-1*:
shows *gyr a* (*b* ⊕ *c*) ∘ *gyr b c* = *gyr* (*a* ⊕ *b*) (*gyr a b c*) ∘ *gyr a b* (**is** *?lhs* = *?rhs*)
⟨*proof*⟩

Thm 2.9, (2.15)

lemma *gyr-nested-1'*:
shows *gyr* (*a* ⊕ *b*) (⊖ (*gyr a b b*)) ∘ *gyr a b* = *id*
⟨*proof*⟩

Thm 2.9, (2.10)

lemma *gyr-nested-2*:

shows $gyr\ a\ (\ominus\ (gyr\ a\ b\ b)) \circ gyr\ a\ b = id$
⟨*proof*⟩

Thm 2.9, (2.11)

lemma *gyr-auto-id1*:
shows $gyr\ (\ominus\ a)\ (a \oplus b) \circ gyr\ a\ b = id$
⟨*proof*⟩

Thm 2.9, (2.12)

lemma *gyr-auto-id2*:
shows $gyr\ b\ (a \oplus b) \circ gyr\ a\ b = id$
⟨*proof*⟩

Thm 2.10, (2.18)

lemma *gyro-plus-def-co*:
shows $a \oplus b = a \oplus_c gyr\ a\ b\ b$
⟨*proof*⟩

Thm 2.11, (2.21)

lemma *gyro-polygonal-addition-lemma*:
shows $(\ominus\ a \oplus b) \oplus gyr\ (\ominus a)\ b\ (\ominus\ b \oplus c) = \ominus\ a \oplus c$
⟨*proof*⟩

Thm 2.12, (2.23)

lemma *gyro-translation-1*:
shows $\ominus\ (\ominus a \oplus b) \oplus (\ominus a \oplus c) = gyr\ (\ominus\ a)\ b\ (\ominus b \oplus c)$
⟨*proof*⟩

Thm 3.13, (3.33a)

lemma *gyro-translation-2a*:
shows $\ominus\ (a \oplus b) \oplus (a \oplus c) = gyr\ a\ b\ (\ominus b \oplus c)$
⟨*proof*⟩

definition *gyro-polygonal-add* $(\oplus_p)$ **where**
$\oplus_p\ a\ b\ c = (\ominus\ a \oplus b) \oplus gyr\ (\ominus\ a)\ b\ (\ominus\ b \oplus c)$

Thm 2.15, (2.34, 2.35)

lemma *gyro-equation-right*:
shows $a \oplus x = b \longleftrightarrow x = \ominus a \oplus b$
⟨*proof*⟩

Thm 2.15, (2.36, 2.37)

lemma *gyro-equation-left*:
shows $x \oplus a = b \longleftrightarrow x = b \ominus_{cb} a$
⟨*proof*⟩

lemma *oplus-ominus-cancel* [*simp*]:
shows $y = x \oplus (\ominus\ x \oplus y)$
⟨*proof*⟩

(2.39)

lemma *cogyro-right-cancel′*:
shows $(b \ominus_{cb} a) \oplus a = b$
⟨*proof*⟩

(2.40)

lemma *gyro-right-cancel′-dual*:
shows $(b \ominus_b a) \oplus_c a = b$
⟨*proof*⟩

Thm 2.19 (2.48)

lemma *gyroaut-gyr-commute-lemma*:
assumes *gyroaut A*
shows $A \circ gyr\ a\ b = gyr\ (A\ a)\ (A\ b) \circ A$ (**is** *?lhs = ?rhs*)
⟨*proof*⟩

Thm 2.20

lemma *gyroaut-gyr-commute*:
assumes *gyroaut A*
shows $gyr\ a\ b = gyr\ (A\ a)\ (A\ b) \longleftrightarrow A \circ gyr\ a\ b = gyr\ a\ b \circ A$
⟨*proof*⟩

2.50

lemma *gyr-commute-misc-1*:
shows $gyr\ (gyr\ a\ b\ a)\ (gyr\ a\ b\ b) = gyr\ a\ b$
⟨*proof*⟩

Thm 2.21 (2.52)

definition
$cogyroaut\ f \longleftrightarrow ((\forall\ a\ b.\ f\ (a \oplus_c b) = f\ a \oplus_c f\ b) \wedge bij\ f)$

lemma *gyro-coaut-iff-gyro-aut*:
shows $gyroaut\ f \longleftrightarrow cogyroaut\ f$
⟨*proof*⟩

Thm 2.25, (2.76)

lemma *gyroplus-inv*:
shows $\ominus\ (a \oplus b) = gyr\ a\ b\ (\ominus\ b \ominus_b a)$
⟨*proof*⟩

Thm 2.25, (2.77)

lemma *inv-gyr*:
shows $inv\ (gyr\ a\ b) = gyr\ (\ominus b)\ (\ominus a)$
⟨*proof*⟩

Thm 2.26, (2.86)

lemma *gyr-aut-inv-1*:
shows $inv\ (gyr\ a\ b) = gyr\ a\ (\ominus\ (gyr\ a\ b\ b))$

⟨*proof*⟩

Thm 2.26, (2.87)

lemma *gyr-aut-inv-2*:
shows $inv\ (gyr\ a\ b) = gyr\ (\ominus\ a)\ (a \oplus b)$
⟨*proof*⟩

Thm 2.26, (2.88)

lemma *gyr-aut-inv-3*:
shows $inv\ (gyr\ a\ b) = gyr\ b\ (a \oplus b)$
⟨*proof*⟩

Thm 2.26, (2.89)

lemma *gyr-1*:
shows $gyr\ a\ b = gyr\ b\ (\ominus b \ominus_b a)$
⟨*proof*⟩

Thm 2.26, (2.90)

lemma *gyr-2*:
shows $gyr\ a\ b = gyr\ (\ominus a)\ (\ominus b \ominus_b a)$
⟨*proof*⟩

Thm 2.26, (2.91)

lemma *gyr-3*:
shows $gyr\ a\ b = gyr\ (\ominus\ (a \oplus b))\ a$
⟨*proof*⟩

Thm 2.27, (2.92)

lemma *gyr-even*:
shows $gyr\ (\ominus\ a)\ (\ominus\ b) = gyr\ a\ b$
⟨*proof*⟩

Thm 2.27, (2.93)

lemma *inv-gyr-sym*:
shows $inv\ (gyr\ a\ b) = gyr\ b\ a$
⟨*proof*⟩

Thm 2.27, (2.94a)

lemma *gyr-nested-3*:
shows $gyr\ b\ (\ominus\ (gyr\ b\ a\ a)) = gyr\ a\ b$
⟨*proof*⟩

Thm 2.27, (2.94b)

lemma *gyr-nested-4*:
shows $gyr\ b\ (gyr\ b\ (\ominus\ a)\ a) = gyr\ a\ (\ominus\ b)$
⟨*proof*⟩

Thm 2.27, (2.94c)

lemma *gyr-nested-5*:

shows *gyr* $(\ominus\ (gyr\ a\ b\ b))\ a = gyr\ a\ b$
⟨*proof*⟩

Thm 2.27, (2.94d)

lemma *gyr-nested-6*:
shows *gyr* $(gyr\ a\ (\ominus\ b)\ b)\ a = gyr\ a\ (\ominus\ b)$
⟨*proof*⟩

Thm 2.28, (i)

lemma *gyro-right-assoc*:
shows $(a \oplus b) \oplus c = a \oplus (b \oplus gyr\ b\ a\ c)$
⟨*proof*⟩

Thm 2.28, (ii)

lemma *gyr-right-loop*:
shows *gyr* $a\ b = gyr\ a\ (b \oplus a)$
⟨*proof*⟩

Thm 2.29, (a)

lemma *gyr-left-coloop*:
shows *gyr* $a\ b = gyr\ (a \ominus_{cb} b)\ b$
⟨*proof*⟩

Thm 2.29, (b)

lemma *gyr-rigth-coloop*:
shows *gyr* $a\ b = gyr\ a\ (b \ominus_{cb} a)$
⟨*proof*⟩

Thm 2.30, (2.101a)

lemma *gyr-misc-1*:
shows *gyr* $(a \oplus b)\ (\ominus\ a) = gyr\ a\ b$
⟨*proof*⟩

Thm 2.30, (2.101b)

lemma *gyr-misc-2*:
shows *gyr* $(\ominus\ a)\ (a \oplus b) = gyr\ b\ a$
⟨*proof*⟩

Thm 2.31, (2.103)

lemma *coautomorphic-inverse*:
shows $\ominus\ (a \oplus_c b) = (\ominus\ b) \oplus_c (\ominus\ a)$
⟨*proof*⟩

Thm 2.32, (2.105a)

lemma *gyr-misc-3*:
shows *gyr* $a\ b\ b = \ominus\ (\ominus\ (a \oplus b) \oplus a)$
⟨*proof*⟩

Thm 2.32, (2.105b)

lemma *gyr-misc-4*:
shows $gyr\ a\ (\ominus\ b)\ b = \ominus\ (a \ominus_b\ b) \oplus a$
⟨*proof*⟩

Thm 2.35, (2.124)

lemma *mixed-gyroassoc-law*: $(a \oplus_c b) \oplus c = a \oplus gyr\ a\ (\ominus\ b)\ (b \oplus c)$
⟨*proof*⟩

Thm 3.2

lemma *gyrocommute-iff-gyroatomorphic-inverse*:
shows $(\forall\ a\ b.\ \ominus\ (a \oplus b) = \ominus\ a \ominus_b b) \longleftrightarrow (\forall\ a\ b.\ a\oplus b = gyr\ a\ b\ (b \oplus a))$
⟨*proof*⟩

Thm 3.4

lemma *cogyro-commute-iff-gyrocommute*:
$(\forall\ a\ b.\ a \oplus_c b = b \oplus_c a) \longleftrightarrow (\forall\ a\ b.\ a\oplus b = gyr\ a\ b\ (b \oplus a))$ (**is** *?lhs* ⟷ *?rhs*)
⟨*proof*⟩

end

class *gyrocommutative-gyrogroup* = *gyrogroup* +
assumes *gyro-commute*: $a \oplus b = gyr\ a\ b\ (b \oplus a)$
begin
lemma *gyroautomorphic-inverse*:
shows $\ominus\ (a \oplus b) = \ominus\ a \ominus_b b$
⟨*proof*⟩

lemma *cogyro-commute*:
shows $a \oplus_c b = b \oplus_c a$
⟨*proof*⟩

Thm 3.5 (3.15)

lemma *gyr-commute-misc-2*:
shows $gyr\ a\ b \circ gyr\ (b \oplus a)\ c = gyr\ a\ (b \oplus c) \circ gyr\ b\ c$
⟨*proof*⟩

Thm 3.6 (3.17, 3.18)

lemma *gyr-parallelogram*:
assumes $d = (b \oplus_c c) \ominus_b a$
shows $gyr\ a\ (\ominus\ b) \circ gyr\ b\ (\ominus\ c) \circ gyr\ c\ (\ominus\ d) = gyr\ a\ (\ominus\ d)$
⟨*proof*⟩

Thm 3.8 (3.23, 3.24)

lemma *gyr-parallelogram-iff*:
$d = (b \oplus_c c) \ominus_b a \longleftrightarrow \ominus c \oplus d = gyr\ c\ (\ominus b)\ (b \ominus_b a)$
⟨*proof*⟩

Thm 3.9 (3.26)

lemma *gyr-commute-misc-3*:

$gyr\ a\ b\ (b \oplus (a \oplus c)) = (a \oplus b) \oplus c$
⟨*proof*⟩

Thm 3.10 (3.28)

lemma *gyro-left-right-cancel*:
shows $(a \oplus b) \ominus_b a = gyr\ a\ b\ b$
⟨*proof*⟩

Thm 3.11 (3.29)

lemma *cogyro-plus-def*:
shows $a \oplus_c b = a \oplus ((\ominus a \oplus b) \oplus a)$
⟨*proof*⟩

Thm 3.12 (3.31)

lemma *cogyro-commute-misc1*:
shows $a \oplus_c (a \oplus b) = a \oplus (b \oplus a)$
⟨*proof*⟩

Thm 3.13 (3.33b)

lemma *gyro-translation-2b*:
shows $(a \oplus b) \ominus_b (a \oplus c) = gyr\ a\ b\ (b \ominus_b c)$
⟨*proof*⟩

Thm 3.14 (3.34)

(3.37)

lemma *gyr-commute-misc-4'*:
shows $gyr\ a\ (b \oplus c) = gyr\ a\ b \circ gyr\ (b \oplus a)\ c \circ gyr\ c\ b$
⟨*proof*⟩

(3.38)

lemma *gyr-commute-misc-4''*:
shows $gyr\ (\ominus b \oplus d)\ (b \oplus c) = gyr\ (\ominus\ b)\ d \circ gyr\ d\ c \circ gyr\ c\ b$
⟨*proof*⟩

Thm 3.14 (3.34)

lemma *gyro-commute-misc-4*:
shows $gyr\ (\ominus\ a \oplus b)\ (a \ominus_b c) = gyr\ a\ (\ominus\ b) \circ gyr\ b\ (\ominus\ c) \circ gyr\ c\ (\ominus\ a)$
⟨*proof*⟩

Thm 3.15 (3.40)

lemma *gyr-inv-2'*:
shows $gyr\ a\ (\ominus b) = gyr\ (\ominus\ a \oplus b)\ (a \oplus b) \circ gyr\ a\ b$
⟨*proof*⟩

Thm 3.17 (3.48)

lemma *gyr-master'*:
shows $gyr\ a\ x \circ gyr\ (\ominus\ (x \oplus a))\ (x \oplus b) \circ gyr\ x\ b = gyr\ (\ominus\ a)\ b$
⟨*proof*⟩

(3.51)

lemma *gyr-master*:
shows $gyr\ a\ x \circ gyr\ (x \oplus a)\ (\ominus\ (x \oplus b)) \circ gyr\ x\ b = gyr\ (\ominus\ a)\ b$
⟨*proof*⟩

(3.52a)

lemma *gyr-master-misc1′*:
shows $gyr\ (\ominus\ a)\ b = gyr\ (\ominus\ (a \oplus a))\ (a \oplus b) \circ gyr\ a\ b$
⟨*proof*⟩

(3.52b)

lemma *gyr-master-misc1′′*:
shows $gyr\ (\ominus\ a)\ b = gyr\ a\ b \circ gyr\ (b \oplus a)\ (\ominus\ (b \oplus b))$
⟨*proof*⟩

(3.53a)

lemma *gyr-master-misc2′*:
shows $gyr\ (\ominus a \oplus b)\ (a \oplus b) = gyr\ (\ominus\ a)\ b \circ gyr\ b\ a$
⟨*proof*⟩

(3.53b)

lemma *gyr-master-misc2′′*:
shows $gyr\ (\ominus a \oplus b)\ (a \oplus b) = gyr\ (\ominus\ a \oplus b)\ b \circ gyr\ b\ (a \oplus b)$
⟨*proof*⟩

Thm 3.18 (3.60)

lemma $gyr\ a\ x \circ gyr\ (\ominus\ (gyr\ x\ a\ (a \ominus_b\ b)))\ (x \oplus b) \circ gyr\ x\ b = gyr\ a\ (\ominus\ b)$
⟨*proof*⟩

definition *gyro-covariant* :: $nat \Rightarrow ('a\ list \Rightarrow 'a) \Rightarrow bool$ **where**
gyro-covariant $n\ T \longleftrightarrow (\forall\ \tau\ xs.\ length\ xs = n \wedge gyroaut\ \tau \longrightarrow (\tau\ (T\ xs)) = T\ (map\ \tau\ xs)) \wedge$
$(\forall\ x\ xs.\ length\ xs = n \longrightarrow x \oplus T\ xs = T\ (map\ (\lambda\ a.\ x \oplus a)\ xs))$

definition *gyro-covariant-3* :: $('a \Rightarrow 'a \Rightarrow 'a \Rightarrow 'a) \Rightarrow bool$ **where**
gyro-covariant-3 $T \longleftrightarrow (\forall\ \tau\ a\ b\ c.\ gyroaut\ \tau \longrightarrow (\tau\ (T\ a\ b\ c)) = T\ (\tau\ a)\ (\tau\ b)\ (\tau\ c)) \wedge$
$(\forall\ x\ a\ b\ c.\ x \oplus T\ a\ b\ c = T\ (x \oplus a)\ (x \oplus b)\ (x \oplus c))$

lemma *gyro-covariant-3*:
shows *gyro-covariant-3* $T \longleftrightarrow$ *gyro-covariant* $3\ (\lambda\ xs.\ T\ (xs\ !\ 0)\ (xs\ !\ 1)\ (xs\ !\ 2))$
⟨*proof*⟩

Thm 3.19 (3.62)

lemma *gyro-covariant-3-parallelogram*:
shows *gyro-covariant-3* $(\lambda\ a\ b\ c.\ (b \oplus_c\ c) \ominus_b\ a)$
⟨*proof*⟩

lemma *gyro-commute-misc6'*:
shows $x \oplus ((b \oplus_c c) \ominus_b a) = ((x \oplus b) \oplus_c (x \oplus c)) \ominus_b (x \oplus a)$
⟨*proof*⟩

(3.66)

lemma *gyro-commute-misc6*:
shows $(x \oplus b) \oplus_c (x \oplus c) = x \oplus ((b \oplus_c c) \oplus x)$
⟨*proof*⟩

(3.67)

lemma *gyro-commute-misc6''*:
shows $(x \oplus b) \oplus_c (x \ominus_b b) = x \oplus x$
⟨*proof*⟩

end

type-synonym *'a rooted-gyrovec* = *'a* × *'a*

context *gyrogroup*
begin

Def 5.2.

fun *head* :: *'a rooted-gyrovec* ⇒ *'a* **where**
head (*p*, *q*) = *q*
fun *tail* :: *'a rooted-gyrovec* ⇒ *'a* **where**
tail (*p*, *q*) = *p*
fun *val* :: *'a rooted-gyrovec* ⇒ *'a* **where**
val (*p*, *q*) = $\ominus$ *p* $\oplus$ *q*
definition *ort* :: *'a* ⇒ *'a rooted-gyrovec* **where**
ort p = (0_g, *p*)

fun *equiv-rooted-gyro-vec* (**infixl** ∼ *100*) **where**
$(p, q) \sim (p', q') \longleftrightarrow \ominus p \oplus q = \ominus p' \oplus q'$

lemma *equivp-equiv-rooted-gyro-vec* [*simp*]:
shows *equivp* (∼)
⟨*proof*⟩

end

Def 5.4.

quotient-type (**overloaded**) *'a gyrovec* = *'a* :: *gyrogroup* × *'a* / *equiv-rooted-gyro-vec*
⟨*proof*⟩

lift-definition *vec* :: *'a::gyrogroup* ⇒ *'a* ⇒ *'a gyrovec* **is** λ *p q*. (*p*, *q*) ⟨*proof*⟩

definition *ort* :: *'a::gyrogroup* ⇒ *'a gyrovec* **where**
ort A = *vec* 0_g *A*

context *gyrocommutative-gyrogroup*
begin

Thm 5.5. (5.4)

lemma *equiv-rooted-gyro-vec-ex-t*:
shows $(p, q) \sim (p', q') \longleftrightarrow (\exists\ t.\ p' = gyr\ p\ t\ (t \oplus p) \wedge q' = gyr\ p\ t\ (t \oplus q))$
(**is** *?lhs* $\longleftrightarrow$ *?rhs*)
⟨*proof*⟩

Thm 5.5. (5.5)

lemma *gyro-translate-commute*:
assumes $p' = gyr\ p\ t\ (t \oplus p) \wedge q' = gyr\ p\ t\ (t \oplus q)$
shows $t = \ominus p \oplus p'$
⟨*proof*⟩

Def 5.6.

fun *gyrovec-translation* :: $'a \Rightarrow 'a$ *rooted-gyrovec* $\Rightarrow 'a$ *rooted-gyrovec* **where**
gyrovec-translation $t\ (p, q) = (gyr\ p\ t\ (t \oplus p),\ gyr\ p\ t\ (t \oplus q))$
end

lift-definition *gyrovec-translation'* :: ('*a*::*gyrocommutative-gyrogroup*) *gyrovec* $\Rightarrow$ '*a rooted-gyrovec* $\Rightarrow$ '*a rooted-gyrovec* **is**
λ (*tp*, *tq*) (*p*, *q*). *gyrovec-translation* ($\ominus$ *tp* $\oplus$ *tq*) (*p*, *q*)
⟨*proof*⟩

(5.14)

lemma
shows *tail* (*gyrovec-translation* t (p, q)) $= p \oplus t$
⟨*proof*⟩

(5.15)

lemma *gyrovec-translation-id*:
shows *gyrovec-translation* 0_g $(p, q) = (p, q)$
⟨*proof*⟩

Thm 5.7.

lemma *equiv-rooted-gyrovec-t*:
shows $(p, q) \sim (p', q') \longleftrightarrow (p', q') =$ *gyrovec-translation* $(\ominus p \oplus p')$ (p, q)
⟨*proof*⟩

Thm 5.8.

lemma *gyrovec-translation-head*:
assumes $(p', x) =$ *gyrovec-translation* t (p, q)
shows $x = p' \oplus (\ominus p \oplus q)$
⟨*proof*⟩

(5.24)

context *gyrocommutative-gyrogroup*

begin

definition *gyrovec-translation-inv'* :: *'a* ⇒ *'a* ⇒ *'a* **where**
gyrovec-translation-inv' p t = ⊖ (*gyr p t t*)

lemma *gyrovec-translation-inv'*:
shows *gyrovec-translation* (*gyrovec-translation-inv' p t*) (*gyrovec-translation t* (*p*, *q*)) = (*p*, *q*)
⟨*proof*⟩

definition *gyrovec-translation-compose'* :: *'a* ⇒ *'a* ⇒ *'a* ⇒ *'a* **where**
gyrovec-translation-compose' p t1 t2 = *t1* ⊕ *gyr t1 p t2*

lemma *gyrovec-translation-compose'*:
gyrovec-translation t2 (*gyrovec-translation t1* (*p*, *q*)) =
gyrovec-translation (*gyrovec-translation-compose' p t1 t2*) (*p*, *q*)
⟨*proof*⟩

fun *equiv-translate* (**infixl** $\sim_t$ *100*) **where**
(*p1*, *q1*) $\sim_t$ (*p2*, *q2*) ⟷ (∃ *t*. *gyrovec-translation t* (*p1*, *q1*) = (*p2*, *q2*))

lemma *equivp-equiv-translate*:
equivp ($\sim_t$)
⟨*proof*⟩

(5.39)

definition *vec* :: *'a* ⇒ *'a* ⇒ *'a* **where**
vec a b = ⊖ *a* ⊕ *b*

(5.40)

lemma *vec* 0_g *b* = *b*
⟨*proof*⟩

(5.41)

lemma
assumes *vec a b* = *v*
shows *b* = *a* ⊕ *v*
⟨*proof*⟩

(5.42)

lemma
(*a* ⊕ *v*) ⊕ *u* = *a* ⊕ (*v* ⊕ *gyr v a u*)
⟨*proof*⟩

(5.43)

lemma
assumes *vec a b* = *v*
shows *a* = ⊖*v* $\oplus_c$ *b*
⟨*proof*⟩

lemma
  shows $(\ominus\ a \oplus b) \oplus gyr\ (\ominus a)\ b\ (\ominus\ b \oplus c) = \ominus a \oplus c$
  ⟨*proof*⟩

definition *torsion-elem*::′*a*⇒ *bool* **where**
  $torsion\text{-}elem\ g \longleftrightarrow g \oplus g = 0_g$

end

class *tf-tw-group* = *gyrocommutative-gyrogroup* +
  assumes *a1*:$\forall\ a.\ torsion\text{-}elem\ a \longrightarrow a = 0_g$
  assumes *a2*:$\forall\ a.\ \exists\ b.\ (b \oplus b = a)$
begin

T3.32

lemma *unique-half*:
  shows $(a \oplus a = c \wedge b \oplus b = c) \longrightarrow a = b$
⟨*proof*⟩

T3.33

lemma *unique-gyro-half*:
  assumes $gh \oplus gh = g$
    $gyr\text{-}h \oplus gyr\text{-}h = gyr\ a\ b\ g$
  shows $gyr\ a\ b\ gh = gyr\text{-}h$
  ⟨*proof*⟩

3.102

lemma *gh-minus*:
  assumes $gh \oplus gh = \ominus\ g$
    $gh2 \oplus gh2 = g$
  shows $\ominus\ gh2 = gh$
  ⟨*proof*⟩

T3.34

lemma *gyration-exclusion*:
  assumes $\exists\ g.\ g \neq 0_g$
  shows $\forall\ a\ b.\ \ gyr\ a\ b \neq\ \ominus \circ id$
⟨*proof*⟩

T3.35

lemma *gyration-exclusion-cons*:
  shows $gyr\ a\ b\ b = \ \ominus\ b \longrightarrow b = 0_g$
⟨*proof*⟩

T3.36

lemma *equation-t3-36*:

shows $x \ominus_b (y \ominus_b x) = y \longleftrightarrow x = y$
⟨*proof*⟩

end

locale *gyrogroup-isomorphism* =
fixes φ :: *'a::gyrocommutative-gyrogroup* $\Rightarrow$ *'b*
fixes *gyrozero'* :: *'b* (0_{g1})
fixes *gyroplus'* :: *'b* $\Rightarrow$ *'b* $\Rightarrow$ *'b* (**infixl** $\oplus_1$ *100*)
fixes *gyroinv'* :: *'b* $\Rightarrow$ *'b* ($\ominus_1$)
assumes *φzero* [*simp*]: $\varphi\ 0_g = 0_{g1}$
assumes *φplus* [*simp*]: $\varphi\ (a \oplus b) = (\varphi\ a) \oplus_1 (\varphi\ b)$
assumes *φminus* [*simp*]: $\varphi\ (\ominus\ a) = \ominus_1 (\varphi\ a)$
assumes *φbij* [*simp*]: *bij* φ
begin

definition *gyr'* **where**
$gyr'\ a\ b\ x = \ominus_1 (a \oplus_1 b) \oplus_1 (a \oplus_1 (b \oplus_1 x))$

lemma *φgyr* [*simp*]:
shows $\varphi\ (gyr\ a\ b\ z) = gyr'\ (\varphi\ a)\ (\varphi\ b)\ (\varphi\ z)$
⟨*proof*⟩

end

sublocale *gyrogroup-isomorphism* ⊆ *gyrogroupoid gyrozero' gyroplus'*
⟨*proof*⟩

sublocale *gyrogroup-isomorphism* ⊆ *gyrocommutative-gyrogroup gyrozero' gyroplus' gyroinv' gyr'*
⟨*proof*⟩

end
theory *More-Real-Vector*
imports *Main HOL−Analysis.Inner-Product HOL.Real-Vector-Spaces*
begin

lemma (**in** *real-vector*) *inner-eq-1*:
assumes *norm a = 1 norm b = 1 inner a b = 1*
shows $a = b$
⟨*proof*⟩

end
theory *GyroVectorSpace*
imports *GyroGroup HOL−Analysis.Inner-Product HOL.Real-Vector-Spaces More-Real-Vector*
begin

locale *gyrocarrier′* =
fixes *to-carrier* :: *′a*::*gyrocommutative-gyrogroup* ⇒ *′b*::{*real-inner*}
assumes *inj-to-carrier* [*simp*]: *inj to-carrier*
assumes *to-carrier-zero* [*simp*]: *to-carrier* 0_g = *0*
begin

definition *carrier* :: *′b set* **where**
carrier = *to-carrier* ' *UNIV*

lemma *bij-betw-to-carrier*:
shows *bij-betw to-carrier UNIV carrier*
⟨*proof*⟩

definition *of-carrier* :: *′b* ⇒ *′a* **where**
of-carrier = *inv to-carrier*

lemma *bij-betw-of-carrier*:
shows *bij-betw of-carrier carrier UNIV*
⟨*proof*⟩

lemma *inj-on-of-carrier* [*simp*]:
shows *inj-on of-carrier carrier*
⟨*proof*⟩

lemma *to-carrier* [*simp*]:
shows ⋀ *b*. *b* ∈ *carrier* ⟹ *to-carrier* (*of-carrier b*) = *b*
⟨*proof*⟩

lemma *of-carrier* [*simp*]:
shows ⋀ *a*. *of-carrier* (*to-carrier a*) = *a*
⟨*proof*⟩

lemma *of-carrier-zero* [*simp*]:
shows *of-carrier 0* = 0_g
⟨*proof*⟩

lemma *to-carrier-zero-iff*:
assumes *to-carrier a* = *0*
shows *a* = 0_g
⟨*proof*⟩

definition *gyronorm* :: *′a* ⇒ *real* (⟪-⟫ [*100*] *100*) **where**
⟪*a*⟫ = *norm* (*to-carrier a*)

definition *gyroinner* :: *′a* ⇒ *′a* ⇒ *real* (**infixl** · *100*) **where**
a · *b* = *inner* (*to-carrier a*) (*to-carrier b*)

lemma *norm-inner*: ⟪*a*⟫ = *sqrt* (*a* · *a*)
⟨*proof*⟩

lemma *norm-zero*:
  shows $\langle\!\langle 0_g \rangle\!\rangle = 0$
  ⟨*proof*⟩

lemma *norm-zero-iff*:
  assumes $\langle\!\langle a \rangle\!\rangle = 0$
  shows $a = 0_g$
  ⟨*proof*⟩

definition *norms* :: *real set* **where**
 $norms = \{x.\ \exists\ a.\ x = \langle\!\langle a \rangle\!\rangle\} \cup \{x.\ \exists\ a.\ x = -\ \langle\!\langle a \rangle\!\rangle\}$

lemma *norm-in-norms* [*simp*]:
  shows $\langle\!\langle a \rangle\!\rangle \in norms$
  ⟨*proof*⟩

lemma *minus-norm-in-norms* [*simp*]:
  shows $-\ \langle\!\langle a \rangle\!\rangle \in norms$
  ⟨*proof*⟩

end

locale *gyrocarrier* = *gyrocarrier′* +
  assumes *inner-gyroauto-invariant*: $\bigwedge u\ v\ a\ b.\ (gyr\ u\ v\ a) \cdot (gyr\ u\ v\ b) = a \cdot b$
begin

lemma *norm-gyr*: $\langle\!\langle gyr\ u\ v\ a \rangle\!\rangle = \langle\!\langle a \rangle\!\rangle$
  ⟨*proof*⟩

end

locale *pre-gyrovector-space* = *gyrocarrier* +
  fixes *scale* :: $real \Rightarrow {'a} \Rightarrow {'a}$ (**infixl** $\otimes$ *105*)
  assumes *scale-1*:
    $\bigwedge a :: {'a}.$
        $1 \otimes a = a$
  assumes *scale-distrib*:
    $\bigwedge (r1 :: real)\ (r2 :: real)\ (a :: {'a}).$
        $(r1 + r2) \otimes a = r1 \otimes a \oplus r2 \otimes a$
  assumes *scale-assoc*:
    $\bigwedge (r1 :: real)\ (r2 :: real)\ (a :: {'a}).$
        $(r1 * r2) \otimes a = r1 \otimes (r2 \otimes a)$
  assumes *scale-prop1*:
    $\bigwedge (r :: real)\ (a :: {'a}).$
        $[\![r \neq 0;\ a \neq 0_g]\!] \Longrightarrow to\text{-}carrier\ (|r| \otimes a)\ /_R\ \langle\!\langle r \otimes a \rangle\!\rangle = to\text{-}carrier\ a\ /_R\ \langle\!\langle a \rangle\!\rangle$
  assumes *gyroauto-property*:
    $\bigwedge (u :: {'a})\ (v :: {'a})\ (r :: real)\ (a :: {'a}).$
        $gyr\ u\ v\ (r \otimes a) = r \otimes (gyr\ u\ v\ a)$

assumes *gyroauto-id*:
$\bigwedge$ $(r1 :: real)$ $(r2 :: real)$ $(v :: 'a)$.
$gyr\ (r1 \otimes v)\ (r2 \otimes v) = id$
begin

lemma *scale-minus1*:
shows $(-1) \otimes a = \ominus a$
⟨*proof*⟩

lemma *minus-norm*:
shows $\langle\!\langle \ominus a \rangle\!\rangle = \langle\!\langle a \rangle\!\rangle$
⟨*proof*⟩

(6.3)

lemma *scale-minus*:
shows $(-r) \otimes a = \ominus (r \otimes a)$
⟨*proof*⟩

lemma *scale-minus'*:
shows $k \otimes (\ominus a) = \ominus (k \otimes a)$
⟨*proof*⟩

lemma *zero-otimes* [*simp*]:
shows $0 \otimes x = 0_g$
⟨*proof*⟩

lemma *times-zero* [*simp*]:
shows $t \otimes 0_g = 0_g$
⟨*proof*⟩

Theorem 6.4 (6.4)

lemma *monodistributive*:
shows $r \otimes (r1 \otimes a \oplus r2 \otimes a) = r \otimes (r1 \otimes a) \oplus r \otimes (r2 \otimes a)$
⟨*proof*⟩

lemma *times2*: $2 \otimes a = a \oplus a$
⟨*proof*⟩

lemma *twosum*: $2 \otimes (a \oplus b) = a \oplus (2 \otimes b \oplus a)$
⟨*proof*⟩

definition *gyrodistance* :: $'a \Rightarrow 'a \Rightarrow real$ $(d_\oplus)$ **where**
$d_\oplus\ a\ b = \langle\!\langle \ominus a \oplus b \rangle\!\rangle$

lemma $d_\oplus\ a\ b = \langle\!\langle b \ominus_b a \rangle\!\rangle$
⟨*proof*⟩

lemma *gyrodistance-metric-nonneg*:
shows $d_\oplus\ a\ b \geq 0$

⟨*proof*⟩

lemma *gyrodistance-metric-zero-iff*:
shows $d_{\oplus}\ a\ b = 0 \longleftrightarrow a = b$
⟨*proof*⟩

lemma *gyrodistance-metric-sym*:
shows $d_{\oplus}\ a\ b = d_{\oplus}\ b\ a$
⟨*proof*⟩

lemma *equation-solving*:
assumes $x \oplus y = a \ominus x \oplus y = b$
shows $x = (1/2) \otimes (a \ominus_{cb}\ b) \wedge y = (1/2) \otimes (a \ominus_{cb}\ b) \oplus b$
⟨*proof*⟩

lemma *double-plus*: $(2 \otimes a) \oplus b = a \oplus (a \oplus b)$
⟨*proof*⟩

lemma *I6-33*:
shows $(1/2) \otimes (a \ominus_{cb}\ b) = (-1/2) \otimes (b \ominus_{cb}\ a)$
⟨*proof*⟩

lemma *I6-34*:
shows $(1/2) \otimes (a \ominus_{cb}\ b) \oplus b = (1/2) \otimes (b \ominus_{cb}\ a) \oplus a$
⟨*proof*⟩

lemma *I6-35*:
shows $gyr\ b\ a = gyr\ b\ ((1/2) \otimes (a \ominus_{cb}\ b) \oplus b) \circ (gyr\ ((1/2) \otimes (a \ominus_{cb}\ b) \oplus b)\ a)$
⟨*proof*⟩

lemma *double-half*:
shows $2 \otimes ((1\ /\ 2) \otimes a) = a$
⟨*proof*⟩

lemma *I6-38*:
shows $a \oplus (1/2) \otimes (\ominus\ a \oplus_c\ b) = (1/2) \otimes (a \oplus b)$
⟨*proof*⟩

lemma *I6-39*:
shows $a \oplus (1/2) \otimes (\ominus\ a \oplus b) = (1/2) \otimes (a \oplus_c\ b)$
⟨*proof*⟩

lemma *I6-40*:
shows $gyr\ ((r + s) \otimes a)\ b\ x = gyr\ (r{\otimes}a)\ (s{\otimes}a \oplus b)\ (gyr\ (s{\otimes}a)\ b\ x)$
⟨*proof*⟩

definition *collinear* :: *'a => 'a => 'a => bool* **where**

collinear x y z ⟷ ($y = z \vee (\exists\, t::real.\ (x = y \oplus t \otimes (\ominus\ y \oplus z)))$)

lemma *collinear-aab*:
shows *collinear a a b*
⟨*proof*⟩

lemma *collinear-bab*:
shows *collinear b a b*
⟨*proof*⟩

lemma *T6-20*:
assumes *collinear p1 a b collinear p2 a b* $a \neq b$ $p1 \neq p2$
shows ∀ *x*. (*collinear x p1 p2* ⟶ *collinear x a b*)
⟨*proof*⟩

lemma *T6-20-1*:
assumes *collinear p1 a b collinear p2 a b* $p1 \neq p2$ $a \neq b$
shows ∀ *x*. (*collinear x a b* ⟶ *collinear x p1 p2*)
⟨*proof*⟩

lemma *collinear-sym1*:
assumes *collinear a b c*
shows *collinear b a c*
⟨*proof*⟩

lemma *collinear-sym2*:
assumes *collinear a b c*
shows *collinear a c b*
⟨*proof*⟩

lemma *collinear-transitive*:
assumes *collinear a b c collinear d b c* $b \neq c$
shows *collinear a d b*
⟨*proof*⟩

lemma *collinear-translate'*:
shows $x = u \oplus t \otimes (\ominus\ u \oplus v) \longleftrightarrow$
$(\ominus\ a \oplus x) = (\ominus\ a \oplus u) \oplus t \otimes (\ominus\ (\ominus\ a \oplus u) \oplus (\ominus\ a \oplus v))$
⟨*proof*⟩

definition *translate* **where**
translate a x = $\ominus\ a \oplus x$

lemma *collinear-translate*:
shows *collinear u v w* ⟷ *collinear* (*translate a u*) (*translate a v*) (*translate a w*)
⟨*proof*⟩

definition *gyroline* :: *'a* ⇒ *'a* ⇒ *'a set* **where**

gyroline a b = {*x*. *collinear x a b*}

definition *between* :: *'a* => *'a* => *'a* => *bool* **where**
between x y z ⟷ (∃ *t*::*real*. *0* ≤ *t* ∧ *t* ≤ *1* ∧ *y* = *x* ⊕ *t* ⊗ (⊖ *x* ⊕ *z*))

lemma *between-xxy* [*simp*]:
shows *between x x y*
⟨*proof*⟩

lemma *between-xyy* [*simp*]:
shows *between x y y*
⟨*proof*⟩

lemma *between-xyx*:
assumes *between x y x*
shows *y* = *x*
⟨*proof*⟩

lemma *between-translate*:
shows *between u v w* ⟷ *between* (*translate a u*) (*translate a v*) (*translate a w*)
⟨*proof*⟩

definition *distance* **where**
distance u v = ⟪⊖ *u* ⊕ *v*⟫

lemma *distance-translate*:
shows *distance u v* = *distance* (*translate a u*) (*translate a v*)
⟨*proof*⟩

end

locale *gyrocarrier-norms-embed'* = *gyrocarrier' to-carrier*
for *to-carrier* :: *'a*::*gyrocommutative-gyrogroup* ⇒ *'b*::{*real-inner*, *real-normed-algebra-1*}
+
assumes *norms-carrier*: *of-real* ' *norms* ⊆ *carrier*
begin

definition *of-real'* :: *real* ⇒ *'a* **where**
of-real' = *of-carrier* ∘ *of-real*

definition *reals* :: *'a set* **where**
reals = *of-carrier* ' *of-real* ' *norms*

lemma *bij-reals-norms*:
shows *bij-betw of-real' norms reals*
⟨*proof*⟩

lemma *inj-on-of-real'*:
shows *inj-on of-real' norms*

⟨*proof*⟩

definition *to-real* :: *'b* ⇒ *real* **where**
to-real = *the-inv-into norms of-real*

lemma *to-real* [*simp*]:
assumes $x \in$ *norms*
shows *to-real* (*of-real* x) = x
⟨*proof*⟩

lemma *of-real* [*simp*]:
assumes $x \in$ *of-real* ` *norms*
shows *of-real* (*to-real* x) = x
⟨*proof*⟩

definition *to-real'* :: *'a* ⇒ *real* **where**
to-real' = *to-real* ∘ *to-carrier*

lemma *bij-betw-to-real'*:
bij-betw to-real' reals norms
⟨*proof*⟩

lemma *to-real'* [*simp*]:
assumes $x \in$ *norms*
shows *to-real'* (*of-real'* x) = x
⟨*proof*⟩

lemma *of-real'* [*simp*]:
assumes $x \in$ *reals*
shows *of-real'* (*to-real'* x) = x
⟨*proof*⟩

lemma *to-real'-norm* [*simp*]:
shows *to-real'* (*of-real'* (⟪a⟫)) = (⟪a⟫)
⟨*proof*⟩

lemma *gyronorm-of-real'*:
assumes $x \in$ *norms*
shows ⟪*of-real'* x⟫ = *abs* x
⟨*proof*⟩

lemma *gyronorm-abs-to-real'*:
assumes $x \in$ *reals*
shows *abs* (*to-real'* x) = ⟪x⟫
⟨*proof*⟩

definition *oplusR* :: *real* ⇒ *real* ⇒ *real* (**infixl** $\oplus_R$ *100*) **where**
$a \oplus_R b$ = *to-real'* (*of-real'* a ⊕ *of-real'* b)

definition *oinvR* :: *real* ⇒ *real* ($\ominus_R$) **where**
$\ominus_R$ *a* = *to-real′* (⊖ (*of-real′ a*))

end

locale *gyrocarrier-norms-embed* = *gyrocarrier-norms-embed′* +
fixes *scale* :: *real* ⇒ *′a* ⇒ *′a* (**infixl** ⊗ *105*)
assumes *oplus-reals*: ⋀ *a b*. ⟦*a* ∈ *reals*; *b* ∈ *reals*⟧ ⟹ *a* ⊕ *b* ∈ *reals*
assumes *oinv-reals*: ⋀ *a*. *a* ∈ *reals* ⟹ ⊖ *a* ∈ *reals*
assumes *otimes-reals*: ⋀ *a r*. *a* ∈ *reals* ⟹ *r* ⊗ *a* ∈ *reals*
begin

definition *otimesR* :: *real* ⇒ *real* ⇒ *real* (**infixl** $\otimes_R$ *100*) **where**
r $\otimes_R$ *a* = *to-real′* (*r* ⊗ (*of-real′ a*))

lemma *oplusR-norms*:
shows ⋀ *a b*. ⟦*a* ∈ *norms*; *b* ∈ *norms*⟧ ⟹ *a* $\oplus_R$ *b* ∈ *norms*
⟨*proof*⟩

lemma *oinvR-norms*:
shows ⋀ *a*. *a* ∈ *norms* ⟹ $\ominus_R$ *a* ∈ *norms*
⟨*proof*⟩

lemma *otimesR-norms*:
shows ⋀ *a r*. *a* ∈ *norms* ⟹ *r* $\otimes_R$ *a* ∈ *norms*
⟨*proof*⟩

lemma *of-real′-oplusR* [*simp*]:
shows *of-real′* ((⟪*a*⟫) $\oplus_R$ (⟪*b*⟫)) = (*of-real′* (⟪*a*⟫) ⊕ *of-real′* (⟪*b*⟫))
⟨*proof*⟩

lemma *of-real′-otimesR* [*simp*]:
shows *of-real′* (*r* $\otimes_R$ (⟪*a*⟫)) = *r* ⊗ (*of-real′* (⟪*a*⟫))
⟨*proof*⟩

lemma *of-real′-oinvR* [*simp*]:
shows *of-real′* ($\ominus_R$ (⟪*a*⟫)) = ⊖ (*of-real′* (⟪*a*⟫))
⟨*proof*⟩

end

locale *gyrovector-space-norms-embed* =
gyrocarrier to-carrier +
gyrocarrier-norms-embed to-carrier +
pre-gyrovector-space to-carrier
for *to-carrier* :: *′a*::*gyrocommutative-gyrogroup* ⇒ *′b*::{*real-inner*, *real-normed-algebra-1*}
+
assumes *homogeneity*:
⋀ (*r* :: *real*) (*a* :: *′a*).

$\langle\!\langle r \otimes a\rangle\!\rangle = |r| \otimes_R (\langle\!\langle a\rangle\!\rangle)$
assumes *gyrotriangle*:
$\bigwedge (a :: 'a)\ (b :: 'a).$
$\langle\!\langle a \oplus b\rangle\!\rangle \le (\langle\!\langle a\rangle\!\rangle) \oplus_R (\langle\!\langle b\rangle\!\rangle)$
begin

lemma *gyrodistance-gyrotriangle*:
shows $d_\oplus\ a\ c \le d_\oplus\ a\ b\ \ \oplus_R\ d_\oplus\ b\ c$
⟨*proof*⟩

end

end
theory *VectorSpace*
imports *Main HOL.Real HOL−Types-To-Sets.Linear-Algebra-On-With*

begin

locale *vector-space-with-domain* =
fixes *dom* :: *'a set*
and *add* :: $'a \Rightarrow 'a \Rightarrow 'a$
and *zero* :: $'a$
and *smult* :: $real \Rightarrow 'a \Rightarrow 'a$
assumes *add-closed*: $[\![x \in dom;\ y \in dom]\!] \Longrightarrow add\ x\ y \in dom$
and *zero-in-dom*: $zero \in dom$
and *add-assoc*: $[\![x \in dom;\ y \in dom;\ z\in dom]\!] \Longrightarrow add\ (add\ x\ y)\ z = add\ x\ (add\ y\ z)$
and *add-comm*: $[\![x \in dom;\ y \in dom]\!] \Longrightarrow add\ x\ y = add\ y\ x$
and *add-zero*: $[\![x \in dom]\!] \Longrightarrow add\ x\ zero = x$
and *add-inv*: $x \in dom \Longrightarrow \exists\, y \in dom.\ add\ x\ y = zero$
and *smult-closed*: $[\![x \in dom]\!] \Longrightarrow smult\ a\ x \in dom$
and *smult-distr-sadd*: $[\![x \in dom]\!] \Longrightarrow smult\ (a + b)\ x = add\ (smult\ a\ x)\ (smult\ b\ x)$
and *smult-assoc*: $[\![x \in dom]\!] \Longrightarrow smult\ a\ \ (smult\ b\ x) = smult\ (a * b)\ \ x$
and *smult-one*: $[\![x \in dom]\!] \Longrightarrow smult\ 1\ x = x$
and *smult-distr-sadd2*: $[\![x \in dom;\ y\in dom]\!] \Longrightarrow smult\ a\ (add\ x\ y) = add\ (smult\ a\ x)\ (smult\ a\ y)$

begin

lemma *inv-unique*:
assumes $a\in dom\ z1\in dom\ z2\in dom$
$add\ a\ z1 = zero$
$add\ a\ z2 = zero$
shows $z1=z2$
⟨*proof*⟩

definition $inv{::}'a\Rightarrow'a$ **where**

inv a = (*if a*∈*dom then* (*THE z*. (*z*∈*dom* ∧ *add a z* = *zero*)) *else undefined*)

definition *minus*::′*a*⇒′*a*⇒′*a* **where**
minus a b = (*if a*∈*dom* ∧ *b*∈*dom then add a* (*inv b*) *else undefined*)

lemma *module-on-with-is-this*:
shows *module-on-with dom add minus inv zero smult*
⟨*proof*⟩

lemma *vector-space-on-with-is-this*:
shows *vector-space-on-with dom add minus inv zero smult*
⟨*proof*⟩

end

end
theory *Abe*
imports *GyroGroup HOL−Analysis.Inner-Product HOL.Real-Vector-Spaces VectorSpace*
begin

locale *one-dim-vector-space-with-domain* =
vector-space-with-domain +
assumes ∀ *y*. ∀ *x*. (*y*∈ *dom* ∧
x∈ *dom* ∧ *x*≠*zero* ⟶ (∃!*r*::*real*. *y* = *smult r x*))

locale *GGV* =
fixes *fi* ::′*a*::*gyrocommutative-gyrogroup* ⇒ ′*b*::*real-inner*
fixes *scale* ::*real* ⇒ ′*a* ⇒ ′*a*
fixes *plus*′::*real* ⇒ *real* ⇒ *real*
fixes *smult*′::*real* ⇒ *real* ⇒ *real*

assumes *inj fi*
assumes *norm* (*fi* (*gyr u v a*)) = *norm* (*fi a*)
assumes *scale 1 a* = *a*
assumes *scale* (*r1*+*r2*) *a* = (*scale r1 a*) ⊕ (*scale r2 a*)
assumes *scale* (*r1*∗*r2*) *a* = *scale r1* (*scale r2 a*)
assumes (*a*≠*gyrozero* ∧ *r*≠*0*)⟶ (*fi* (*scale* |*r*| *a*)) $/_R$ (*norm* (*fi* (*scale r a*))) = (*fi a*) $/_R$ (*norm* (*fi a*))
assumes *gyr u v* (*scale r a*) = *scale r* (*gyr u v a*)
assumes *gyr* (*scale r1 v*) (*scale r2 v*) = *id*
assumes *vector-space-with-domain* {*x*.∃ *a*. *x* = *norm* (*fi a*) ∨ *x* = − *norm* (*fi a*)} *plus*′ *0 smult*′
assumes *norm* (*fi* (*scale r a*)) = *smult*′ |*r*| (*norm* (*fi a*))
assumes *norm* (*fi* (*a* ⊕ *b*)) = *plus*′ (*norm* (*fi a*)) (*norm* (*fi b*))
begin

end

class *gyrolinear-space* =
gyrocommutative-gyrogroup +
fixes *scale* :: *real* ⇒ *'a*::*gyrocommutative-gyrogroup* ⇒ *'a* (**infixl** ⊗ *105*)
assumes *scale-1*: ⋀ *a* :: *'a*. *1* ⊗ *a* = *a*
assumes *scale-distrib*: ⋀ (*r1* :: *real*) (*r2* :: *real*) (*a* :: *'a*). (*r1* + *r2*) ⊗ *a* = *r1* ⊗ *a* ⊕ *r2* ⊗ *a*
assumes *scale-assoc*: ⋀ (*r1* :: *real*) (*r2* :: *real*) (*a* :: *'a*). (*r1* ∗ *r2*) ⊗ *a* = *r1* ⊗ (*r2* ⊗ *a*)
assumes *gyroauto-property*: ⋀ (*u* :: *'a*) (*v* :: *'a*) (*r* :: *real*) (*a* :: *'a*). *gyr u v* (*r* ⊗ *a*) = *r* ⊗ (*gyr u v a*)
assumes *gyroauto-id*: ⋀ (*r1* :: *real*) (*r2* :: *real*) (*v* :: *'a*). *gyr* (*r1* ⊗ *v*) (*r2* ⊗ *v*) = *id*

begin

end

locale *normed-gyrolinear-space* =
fixes *norm'*::*'a*::*gyrolinear-space* ⇒ *real*
fixes *f*::*real* ⇒ *real*
assumes ∀ *a*::*'a*. (*norm' a* ≥ *0*)
assumes ∀ *y*::*real*. (*y*∈ (*norm'* ‘ *UNIV*) ⟶ (*f y*) ≥ *0*)
assumes *bij-betw f* (*norm'* ‘ *UNIV*) {*x*::*real*. *x*≥*0*}
assumes ∀ *y*::*real*. ∀ *z*::*real*. ((*y*∈ *norm'* ‘ *UNIV* ∧
z∈ *norm'* ‘ *UNIV* ∧ *y*>*z*)⟶ (*f y*) > (*f z*))

assumes ∀ *x*::*'a*. ∀ *y*::*'a*. *f*(*norm'* (*gyroplus x y*)) ≤ (*f* (*norm' x*)) + (*f* (*norm' y*))
assumes *f* (*norm'* (*scale r x*)) = |*r*| ∗ (*f* (*norm' x*))
assumes *norm'* (*gyr u v x*) = *norm' x*
assumes ∀ *x*::*'a*. ((*norm' x*) = *0* ⟷ *x* = *gyrozero*)
begin

definition *norms*::*real set* **where**
norms = *norm'* ‘ *UNIV*

definition *norms-neg*::*real set* **where**
norms-neg = (λ*x*. −*1* ∗ *norm' x*) ‘ *UNIV*

definition *norms-all*::*real set* **where**
norms-all = *norms* ∪ *norms-neg*

lemma *norms-neq-not-empty*:
shows *norms-neg* ≠ {}
⟨*proof*⟩

lemma *zero-only-norms-norms-neg*:
assumes *x∈norms x∈norms-neg*
shows *x=0*
⟨*proof*⟩

lemma *a1-a2*:
shows ∃*f'*:: *real* ⇒ *real*. ((∀*x*::*real*. ∀*y*::*real*. (*x*∈*norms-all* ∧ *y* ∈*norms-all* ∧ *x*>*y*)⟶ (*f' x*) > (*f' y*))
∧ (*f' 0*) = *0* ∧ *bij-betw f' norms-all UNIV*)
⟨*proof*⟩

end

locale *normed-gyrolinear-space'* =
fixes *norm'*::*'a*::*gyrolinear-space* ⇒ *real*
fixes *f'*::*real* ⇒ *real*
assumes ∀*a*::*'a*. (*norm' a* ≥ *0*)
assumes *bij-betw f'* ((*norm'* ‘ *UNIV*) ∪ ((λ*x*. −*1* ∗ *norm' x*) ‘ *UNIV*)) *UNIV*
assumes ∀*y*::*real*. ∀*z*::*real*. ((*y*∈ ((*norm'* ‘ *UNIV*) ∪ ((λ*x*. −*1* ∗ *norm' x*) ‘ *UNIV*)) ∧
z∈ ((*norm'* ‘ *UNIV*) ∪ ((λ*x*. −*1* ∗ *norm' x*) ‘ *UNIV*)) ∧ *y*>*z*)⟶ (*f' y*) > (*f' z*))
assumes *f' 0* = *0*
assumes ∀*x*::*'a*. ∀*y*::*'a*. *f'*(*norm'* (*gyroplus x y*)) ≤ (*f'* (*norm' x*)) + (*f'* (*norm' y*))
assumes *f'* (*norm'* (*scale r x*)) = |*r*| ∗ (*f'* (*norm' x*))
assumes *norm'* (*gyr u v x*) = *norm' x*
assumes ∀*x*::*'a*. ((*norm' x*) = *0* ⟷ *x* = *gyrozero*)
begin

definition *norms*::*real set* **where**
norms = *norm'* ‘ *UNIV*

definition *norms-neg*::*real set* **where**
norms-neg = (λ*x*. −*1* ∗ *norm' x*) ‘ *UNIV*

definition *norms-all*::*real set* **where**
norms-all = *norms* ∪ *norms-neg*

lemma *norms-neq-not-empty*:
shows *norms-neg* ≠ {}
⟨*proof*⟩

lemma *zero-only-norms-norms-neg*:
assumes *x∈norms x∈norms-neg*
shows *x=0*
⟨*proof*⟩

definition *norm-oplus-f*::*real* ⇒ *real* ⇒ *real* (**infixl** $\oplus_f$ *105*)
  where $a \oplus_f b$ = (*if* (*a*∈*norms-all* ∧ *b*∈*norms-all*) *then* (*inv-into norms-all f'*) ((*f' a*) + (*f' b*))
else undefined)

definition *norm-otimes-f*::*real* ⇒ *real* ⇒ *real* (**infixl** $\otimes_f$ *105*)
  where $r \otimes_f a$ = (*if* (*a*∈*norms-all*) *then* (*inv-into norms-all f'*) (*r* ∗ (*f' a*))
else undefined)

lemma *vector-space-of-norms*:
  shows *vector-space-with-domain norms-all norm-oplus-f 0 norm-otimes-f*
⟨*proof*⟩

lemma *r2*:
  shows *norm'* (*x* ⊕ *y*) ≤ (*norm' x*) $\oplus_f$ (*norm' y*)
⟨*proof*⟩

lemma *r3*:
  shows *norm'* (*r* ⊗ *x*) = |*r*| $\otimes_f$ (*norm' x*)
  ⟨*proof*⟩

lemma *one-dim-vs*:
 shows *one-dim-vector-space-with-domain norms-all norm-oplus-f 0 norm-otimes-f*
⟨*proof*⟩

end

locale *normed-gyrolinear-space''* =
  fixes *norm'*::*'a*::*gyrolinear-space* ⇒ *real*
  fixes *oplus'*::*real* ⇒ *real* ⇒ *real*
  fixes *otimes'*::*real*⇒*real* ⇒ *real*
  assumes ∀ *a*::*'a*. (*norm' a* ≥ *0*)
  assumes *ax-space*: *one-dim-vector-space-with-domain* ((*norm'* ‘ *UNIV*) ∪ ((λ*x*. −*1* ∗ *norm' x*) ‘ *UNIV*))
      oplus' 0 otimes'
  assumes *ax3*: ∀ *x*::*'a*. ∀ *y*::*'a*. (*norm'* (*gyroplus x y*)) ≤ *oplus'* (*norm' x*) (*norm' y*)
  assumes (*norm'* (*scale r x*)) = *otimes'* |*r*| (*norm' x*)
  assumes *norm'* (*gyr u v x*) = *norm' x*
  assumes ∀ *x*::*'a*. ((*norm' x*) = *0* ⟷ *x* = *gyrozero*)
begin

definition *norms*::*real set* **where**
  norms = *norm'* ‘ *UNIV*

definition *norms-neg*::*real set* **where**
norms-neg = (λ*x*. −*1* ∗ *norm′ x*) ‘ *UNIV*

definition *norms-all*::*real set* **where**
norms-all = *norms* ∪ *norms-neg*

lemma *norms-neq-not-empty*:
shows *norms-neg* ≠ {}
⟨*proof*⟩

lemma *zero-only-norms-norms-neg*:
assumes *x*∈*norms* *x*∈*norms-neg*
shows *x*=*0*
⟨*proof*⟩

lemma *not-trivial-domen-has-pos*:
assumes ∃ *x*. (*x*∈*norms-all* ∧ *x*≠*0*)
shows ∃ *x*. (*x*∈*norms* ∧ *x*≠*0*)
⟨*proof*⟩

lemma *iso-with-real*:
assumes ∃ *x*. (*x*∈*norms-all* ∧ *x*≠*0*)
shows ∃ *g*. (*bij-betw g norms-all UNIV* ∧ (*g 0*) = *0* ∧
(∀ *u*.∀ *v*. (*u*∈*norms-all* ∧ *v*∈*norms-all* ⟶ *g* (*oplus′ u v*) = (*g u*) + (*g v*)))
∧ (∀ *u*.∀ *r*::*real*. (*u*∈*norms-all* ⟶ *g* (*otimes′ r u*) = *r*∗(*g u*)))
)
⟨*proof*⟩

definition *g-iso*::(*real*⇒*real*)⇒*bool* **where**
g-iso g ⟷ (*bij-betw g norms-all UNIV* ∧ (*g 0*) = *0* ∧
(∀ *u*.∀ *v*. (*u*∈*norms-all* ∧ *v*∈*norms-all* ⟶ *g* (*oplus′ u v*) = (*g u*) + (*g v*)))
∧ (∀ *u*.∀ *r*::*real*. (*u*∈*norms-all* ⟶ *g* (*otimes′ r u*) = *r*∗(*g u*))))

lemma *iso-neg-with-real*:
assumes ∃ *x*. (*x*∈*norms-all* ∧ *x*≠*0*)
shows *g-iso g* ⟶ *g-iso* (λ*x*. −*1* ∗ (*g x*))
⟨*proof*⟩

lemma *iso-with-real-positive-on-norms*:
assumes ∃ *x*. (*x*∈*norms-all* ∧ *x*≠*0*)
shows ∃ *g*. (*g-iso g* ∧ (∀ *x*.(*x*∈*norms* ⟶ (*g x*)≥*0*))
∧ *bij-betw* (λ*x*. *if x* ∈ *norms then* (*g x*) *else undefined*) *norms* {*r*::*real*. *r*≥*0*})
⟨*proof*⟩

lemma *comparing-norms-help*:

assumes $x{\in}norms$ $y{\in}norms\text{-}all$
$x{\le}y$
shows $y{\in}$ $norms$
⟨*proof*⟩

lemma *existence-of-f*:
assumes $\exists\, x.\ (x{\in}norms\text{-}all \wedge x{\neq}0)$
shows $\exists f.\ (bij\text{-}betw\ f\ norms\ \{x{::}real.\ x{\ge}0\}$
$\wedge$ $(\forall\, y{::}real.\ \forall\, z{::}real.\ ((\ y{\in}\ norms \wedge$
$z{\in}\ \ norms \wedge y{>}z){\longrightarrow}\ (f\ y) > (f\ z)))$
$\wedge$ $(\forall\, x.\ \forall\, y.\ f(norm'\ (x \oplus y)) \le (f\ (norm'\ x)) + (f\ (norm'\ y)))$
$\wedge$ $(\forall\, r{::}real.\ (\forall\, x.\ (f\ (norm'\ (r \otimes x)) = |r| * (f\ (norm'\ x))))))$
⟨*proof*⟩

end

end
theory *GyroVectorSpaceIsomorphism*
imports *GyroVectorSpace*
begin

locale $gyrocarrier\text{-}isomorphism' =$
$gyrocarrier\text{-}norms\text{-}embed'\ to\text{-}carrier\ +$
$gyrocarrier\ to\text{-}carrier\ +$
G: $gyrocarrier\text{-}norms\text{-}embed'\ to\text{-}carrier'$
for $to\text{-}carrier :: 'a{::}gyrocommutative\text{-}gyrogroup \Rightarrow 'b{::}\{real\text{-}inner,\ real\text{-}normed\text{-}algebra\text{-}1\}$
and
$to\text{-}carrier' :: 'c{::}gyrocommutative\text{-}gyrogroup \Rightarrow 'd{::}\{real\text{-}inner,\ real\text{-}normed\text{-}algebra\text{-}1\}$
+
fixes $\varphi :: 'a \Rightarrow 'c$
begin

definition $\varphi_R :: real \Rightarrow real$ **where**
$\varphi_R\ x = G.to\text{-}real'\ (\varphi\ (of\text{-}real'\ x))$

end

locale $gyrocarrier\text{-}isomorphism = gyrocarrier\text{-}isomorphism' +$
assumes φbij [*simp*]:
$bij\ \varphi$
assumes $\varphi plus$ [*simp*]:
$\bigwedge u\ v :: 'a.\ \varphi\ (u \oplus v) = \varphi\ u \oplus \varphi\ v$
assumes $\varphi inner\text{-}unit$:
$\bigwedge u\ v :: 'a.\ [\![u \neq 0_g;\ v \neq 0_g]\!] \Longrightarrow$
$inner\ (to\text{-}carrier'\ (\varphi\ u)\ /_R\ G.gyronorm\ (\varphi\ u))\ (to\text{-}carrier'\ (\varphi\ v)$

$/_R$ *G.gyronorm* (φ *v*)) =
inner (*to-carrier u* $/_R$ *gyronorm u*) (*to-carrier v* $/_R$ *gyronorm v*)
assumes φ_R*gyronorm* [*simp*]:
$\bigwedge$ *a*. φ_R (*gyronorm a*) = *G.gyronorm* (φ *a*)
begin

lemma φ*inv*φ [*simp*]:
shows φ (*inv* φ *a*) = *a*
⟨*proof*⟩

lemma *inv*$\varphi\varphi$ [*simp*]:
shows (*inv* φ) (φ *a*) = *a*
⟨*proof*⟩

lemma φ*zero* [*simp*]:
shows φ 0_g = 0_g
⟨*proof*⟩

lemma φ*minus* [*simp*]:
shows φ ($\ominus$ *a*) = $\ominus$ (φ *a*)
⟨*proof*⟩

lemma *inv*φ*plus*[*simp*]:
shows (*inv* φ)(*a* $\oplus$ *b*) = *inv* φ *a* $\oplus$ *inv* φ *b*
⟨*proof*⟩

lemma φ*gyr* [*simp*]:
shows φ (*gyr u v a*) = *gyr* (φ *u*) (φ *v*) (φ *a*)
⟨*proof*⟩

lemma *inv*φ*gyr* [*simp*]:
shows (*inv* φ) (*gyr u v a*) = *gyr* (*inv* φ *u*) (*inv* φ *v*) (*inv* φ *a*)
⟨*proof*⟩

lemma φ*inner*:
assumes *u* $\neq$ 0_g *v* $\neq$ 0_g
shows *G.gyroinner* (φ *u*) (φ *v*) =
(*G.gyronorm* (φ *u*) / *gyronorm u*) $*_R$ (*G.gyronorm* (φ *v*) / *gyronorm v*)
$*_R$ *gyroinner u v*
⟨*proof*⟩

lemma *gyronorm′gyr*:
shows *G.gyronorm* (*gyr u v a*) = *G.gyronorm a*
⟨*proof*⟩

end

sublocale *gyrocarrier-isomorphism* ⊆ *gyrocarrier to-carrier′*
⟨*proof*⟩

locale *pre-gyrovector-space-isomorphism'* =
pre-gyrovector-space to-carrier scale +
gyrocarrier-norms-embed' to-carrier +
GC: *gyrocarrier-norms-embed' to-carrier'*
for *to-carrier* :: *'a*::*gyrocommutative-gyrogroup* ⇒ *'b*::{*real-inner*, *real-normed-algebra-1*}
and
to-carrier' :: *'c*::*gyrocommutative-gyrogroup* ⇒ *'d*::{*real-inner*, *real-normed-algebra-1*}
and
scale :: *real* ⇒ *'a* ⇒ *'a* **and**
scale' :: *real* ⇒ *'c* ⇒ *'c* +
fixes φ :: *'a* ⇒ *'c*

sublocale *pre-gyrovector-space-isomorphism'* ⊆ *gyrocarrier-isomorphism'*
⟨*proof*⟩

locale *pre-gyrovector-space-isomorphism* =
pre-gyrovector-space-isomorphism' +
gyrocarrier-isomorphism +
assumes *φscale* [*simp*]:
⋀ *r* :: *real*. ⋀ *u* :: *'a*. φ (*scale r u*) = *scale' r* (φ *u*)
begin

lemma *scale'-1*:
shows *scale' 1 a* = *a*
⟨*proof*⟩

lemma *scale'-distrib*:
shows *scale'* (*r1* + *r2*) *a* = *scale' r1 a* ⊕ *scale' r2 a*
⟨*proof*⟩

lemma *scale'-assoc*:
shows *scale'* (*r1* ∗ *r2*) *a* = *scale' r1* (*scale' r2 a*)
⟨*proof*⟩

lemma *scale'-gyroauto-id*:
shows *gyr* (*scale' r1 v*) (*scale' r2 v*) = *id*
⟨*proof*⟩

lemma *scale'-gyroauto-property*:
shows *gyr u v* (*scale' r a*) = *scale' r* (*gyr u v a*)
⟨*proof*⟩

end

locale *gyrovector-space-isomorphism'* =
pre-gyrovector-space-isomorphism +
gyrovector-space-norms-embed scale +
GC: *gyrocarrier-norms-embed to-carrier' scale'* +

assumes $\varphi reals$:
  φ ‘ $reals = GC.reals$
begin

lemma $\varphi_R norms$:
  assumes $a \in norms$
  shows $\varphi_R\ a \in GC.norms$
  ⟨*proof*⟩

lemma $\varphi of\text{-}real'$[*simp*]:
  assumes $a \in norms$
  shows $\varphi\ (of\text{-}real'\ a) = GC.of\text{-}real'\ (\varphi_R\ a)$
  ⟨*proof*⟩

lemma $\varphi gyronorm$ [*simp*]:
  shows $\varphi\ (of\text{-}real'\ (gyronorm\ a)) = GC.of\text{-}real'\ (GC.gyronorm\ (\varphi\ a))$
  ⟨*proof*⟩

lemma $\varphi_R plus$ [*simp*]:
  assumes $a \in norms\ b \in norms$
  shows $\varphi_R\ (a \oplus_R b) = GC.oplusR\ (\varphi_R\ a)\ (\varphi_R\ b)$
⟨*proof*⟩

lemma $\varphi_R plus'$ [*simp*]:
  $\varphi_R\ ((\langle\!\langle a\rangle\!\rangle) \oplus_R (\langle\!\langle b\rangle\!\rangle)) = GC.oplusR\ (\varphi_R\ (\langle\!\langle a\rangle\!\rangle))\ (\varphi_R\ (\langle\!\langle b\rangle\!\rangle))$
  ⟨*proof*⟩

lemma $\varphi_R times$ [*simp*]:
  assumes $a \in norms$
  shows $\varphi_R\ (r \otimes_R a) = GC.otimesR\ r\ (\varphi_R\ a)$
⟨*proof*⟩

lemma $\varphi_R times'$ [*simp*]:
  shows $\varphi_R\ (r \otimes_R (\langle\!\langle a\rangle\!\rangle)) = GC.otimesR\ r\ (\varphi_R\ (\langle\!\langle a\rangle\!\rangle))$
  ⟨*proof*⟩

lemma $\varphi_R inv$ [*simp*]:
  assumes $a \in norms$
  shows $\varphi_R\ (\ominus_R\ a) = GC.oinvR\ (\varphi_R\ a)$
⟨*proof*⟩

lemma $\varphi_R inv'$ [*simp*]:
  $\varphi_R\ (\ominus_R\ (\langle\!\langle a\rangle\!\rangle)) = GC.oinvR\ (\varphi_R\ (\langle\!\langle a\rangle\!\rangle))$
  ⟨*proof*⟩

lemma $scale'\text{-}prop1'$:
  assumes $u \neq 0_g\ r \neq 0$
  shows $to\text{-}carrier'\ (\varphi\ (scale\ |r|\ u))\ /_R\ GC.gyronorm\ (\varphi\ (scale\ |r|\ u)) =$

(*to-carrier'* (φ *u*) $/_R$ *GC.gyronorm* (φ *u*)) (**is** *?a* = *?b*)
⟨*proof*⟩

lemma *scale'-prop1*:
assumes *a* ≠ 0_g *r* ≠ *0*
shows *to-carrier'* (*scale'* |*r*| *a*) $/_R$ *GC.gyronorm* (*scale'* *r* *a*) = *to-carrier'* *a* $/_R$ *GC.gyronorm a*
⟨*proof*⟩

lemma *scale'-homogeneity*:
shows *GC.gyronorm* (*scale'* *r* *a*) = *GC.otimesR* |*r*| (*GC.gyronorm a*)
⟨*proof*⟩

end

sublocale *gyrovector-space-isomorphism'* ⊆ *GV*: *pre-gyrovector-space to-carrier' scale'*
⟨*proof*⟩

locale *gyrovector-space-isomorphism* =
gyrovector-space-isomorphism' +
assumes φ_R*mono*:
⋀ *a b*. ⟦*a* ∈ *norms*; *b* ∈ *norms*; *0* ≤ *a*; *a* ≤ *b*⟧ ⟹ φ_R *a* ≤ φ_R *b*
begin

lemma *scale'-triangle*:
shows *GC.gyronorm* (*a* ⊕ *b*) ≤ *GC.oplusR* (*GC.gyronorm a*) (*GC.gyronorm b*)
⟨*proof*⟩

end

sublocale *gyrovector-space-isomorphism* ⊆ *gyrovector-space-norms-embed scale' to-carrier'*
⟨*proof*⟩

locale *gyrocarrier-isomorphism-norms-embed'* = *gyrovector-space-norms-embed scale to-carrier* +
GC: *gyrocarrier-norms-embed' to-carrier'*
for *to-carrier* :: *'a*::*gyrocommutative-gyrogroup* ⇒ *'b*::{*real-inner*, *real-normed-algebra-1*}
and
to-carrier' :: *'c*::*gyrocommutative-gyrogroup* ⇒ *'d*::{*real-inner*, *real-normed-algebra-1*}
and
scale :: *real* ⇒ *'a* ⇒ *'a* +
fixes *scale'* :: *real* ⇒ *'c* ⇒ *'c*
fixes φ :: *'a* ⇒ *'c*
begin

definition φ_R :: *real* ⇒ *real* **where**

$\varphi_R\ x = GC.to\text{-}real'\ (\varphi\ (of\text{-}real'\ x))$

end

locale *gyrocarrier-isomorphism-norms-embed* = *gyrocarrier-isomorphism-norms-embed'* +
assumes *φbij*:
$bij\ \varphi$
assumes *φplus* [*simp*]:
$\bigwedge u\ v :: 'a.\ \varphi\ (u \oplus v) = \varphi\ u \oplus \varphi\ v$
assumes *φscale* [*simp*]:
$\bigwedge r :: real.\ \bigwedge u :: 'a.\ \varphi\ (scale\ r\ u) = scale'\ r\ (\varphi\ u)$
assumes *φreals*:
$\varphi\ `\ reals = GC.reals$
assumes $\varphi_R gyronorm$ [*simp*]:
$\bigwedge a.\ \varphi_R\ (gyronorm\ a) = GC.gyronorm\ (\varphi\ a)$
assumes *GCoinvRminus*:
$\bigwedge a.\ a \in GC.norms \Longrightarrow GC.oinvR\ a = -a$
begin

lemma *φinvφ* [*simp*]:
shows $\varphi\ (inv\ \varphi\ a) = a$
⟨*proof*⟩

lemma *φzero* [*simp*]:
shows $\varphi\ 0_g = 0_g$
⟨*proof*⟩

lemma *φminus* [*simp*]:
shows $\varphi\ (\ominus\ a) = \ominus\ (\varphi\ a)$
⟨*proof*⟩

lemma *φgyronorm* [*simp*]:
shows $\varphi\ (of\text{-}real'\ (gyronorm\ a)) = GC.of\text{-}real'\ (GC.gyronorm\ (\varphi\ a))$
⟨*proof*⟩

lemma $\varphi_R inv'$ [*simp*]:
$\varphi_R\ (\ominus_R\ (\langle\!\langle a \rangle\!\rangle)) = GC.oinvR\ (\varphi_R\ (\langle\!\langle a \rangle\!\rangle))$
⟨*proof*⟩
end

sublocale *gyrocarrier-isomorphism-norms-embed* ⊆ *GV'*: *gyrocarrier-norms-embed to-carrier' scale'*
⟨*proof*⟩

end
theory *MoreComplex*
imports *Complex-Main HOL−Analysis.Inner-Product*
begin

lemma *real-compex-cmod*:
  fixes *r*::*real*
  shows $cmod(r * z) = abs\ r * cmod\ z$
  ⟨*proof*⟩

lemma *cnj-closed-for-unit-disc*:
  assumes $cmod\ z1 < 1$
  shows $cmod\ (cnj\ z1) < 1$
  ⟨*proof*⟩

lemma *mult-closed-for-unit-disc*:
  assumes $cmod\ z1 < 1$ $cmod\ z2 < 1$
  shows $cmod\ (z1*z2) < 1$
  ⟨*proof*⟩

lemma *cnj-cmod*:
  shows $z1 * cnj\ z1 = (cmod\ z1)\hat{}2$
  ⟨*proof*⟩

lemma *cnj-cmod-1*:
  assumes $cmod\ z1 = 1$
  shows $z1 * cnj\ z1 = 1$
  ⟨*proof*⟩

lemma *den-not-zero*:
  assumes $cmod\ a < 1$ $cmod\ b < 1$
  shows $1 + cnj\ a * b \neq 0$
  ⟨*proof*⟩

lemma *cmod-mix-cnj*:
  assumes $cmod\ u < 1$ $cmod\ v < 1$
  shows $cmod\ ((1 + u*cnj\ v) / (1 + v*cnj\ u)) = 1$
  ⟨*proof*⟩

lemma *cnj-mix-ex-real-k*:
  assumes $v \neq 0$
  shows $x * cnj\ v = v * cnj\ x \longleftrightarrow (\exists\ (k::real).\ x = k * v)$
⟨*proof*⟩

lemma *two-inner-cnj*:
  shows $2 * inner\ u\ v = cnj\ u * v + cnj\ v * u$
  ⟨*proof*⟩

abbreviation $cor \equiv complex\text{-}of\text{-}real$

lemma *abs-inner-lt-1*:
  assumes $norm\ u < 1$ $norm\ v < 1$
  shows $abs\ (inner\ u\ v) < 1$

⟨*proof*⟩

lemma *inner-lt-1*:
assumes *norm u < 1 norm v < 1*
shows *inner u v < 1*
⟨*proof*⟩

lemma *inner-def1*:
shows *inner z1 z2 = (z1 ∗ cnj z2 + z2 ∗ cnj z1) / 2*
⟨*proof*⟩

lemma *inner-def2*:
shows *inner z1 z2 = Re (cnj z1 ∗ z2)*
⟨*proof*⟩

end
theory *GammaFactor*
imports *Complex-Main MoreComplex*
begin

definition *gamma-factor* :: *'a::real-inner ⇒ real* (*γ*) **where**
γ u = (if norm u < 1 then
1 / sqrt (1 − (norm u)^2*)*
else
0)

lemma *gamma-factor-nonzero*:
assumes *norm u < 1*
shows *1 / sqrt (1 − (norm u)*^{2}*) ≠ 0*
⟨*proof*⟩

lemma *gamma-factor-increasing*:
fixes *t1 t2 ::real*
assumes *0 ≤ t2 t2 < t1 t1 < 1*
shows *γ t2 < γ t1*
⟨*proof*⟩

lemma *gamma-factor-increase-reverse*:
fixes *t1 t2 :: real*
assumes *t1 ≥ 0 t1 < 1 t2 ≥ 0 t2 < 1*
assumes *γ t1 > γ t2*
shows *t1 > t2*
⟨*proof*⟩

lemma *gamma-factor-u-normu*:
fixes *u :: real*
assumes *0 ≤ u u ≤ 1*
shows *γ u = γ (norm u)*

⟨proof⟩

lemma *gamma-factor-positive*:
assumes *norm u < 1*
shows *γ u > 0*
⟨proof⟩

lemma *norm-square-gamma-factor*:
assumes *norm u < 1*
shows (*norm u*)^*2* = *1* − *1* / (*γ u*)^*2*
⟨proof⟩

lemma *norm-square-gamma-factor′*:
assumes *norm u < 1*
shows (*norm u*)^*2* = ((*γ u*)^*2* − *1*) / (*γ u*)^*2*
⟨proof⟩

lemma *gamma-factor-square-norm*:
assumes *norm u < 1*
shows $(\gamma\ u)^2 = 1\ /\ (1 - (norm\ u)^2)$
⟨proof⟩

lemma *gamma-expression-eq-one-1*:
assumes *norm u < 1*
shows *1* / *γ u* + (*γ u* ∗ (*norm u*)^*2*) / (*1* + *γ u*) = *1*
⟨proof⟩

lemma *gamma-expression-eq-one-2*:
assumes *norm u < 1*
shows ((*γ u*)^*2* ∗ (*norm u*)^*2*) / (*1* + *γ u*)^*2* + (*2* ∗ *γ u*) / (*γ u* ∗ (*1* + *γ u*))
= *1*
⟨proof⟩

end
theory *PoincareDisc*
imports *Complex-Main HOL−Analysis.Inner-Product GammaFactor*
begin

typedef *PoincareDisc* = {*z*::*complex*. *cmod z < 1*}
morphisms *to-complex of-complex*
⟨proof⟩

setup-lifting *type-definition-PoincareDisc*

lemma *poincare-disc-two-elems*:
shows ∃ *z1 z2*::*PoincareDisc*. *z1* ≠ *z2*
⟨proof⟩

lift-definition *inner-p* :: *PoincareDisc* ⇒ *PoincareDisc* ⇒ *real* (**infixl** · *100*) **is** *inner* ⟨*proof*⟩

lift-definition *norm-p* :: *PoincareDisc* ⇒ *real* (⟪-⟫ [*100*] *101*) **is** *norm* ⟨*proof*⟩

lemma *norm-lt-one*:
shows ⟪*u*⟫ < *1*
⟨*proof*⟩

lemma *norm-geq-zero*:
shows ⟪*u*⟫ ≥ *0*
⟨*proof*⟩

lemma *square-norm-inner*:
shows $(⟪u⟫)^2$ = *u* · *u*
⟨*proof*⟩

lift-definition *gammma-factor-p* :: *PoincareDisc* ⇒ *real* (γ_p) **is** *gamma-factor* ⟨*proof*⟩

lemma *gamma-factor-p-nonzero* [*simp*]:
shows γ_p *u* ≠ *0*
⟨*proof*⟩

lemma *gamma-factor-p-positive* [*simp*]:
shows γ_p *u* > *0*
⟨*proof*⟩

lemma *norm-square-gamma-factor-p*:
shows (⟪*u*⟫)^*2* = *1* − *1* / (γ_p *u*)^*2*
⟨*proof*⟩

lemma *norm-square-gamma-factor-p′*:
shows (⟪*u*⟫)^*2* = ((γ_p *u*)^*2* − *1*) / (γ_p *u*)^*2*
⟨*proof*⟩

lemma *gamma-factor-p-square-norm*:
shows $(\gamma_p\ u)^2$ = *1* / (*1* − $(⟪u⟫)^2$)
⟨*proof*⟩

end
theory *MobiusGyroGroup*
imports *Complex-Main HOL.Real-Vector-Spaces HOL.Transcendental MoreComplex*
GyroGroup PoincareDisc
begin

definition *ozero-m′* :: *complex* **where**

ozero-m′ = *0*

lift-definition *ozero-m* :: *PoincareDisc* (0_m) **is** *ozero-m′*
⟨*proof*⟩

lemma *to-complex-0* [*simp*]:
shows *to-complex* 0_m = *0*
⟨*proof*⟩

lemma *to-complex-0-iff* [*iff*]:
shows *to-complex x* = *0* ⟷ *x* = 0_m
⟨*proof*⟩

definition *oplus-m′* :: *complex* ⇒ *complex* ⇒ *complex* **where**
oplus-m′ a z = (*a* + *z*) / (*1* + (*cnj a*) ∗ *z*)

lemma *oplus-m′-in-disc*:
assumes *cmod c1* < *1 cmod c2* < *1*
shows *cmod* (*oplus-m′ c1 c2*) < *1*
⟨*proof*⟩

lift-definition *oplus-m* :: *PoincareDisc* ⇒ *PoincareDisc* ⇒ *PoincareDisc* (**infixl** $\oplus_m$ *100*) **is** *oplus-m′*
⟨*proof*⟩

definition *ominus-m′* :: *complex* ⇒ *complex* **where**
ominus-m′ z = − *z*

lemma *ominus-m′-in-disc*:
assumes *cmod z* < *1*
shows *cmod* (*ominus-m′ z*) < *1*
⟨*proof*⟩

lift-definition *ominus-m* :: *PoincareDisc* ⇒ *PoincareDisc* ($\ominus_m$) **is** *ominus-m′*
⟨*proof*⟩

lemma *m-left-id*:
shows $0_m \oplus_m a = a$
⟨*proof*⟩

lemma *m-left-inv*:
shows $\ominus_m a \oplus_m a = 0_m$
⟨*proof*⟩

definition *gyr-m′* :: *complex* ⇒ *complex* ⇒ *complex* ⇒ *complex* **where**
gyr-m′ a b z = ((*1* + *a* ∗ *cnj b*) / (*1* + *cnj a* ∗ *b*)) ∗ *z*

lift-definition gyr_m :: *PoincareDisc* ⇒ *PoincareDisc* ⇒ *PoincareDisc* ⇒ *PoincareDisc* **is** *gyr-m′*

⟨*proof*⟩

lemma *gyr-m-commute*:
$a \oplus_m b = gyr_m\ a\ b\ (b \oplus_m a)$
⟨*proof*⟩

lemma *gyr-m-left-assoc*:
$a \oplus_m (b \oplus_m z) = (a \oplus_m b) \oplus_m gyr_m\ a\ b\ z$
⟨*proof*⟩

lemma *gyr-m-inv*:
$gyr_m\ a\ b\ (gyr_m\ b\ a\ z) = z$
⟨*proof*⟩

lemma *gyr-m-bij*:
shows *bij* $(gyr_m\ a\ b)$
⟨*proof*⟩

lemma *gyr-m-not-degenerate*:
shows $\exists\ z1\ z2.\ gyr_m\ a\ b\ z1 \neq gyr_m\ a\ b\ z2$
⟨*proof*⟩

lemma *gyr-m-left-loop*:
shows $gyr_m\ a\ b = gyr_m\ (a \oplus_m b)\ b$
⟨*proof*⟩

lemma *gyr-m-distrib*:
shows $gyr_m\ a\ b\ (a' \oplus_m b') = gyr_m\ a\ b\ a' \oplus_m gyr_m\ a\ b\ b'$
⟨*proof*⟩

interpretation *Mobius-gyrogroup*: *gyrogroup ozero-m oplus-m ominus-m* gyr_m
⟨*proof*⟩

interpretation *Mobius-gyrocommutative-gyrogroup*: *gyrocommutative-gyrogroup ozero-m oplus-m ominus-m* gyr_m
⟨*proof*⟩

instantiation *PoincareDisc* :: *gyrogroupoid*
begin
definition *gyrozero-PoincareDisc* **where**
gyrozero-PoincareDisc = *ozero-m*
definition *gyroplus-PoincareDisc* **where**
gyroplus-PoincareDisc = *oplus-m*
instance ⟨*proof*⟩
end

instantiation *PoincareDisc* :: *gyrogroup*
begin
definition *gyroinv-PoincareDisc* **where**

gyroinv-PoincareDisc = *ominus-m*
definition *gyr-PoincareDisc* **where**
gyr-PoincareDisc = gyr_m
instance ⟨*proof*⟩
end

instantiation *PoincareDisc* :: *gyrocommutative-gyrogroup*
begin
instance ⟨*proof*⟩
end

lemma *oplusM-reals*:
assumes *Im* (*to-complex x*) = *0* *Im* (*to-complex y*) = *0*
shows *Im* (*to-complex* ($x \oplus_m y$)) = *0*
⟨*proof*⟩

lemma *oplusM-pos-reals*:
assumes *Im* (*to-complex x*) = *0* *Im* (*to-complex y*) = *0*
assumes *Re* (*to-complex x*) ≥ *0* *Re* (*to-complex y*) ≥ *0*
shows *Re* (*to-complex* ($x \oplus_m y$)) ≥ *0*
⟨*proof*⟩

definition gyr_m*-alternative* :: *PoincareDisc* ⇒ *PoincareDisc* ⇒ *PoincareDisc* ⇒ *PoincareDisc* **where**
gyr_m*-alternative u v w* = $\ominus_m (u \oplus_m v) \oplus_m (u \oplus_m (v \oplus_m w))$

lemma *gyr-m-alternative-gyr-m*:
shows gyr_m*-alternative u v w* = gyr_m *u v w*
⟨*proof*⟩

definition *oplus-m′-alternative* :: *complex* ⇒ *complex* ⇒ *complex* **where**
oplus-m′-alternative u v =
((*1* + *2*∗*inner u v* + (*norm v*)^*2*) $*_R$ *u* + (*1* − (*norm u*)^*2*) $*_R$ *v*) /
(*1* + *2*∗*inner u v* + (*norm u*)^*2* ∗ (*norm v*)^*2*)

lemma *oplus-m′-alternative*:
assumes *cmod u* < *1* *cmod v* < *1*
shows *oplus-m′-alternative u v* = *oplus-m′ u v*
⟨*proof*⟩

lift-definition *oplus-m-alternative* :: *PoincareDisc* ⇒ *PoincareDisc* ⇒ *PoincareDisc* **is** *oplus-m′-alternative*
⟨*proof*⟩

end
theory *Gyrotrigonometry*
imports *Main GyroVectorSpace*

begin

datatype *'a otriangle* = *M-gyrotriangle* (*A*:*'a*) (*B*:*'a*) (*C*:*'a*)

context *pre-gyrovector-space*
begin

definition *unit* :: *'a* ⇒ *'b* **where**
  unit a = *to-carrier a* $/_R$ ⟪*a*⟫

lemma *norm-inner-le-1*:
  fixes *a b* :: *'b*
  assumes *norm a* ≤ *1 norm b* ≤ *1*
  shows *norm* (*inner a b*) ≤ *1*
  ⟨*proof*⟩

lemma *norm-inner-unit*:
  shows *norm* (*inner* (*unit* (⊖ *a* ⊕ *b*)) (*unit* (⊖ *a* ⊕ *c*))) ≤ *1*
⟨*proof*⟩

definition *angle* :: *'a* ⇒ *'a* ⇒ *'a* ⇒ *real* **where**
  angle a b c = *arccos* (*inner* (*unit* (⊖ *a* ⊕ *b*)) (*unit* (⊖ *a* ⊕ *c*)))

definition *o-ray* :: *'a* ⇒ *'a* ⇒ *'a set* **where**
  o-ray x p = {*s*::*'a*. ∃ *t*::*real*. *t* ≥ *0* ∧ *s* = (*x* ⊕ *t* ⊗ (⊖ *x* ⊕ *p*))}

lemma *T8-5*:
  assumes *b2* ∈ *o-ray a1 b1 b2* ≠ *a1*
          c2 ∈ *o-ray a1 c1 c2* ≠ *a1*
  shows *angle a1 b1 c1* = *angle a1 b2 c2*
⟨*proof*⟩

definition *get-a* :: *'a otriangle* ⇒ *'a* **where**
  get-a t = ⊖ (*C t*) ⊕ (*B t*)
definition *get-b* :: *'a otriangle* ⇒ *'a* **where**
  get-b t = ⊖ (*C t*) ⊕ (*A t*)
definition *get-c* :: *'a otriangle* ⇒ *'a* **where**
  get-c t = ⊖ (*B t*) ⊕ (*A t*)

definition *get-alpha* :: *'a otriangle* ⇒ *real* **where**
  get-alpha t = *angle* (*A t*) (*B t*) (*C t*)
definition *get-beta* :: *'a otriangle* ⇒ *real* **where**
  get-beta t = *angle* (*B t*) (*C t*) (*A t*)
definition *get-gamma* :: *'a otriangle* ⇒ *real* **where**
  get-gamma t = *angle* (*C t*) (*A t*) (*B t*)

definition *cong-gyrotriangles* :: *'a otriangle* ⇒ *'a otriangle* ⇒ *bool* **where**
  cong-gyrotriangles t1 t2 ⟷
    (⟪*get-a t1*⟫ = ⟪*get-a t2*⟫ ∧ ⟪*get-b t1*⟫ = ⟪*get-b t2*⟫ ∧ ⟪*get-c t1*⟫ = ⟪*get-c t2*⟫

∧
  (*get-alpha t1* = *get-alpha t2*) ∧ (*get-beta t1* = *get-beta t2*) ∧ (*get-gamma t1* = *get-gamma t2*))
end

end
theory *HyperbolicFunctions*
imports *HOL.Transcendental*
begin

lemma *artanh-abs-tanh*:
  fixes *x::real*
  shows *artanh* (*abs* (*tanh x*)) = *abs x*
⟨*proof*⟩

lemma *artanh-nonneg*:
  fixes *x* :: *real*
  assumes *0* ≤ *x x* < *1*
  shows *artanh x* ≥ *0*
⟨*proof*⟩

lemma *artanh-not-0*:
  fixes *x* :: *real*
  assumes *x* > *0 x* < *1*
  shows *artanh x* ≠ *0*
  ⟨*proof*⟩

lemma *tanh-not-0*:
  fixes *x* :: *real*
  assumes *x* > *0 x* < *1*
  shows *tanh x* ≠ *0*
  ⟨*proof*⟩

lemma *tanh-monotone*:
  fixes *x y* :: *real*
  assumes *x* > *y*
  shows *tanh x* > *tanh y*
  ⟨*proof*⟩

lemma *artanh-monotone1*:
  fixes *x::real*
  assumes *x* ≥ *0 x* < *1 y* ≥ *0 y* < *1 x* ≤ *y*
  shows (*1*+*x*) / (*1*−*x*) ≤ (*1*+*y*) / (*1*−*y*)
  ⟨*proof*⟩

lemma *artanh-monotone2*:
  fixes *x::real*
  assumes *x*≥*0 x*<*1 y*≥*0 y*<*1 x*≤*y*
  shows *ln* ((*1*+*x*)/(*1*−*x*)) ≤ *ln*((*1*+*y*)/(*1*−*y*))

⟨*proof*⟩

lemma *artanh-monotone*:
fixes $x\ y :: real$
assumes $x \geq 0\ x < 1\ 0 \leq y\ y < 1$
assumes $x \leq y$
shows $artanh\ x \leq artanh\ y$
⟨*proof*⟩

lemma *tanh-artanh-nonneg*:
fixes $x\ r :: real$
assumes $r \geq 0\ x \geq 0\ x < 1$
shows $tanh\ (r * artanh\ x) \geq 0$
⟨*proof*⟩

lemma *tanh-artanh-mono*:
fixes $x\ y :: real$
assumes $0 \leq x\ x < 1\ 0 \leq y\ y < 1$
assumes $x \leq y$
shows $tanh\ (2 * artanh\ x) \leq tanh\ (2 * artanh\ y)$
⟨*proof*⟩

lemma *tanh-def'*:
fixes $x :: real$
shows $tanh\ x = (exp\ (2*x) - 1) / (exp\ (2*x) + 1)$
⟨*proof*⟩

lemma *tanh-artanh*:
fixes $x :: real$
assumes $-1 < x\ x < 1$
shows $tanh\ (artanh\ x) = x$
⟨*proof*⟩

end
theory *MobiusGyroVectorSpace*
imports *Main MobiusGyroGroup GyroVectorSpace Gyrotrigonometry GammaFactor HyperbolicFunctions*
begin

lemma *norms*:
shows $\{x.\ \exists a.\ x = cmod\ (to\text{-}complex\ a)\} \cup \{x.\ \exists a.\ x = -\ cmod\ (to\text{-}complex\ a)\} = \{x.\ |x| < 1\}$
⟨*proof*⟩

global-interpretation *Mobius-gyrocarrier'*: *gyrocarrier'*
where $to\text{-}carrier = to\text{-}complex$
rewrites

Mobius-gyrocarrier′.gyroinner = inner-p **and**
Mobius-gyrocarrier′.gyronorm = norm-p **and**
Mobius-gyrocarrier′.carrier = {*z. cmod z* < *1*} **and**
Mobius-gyrocarrier′.norms = {*x. abs x* < *1*}
defines
of-complex = gyrocarrier′.of-carrier to-complex
⟨*proof*⟩

lemma *Mobius-gyrocarrier′-norms* [*simp*]:
shows *gyrocarrier′.norms to-complex* = {*x. abs x* < *1*}
⟨*proof*⟩

lemma *Mobius-gyrocarrier′-carrier* [*simp*]:
shows *gyrocarrier′.carrier to-complex* = {*z. cmod z* < *1*}
⟨*proof*⟩

lemma *moebius-gyroauto*:
shows gyr_m *u v a* · gyr_m *u v b* = *a* · *b*
⟨*proof*⟩

interpretation *Mobius-gyrocarrier*: *gyrocarrier*
where *to-carrier* = *to-complex*
⟨*proof*⟩

global-interpretation *Mobius-gyrocarrier-norms-embed′*: *gyrocarrier-norms-embed′*
where *to-carrier* = *to-complex*
rewrites
Mobius-gyrocarrier-norms-embed′.reals = *of-complex* ‘ *cor* ‘ {*x. abs x* < *1*}
⟨*proof*⟩

lemma *Mobius-gyrocarrier-norms-embed′-to-real′*:
assumes *x* ∈ *Mobius-gyrocarrier-norms-embed′.reals*
shows *Mobius-gyrocarrier-norms-embed′.to-real′ x* = *Re* (*to-complex x*)
⟨*proof*⟩

lemma *Mobius-gyrocarrier-norms-embed′-of-real′*:
assumes *x* ∈ *Mobius-gyrocarrier′.norms*
shows *Mobius-gyrocarrier-norms-embed′.of-real′ x* = *PoincareDisc.of-complex* (*cor x*)
⟨*proof*⟩

lemma *gyronorm-Re*:
assumes *Re* (*to-complex x*) ≥ *0 Im* (*to-complex x*) = *0*
shows ⟪*x*⟫ = *Re* (*to-complex x*)
⟨*proof*⟩

lemma *Mobius-gyrocarrier-norms-embed′-reals* [*simp*]:
  shows *gyrocarrier-norms-embed′.reals to-complex* = *of-complex* ‘ *cor* ‘ {*x*. |*x*| < *1*}
  ⟨*proof*⟩

definition *otimes′-k* :: *real* ⇒ *complex* ⇒ *real* **where**
  otimes′-k r z = ((*1* + *cmod z*) *powr r* − (*1* − *cmod z*) *powr r*) /
                  ((*1* + *cmod z*) *powr r* + (*1* − *cmod z*) *powr r*)

lemma *otimes′-k-tanh*:
  assumes *cmod z* < *1*
  shows *otimes′-k r z* = *tanh* (*r* ∗ *artanh* (*cmod z*))
⟨*proof*⟩

lemma *cmod-otimes′-k*:
  assumes *cmod z* < *1*
  shows *cmod* (*otimes′-k r z*) < *1*
  ⟨*proof*⟩

definition *otimes′* :: *real* ⇒ *complex* ⇒ *complex* **where**
  otimes′ r z = (*if z* = *0 then 0 else cor* (*otimes′-k r z*) ∗ (*z* / *cmod z*))

lemma *cmod-otimes′*:
  assumes *cmod z* < *1*
  shows *cmod* (*otimes′ r z*) = *abs* (*otimes′-k r z*)
⟨*proof*⟩

lift-definition *otimes* :: *real* ⇒ *PoincareDisc* ⇒ *PoincareDisc* (**infixl** ⊗ *105*) **is** *otimes′*
  ⟨*proof*⟩

lemma *otimes-distrib-lemma′*:
  fixes *ax bx ay by* :: *real*
  assumes *ax* + *bx* ≠ *0 ay* + *by* ≠ *0*
  shows (*ax* ∗ *ay* − *bx* ∗ *by*) / (*ax* ∗ *ay* + *bx* ∗ *by*) =
         ((*ax* − *bx*)/(*ax* + *bx*) + (*ay* − *by*)/(*ay* + *by*)) /
          (*1* + ((*ax* − *bx*)/(*ax* + *bx*))∗((*ay* − *by*)/(*ay* + *by*))) (**is** *?lhs* = *?rhs*)
⟨*proof*⟩

lemma *otimes-distrib-lemma*:
  assumes *cmod a* < *1*
  shows *otimes′-k* (*r1* + *r2*) *a* = *oplus-m′* (*otimes′-k r1 a*) (*otimes′-k r2 a*)
  ⟨*proof*⟩

lemma *otimes-oplus-m-distrib*:
  shows (*r1* + *r2*) ⊗ *a* = *r1* ⊗ *a* $\oplus_m$ *r2* ⊗ *a*
⟨*proof*⟩

lemma *otimes-assoc*:
  shows $(r1 * r2) \otimes a = r1 \otimes (r2 \otimes a)$
⟨*proof*⟩

lemma *otimes-scale-prop*:
  fixes r :: *real*
  assumes $r \neq 0$
  shows *to-complex* $(|r| \otimes a)$ / $\langle\langle r \otimes a\rangle\rangle$ = *to-complex* a / $\langle\langle a\rangle\rangle$
⟨*proof*⟩

lemma *gamma-factor-eq1-lemma1*:
  shows $cmod(1 + cnj\ a * b)*cmod(1 + cnj\ a * b) - cmod(a+b)*cmod(a+b) =$
  $(1 - cmod\ a * cmod\ a) * (1 - cmod\ b * cmod\ b)$
⟨*proof*⟩

lemma *gamma-factor-eq1-lemma2*:
  fixes x y::*real*
  assumes $y > 0$
  shows $1 / sqrt(1 - (x*x)/(y*y)) = abs\ y / sqrt(y*y - x*x)$
⟨*proof*⟩

lemma *gamma-factor-norm-oplus-m*:
  shows γ $(\langle\langle a \oplus_m b\rangle\rangle) =$
  γ (*to-complex* a) $*$
  γ (*to-complex* b) $*$
  cmod (1 + *cnj* (*to-complex* a) $*$ (*to-complex* b))
⟨*proof*⟩

lemma *gamma-factor-norm-oplus-m′*:
  shows γ_p (*of-complex* (*cor* $(\langle\langle a \oplus_m b\rangle\rangle)$)) =
  γ_p (a) $*$
  γ_p (b) $*$
  cmod (1 + *cnj* (*to-complex* a) $*$ (*to-complex* b))
⟨*proof*⟩

lemma *gamma-factor-oplus-m-triangle-lemma*:
  fixes x y ::*real*
  assumes $x \geq 0$ $x < 1$ $y \geq 0$ $y < 1$
  shows $1 / sqrt\ (1 - ((x+y)*(x+y))/((1+x*y)*(1+x*y))) =$
  $(1+x*y) / (sqrt\ (1-x*x) * sqrt\ (1-y*y))$
⟨*proof*⟩

lemma *gamma-factor-oplus-m-triangle*:
  shows γ $(\langle\langle a \oplus_m b\rangle\rangle) \leq \gamma$ (*to-complex* ((*of-complex* $(\langle\langle a\rangle\rangle)$) $\oplus_m$ (*of-complex* $(\langle\langle b\rangle\rangle)$)))
⟨*proof*⟩

lemma *mobius-triangle*:
shows $\langle\langle a \oplus_m b\rangle\rangle \leq \langle\langle$*of-complex* $(\langle\langle a\rangle\rangle) \oplus_m$ *of-complex* $(\langle\langle b\rangle\rangle)\rangle\rangle$
⟨*proof*⟩

lemma *mobius-triangle'*:
shows $\langle\langle a \oplus_m b\rangle\rangle \leq$ *Re* (*to-complex* (*of-complex* $(\langle\langle a\rangle\rangle) \oplus_m$ *of-complex* $(\langle\langle b\rangle\rangle)$))
⟨*proof*⟩

lemma *mobius-gyroauto-norm*:
shows $\langle\langle gyr_m\ a\ b\ v\rangle\rangle = \langle\langle v\rangle\rangle$
⟨*proof*⟩

lemma *otimes-homogenity*:
shows $\langle\langle r \otimes a\rangle\rangle =$ *cmod* (*to-complex* ($|r| \otimes$ *of-complex* $(\langle\langle a\rangle\rangle)$))
⟨*proof*⟩

lemma *otimes-homogenity'*:
shows $\langle\langle r \otimes a\rangle\rangle =$ *Re* (*to-complex* ($|r| \otimes$ *of-complex* $(\langle\langle a\rangle\rangle)$))
⟨*proof*⟩

lemma *gyr-m-gyrospace*:
shows $gyr_m\ (r1 \otimes v)\ (r2 \otimes v) = id$
⟨*proof*⟩

lemma *gyr-m-gyrospace2*:
shows $gyr_m\ u\ v\ (r \otimes a) = r \otimes (gyr_m\ u\ v\ a)$
⟨*proof*⟩

lemma *reals'*:
shows *cor* ‘ $\{x.\ abs\ x < 1\} = \{z.\ cmod\ z < 1 \wedge Im\ z = 0\}$
⟨*proof*⟩

lemma *zero-times-m* [*simp*]:
shows $0 \otimes x = 0_m$
⟨*proof*⟩

interpretation *Mobius-gyrocarrier-norms-embed*: *gyrocarrier-norms-embed to-complex otimes*
⟨*proof*⟩

interpretation *Mobius-pre-gyrovector-space*: *pre-gyrovector-space to-complex otimes*
⟨*proof*⟩

interpretation *Mobius-gyrovector-space*: *gyrovector-space-norms-embed otimes to-complex*

⟨*proof*⟩

lemma *norm-scale-tanh*:
  shows $\langle\!\langle r \otimes z \rangle\!\rangle = |tanh\ (r * artanh\ (\langle\!\langle z \rangle\!\rangle))|$
⟨*proof*⟩

lemma *ominus-m-scale*:
  shows $k \otimes (\ominus_m\ u) = \ominus_m\ (k \otimes u)$
  ⟨*proof*⟩

lemma *otimes-2-oplus-m*: $2 \otimes u = u \oplus_m u$
  ⟨*proof*⟩

definition *half'* :: *complex* ⇒ *complex* **where**
  $half'\ v = (\gamma\ v\ /\ (1 + \gamma\ v)) *_R v$

lift-definition *half* :: *PoincareDisc* ⇒ *PoincareDisc* **is** *half'*
  ⟨*proof*⟩

lemma *otimes-2-half*:
  shows $2 \otimes (half\ v) = v$
⟨*proof*⟩

lemma *half*:
  shows $half\ v = (1/2) \otimes v$
  ⟨*proof*⟩

lemma *half'*:
  assumes $cmod\ u < 1$
  shows $otimes'\ (1/2)\ u = half'\ u$
  ⟨*proof*⟩

lemma *half-gamma'*:
  shows $to\text{-}complex\ ((1\ /\ 2) \otimes u) =$
    $(\gamma\ (to\text{-}complex\ u))\ /\ (1 + \gamma\ (to\text{-}complex\ u)) * to\text{-}complex\ u$
  ⟨*proof*⟩

definition *double'* :: *complex* ⇒ *complex* **where**
  $double'\ v = (2 * (\gamma\ v)^2\ /\ (2 * (\gamma\ v)^2 - 1)) *_R v$

lemma *double'-cmod*:
  assumes $cmod\ v < 1$
  shows $2 * (\gamma\ v)^2\ /\ (2 * (\gamma\ v)^2 - 1) = 2\ /\ (1 + (cmod\ v)^2)$ (**is** *?lhs* = *?rhs*)
⟨*proof*⟩

lemma *cmod-double'*:
  assumes $cmod\ v < 1$
  shows $cmod\ (double'\ v) = 2{*}cmod\ v\ /\ (1 + (cmod\ v)^2)$
⟨*proof*⟩

lift-definition *double* :: *PoincareDisc* ⇒ *PoincareDisc* **is** *double′*
⟨*proof*⟩

lemma *double′-otimes′-2*:
  assumes *cmod v < 1*
  shows *double′ v = otimes′ 2 v*
⟨*proof*⟩

lemma *double*:
  shows *double u = 2 ⊗ u*
  ⟨*proof*⟩

end
theory *Einstein*
  imports *Complex-Main GyroGroup GyroVectorSpace GyroVectorSpaceIsomorphism GammaFactor HOL.Real-Vector-Spaces*
  MobiusGyroGroup MobiusGyroVectorSpace HOL.Transcendental
begin

Einstein zero

definition *ozero-e′* :: *complex* **where**
  ozero-e′ = 0

lift-definition *ozero-e* :: *PoincareDisc* (0_e) **is** *ozero-e′*
  ⟨*proof*⟩

lemma *ozero-e-ozero-m*:
  shows $0_e = 0_m$
  ⟨*proof*⟩

Einstein addition

definition *oplus-e′* :: *complex* ⇒ *complex* ⇒ *complex* **where**
  oplus-e′ u v = (*1* / (*1* + *inner u v*)) $*_R$ (*u* + (*1* / γ *u*) $*_R$ *v* + ((γ *u* / (*1* + γ *u*)) ∗ (*inner u v*)) $*_R$ *u*)

lemma *noroplus-m′-e*:
  assumes *norm u < 1 norm v <1*
  shows *norm* (*oplus-e′ u v*)^*2* =
      1 / (*1* + *inner u v*)^*2* ∗ (*norm*(*u*+*v*)^*2* − ((*norm u*)^*2* ∗(*norm v*)^*2* − (*inner u v*)^*2*))
⟨*proof*⟩

lemma *gamma-oplus-e′*:
  assumes *norm u < 1 norm v < 1*
  shows *1* / *sqrt*(*1* − *norm* (*oplus-e′ u v*)^*2*) = γ *u* ∗ γ *v* ∗ (*1* + *inner u v*)
⟨*proof*⟩

lemma *gamma-oplus-e′-not-zero*:
  assumes *norm u < 1 norm v < 1*
  shows *1 / sqrt(1 − norm(oplus-e′ u v)^2) ≠ 0*
  ⟨*proof*⟩

lemma *oplus-e′-in-unit-disc*:
  assumes *norm u < 1 norm v < 1*
  shows *norm (oplus-e′ u v) < 1*
⟨*proof*⟩

lemma *gamma-factor-oplus-e′*:
  assumes *norm u < 1 norm v < 1*
  shows *γ (oplus-e′ u v) = (γ u) ∗ (γ v) ∗ (1 + inner u v)*
⟨*proof*⟩

lift-definition *oplus-e* :: *PoincareDisc ⇒ PoincareDisc ⇒ PoincareDisc* (**infixl** $\oplus_e$ *100*) **is** *oplus-e′*
  ⟨*proof*⟩

definition *ominus-e′* :: *complex ⇒ complex* **where**
  ominus-e′ v = − v

lemma *ominus-e′-in-unit-disc*:
  assumes *norm z < 1*
  shows *norm (ominus-e′ z) < 1*
  ⟨*proof*⟩

lift-definition *ominus-e* :: *PoincareDisc ⇒ PoincareDisc* ($\ominus_e$) **is** *ominus-e′*
  ⟨*proof*⟩

lemma *ominus-e-ominus-m*:
  shows $\ominus_e$ *a* = $\ominus_m$ *a*
  ⟨*proof*⟩

lemma *ominus-e-scale*:
  shows *k ⊗* ($\ominus_e$ *u*) = $\ominus_e$ (*k ⊗ u*)
  ⟨*proof*⟩

lemma *gamma-factor-p-positive*:
  shows γ_p *a > 0*
  ⟨*proof*⟩

lemma *gamma-factor-p-oplus-e*:
  shows γ_p (*u* $\oplus_e$ *v*) = γ_p *u* ∗ γ_p *v* ∗ (*1 + u · v*)
  ⟨*proof*⟩

abbreviation γ_2 :: *complex* $\Rightarrow$ *real* **where**
$\gamma_2\ u \equiv \gamma\ u\ /\ (1 + \gamma\ u)$

lemma *norm-square-gamma-half-scale*:
assumes *norm u* < 1
shows $(norm\ (\gamma_2\ u *_R u))^2 = (\gamma\ u - 1)\ /\ (1 + \gamma\ u)$
⟨*proof*⟩

lemma *norm-half-square-gamma*:
assumes *norm u* < 1
shows $(norm\ (half'\ u))^2 = (\gamma_2\ u)^2 * (cmod\ u)^2$
⟨*proof*⟩

lemma *norm-half-square-gamma'*:
assumes *cmod u* < 1
shows $(norm\ (half'\ u))^2 = (\gamma\ u - 1)\ /\ (1 + \gamma\ u)$
⟨*proof*⟩

lemma *inner-half-square-gamma*:
assumes *cmod u* < 1 *cmod v* < 1
shows $inner\ (half'\ u)\ (half'\ v) = \gamma_2\ u * \gamma_2\ v * inner\ u\ v$
⟨*proof*⟩

lemma *iso-me-help1*:
assumes *norm v* < 1
shows $1 + (\gamma\ v - 1)\ /\ (1 + \gamma\ v) = 2 * \gamma\ v\ /\ (1 + \gamma\ v)$
⟨*proof*⟩

lemma *iso-me-help2*:
assumes *norm v* < 1
shows $1 - (\gamma\ v - 1)\ /\ (1 + \gamma\ v) = 2\ /\ (1 + \gamma\ v)$
⟨*proof*⟩

lemma *iso-me-help3*:
assumes *norm v* < 1 *norm u* <1
shows $1 + ((\gamma\ v - 1)\ /\ (1 + \gamma\ v)) * ((\gamma\ u - 1)\ /\ (1 + \gamma\ u)) =$
$2 * (1 + (\gamma\ u) * (\gamma\ v))\ /\ ((1 + \gamma\ v) * (1 + \gamma\ u))$ (**is** *?lhs* = *?rhs*)
⟨*proof*⟩

lemma *half'-oplus-e'*:
fixes *u v* :: *complex*
assumes *cmod u* < 1 *cmod v* < 1
shows *half'* (*oplus-e' u v*) =
$\gamma\ u * \gamma\ v\ /\ (\gamma\ u * \gamma\ v * (1 + inner\ u\ v) + 1) * (u + (1\ /\ \gamma\ u) * v + (\gamma\ u\ /\ (1 + \gamma\ u)) * inner\ u\ v * u)$
⟨*proof*⟩

lemma *oplus-m'-half'*:

fixes $u\ v$ :: *complex*
assumes $cmod\ u < 1$ $cmod\ v < 1$
shows $oplus\text{-}m'\ (half'\ u)\ (half'\ v) =$
$(\gamma\ u * \gamma\ v\ /\ (\gamma\ u * \gamma\ v * (1 + inner\ u\ v) + 1)) *$
$(u + (1\ /\ \gamma\ u) * v + (\gamma\ u\ /\ (1 + \gamma\ u) * inner\ u\ v) * u)$
⟨*proof*⟩

lemma *iso-me-oplus*:
shows $(1/2) \otimes (u \oplus_e v) = ((1/2) \otimes u) \oplus_m ((1/2) \otimes v)$
⟨*proof*⟩

lemma *oplus-e-oplus-m*:
shows $u \oplus_e v = 2 \otimes ((1/2) \otimes u \oplus_m (1/2) \otimes v)$
⟨*proof*⟩

lemma *iso-two-me-oplus*:
shows $2 \otimes (u \oplus_m v) = (2 \otimes u) \oplus_e (2 \otimes v)$
⟨*proof*⟩

lemma *iso-two-me-ominus*:
shows $2 \otimes (\ominus_m u) = \ominus_e (2 \otimes u)$
⟨*proof*⟩

lemma *iso-two-me-zero*:
shows $2 \otimes 0_m = 0_e$
⟨*proof*⟩

lemma *iso-two-me-bij*:
shows $bij\ (\lambda\ x{::}PoincareDisc.\ 2 \otimes x)$
⟨*proof*⟩

definition gyr_e::*PoincareDisc* ⇒ *PoincareDisc* ⇒ *PoincareDisc* ⇒ *PoincareDisc*
where
$gyr_e\ u\ v\ w = \ominus_e (u \oplus_e v) \oplus_e (u \oplus_e (v \oplus_e w))$

typedef *PoincareDiscM* = *UNIV*::*PoincareDisc set*
⟨*proof*⟩
setup-lifting *type-definition-PoincareDiscM*

lift-definition *zero-M* :: *PoincareDiscM* (0_M) **is** 0_m ⟨*proof*⟩

lift-definition *ominus-M* :: *PoincareDiscM* ⇒ *PoincareDiscM* $(\ominus_M)$ **is** $(\ominus_m)$
⟨*proof*⟩

lift-definition *oplus-M* :: *PoincareDiscM* ⇒ *PoincareDiscM* ⇒ *PoincareDiscM*
(**infixl** $\oplus_M$ *100*) **is** $(\oplus_m)$ ⟨*proof*⟩

lift-definition *gyr-M* :: *PoincareDiscM* ⇒ *PoincareDiscM* ⇒ *PoincareDiscM* ⇒ *PoincareDiscM* **is** gyr_m ⟨*proof*⟩

lift-definition *to-complex-M* :: *PoincareDiscM* ⇒ *complex* **is** *to-complex* ⟨*proof*⟩

interpretation *gyrogroupoid-M*: *gyrogroupoid zero-M oplus-M* ⟨*proof*⟩

instantiation *PoincareDiscM* :: *gyrogroupoid*
begin
definition *gyrozero-PoincareDiscM* **where** *gyrozero-PoincareDiscM* = 0_M
definition *gyroplus-PoincareDiscM* **where** *gyroplus-PoincareDiscM* = *oplus-M*
instance
⟨*proof*⟩
end

instantiation *PoincareDiscM* :: *gyrocommutative-gyrogroup*
begin
definition *gyroinv-PoincareDiscM* **where** *gyroinv-PoincareDiscM* = *ominus-M*
definition *gyr-PoincareDiscM* **where** *gyr-PoincareDiscM* = *gyr-M*
instance ⟨*proof*⟩
end

typedef *PoincareDiscE* = *UNIV*::*PoincareDisc set*
⟨*proof*⟩
setup-lifting *type-definition-PoincareDiscE*

lift-definition *zero-E* :: *PoincareDiscE* (0_E) **is** 0_e ⟨*proof*⟩

lift-definition *ominus-E* :: *PoincareDiscE* ⇒ *PoincareDiscE* ($\ominus_E$) **is** ($\ominus_e$) ⟨*proof*⟩

lift-definition *oplus-E* :: *PoincareDiscE* ⇒ *PoincareDiscE* ⇒ *PoincareDiscE* (**infixl** $\oplus_E$ *100*) **is** ($\oplus_e$) ⟨*proof*⟩

lift-definition *gyr-E* :: *PoincareDiscE* ⇒ *PoincareDiscE* ⇒ *PoincareDiscE* ⇒ *PoincareDiscE* **is** gyr_e ⟨*proof*⟩

lift-definition *to-complex-E* :: *PoincareDiscE* ⇒ *complex* **is** *to-complex* ⟨*proof*⟩

lift-definition φ_{ME} :: *PoincareDiscM* ⇒ *PoincareDiscE* **is** λ *x*::*PoincareDisc*. *2* ⊗ *x* ⟨*proof*⟩

interpretation *Einstein-gyrogroup-iso*:
gyrogroup-isomorphism φ_{ME} *zero-E oplus-E ominus-E*
rewrites
Einstein-gyrogroup-iso.gyr′ = *gyr-E*
⟨*proof*⟩

instantiation *PoincareDiscE* :: *gyrogroupoid*

begin
definition *gyrozero-PoincareDiscE* **where** *gyrozero-PoincareDiscE* = 0_E
definition *gyroplus-PoincareDiscE* **where** *gyroplus-PoincareDiscE* = *oplus-E*
instance
⟨*proof*⟩
end

instantiation *PoincareDiscE* :: *gyrocommutative-gyrogroup*
begin
definition *gyroinv-PoincareDiscE* **where** *gyroinv-PoincareDiscE* = *ominus-E*
definition *gyr-PoincareDiscE* **where** *gyr-PoincareDiscE* = *gyr-E*
instance ⟨*proof*⟩

end

lift-definition *scale-M* :: *real* ⇒ *PoincareDiscM* ⇒ *PoincareDiscM* **is** (⊗) ⟨*proof*⟩

lift-definition *scale-E* :: *real* ⇒ *PoincareDiscE* ⇒ *PoincareDiscE* **is** (⊗) ⟨*proof*⟩

lemma *gyrocarrier′M*:
shows *gyrocarrier′ to-complex-M*
⟨*proof*⟩

lemma *gyrocarrier-norms-embed′M*:
shows *gyrocarrier-norms-embed′ to-complex-M*
⟨*proof*⟩

lemma *of-carrier-M*:
assumes *cmod z* < *1*
shows *gyrocarrier′.of-carrier to-complex-M z* = *Abs-PoincareDiscM* (*PoincareDisc.of-complex* *z*)
⟨*proof*⟩

global-interpretation *GCM*: *gyrocarrier-norms-embed′ to-complex-M*
rewrites *GCM.norms* = {*x*. *abs x* < *1*} **and**
GCM.reals = *Abs-PoincareDiscM* ‘ *PoincareDisc.of-complex* ‘ *cor* ‘ {*x*. |*x*| < *1*}
defines *of-complex-M* = *gyrocarrier′.of-carrier to-complex-M*
⟨*proof*⟩

lemma *of-real′-M*:
assumes *abs x* < *1*
shows *GCM.of-real′ x* = *Abs-PoincareDiscM* (*PoincareDisc.of-complex* (*cor x*))
⟨*proof*⟩

lemma *to-real′-M*:
assumes *z* ∈ *GCM.reals*
shows *GCM.to-real′ z* = *Re* (*to-complex-M z*)

⟨*proof*⟩

lemma *gyronorm-M-lt-1* [*simp*]:
shows *abs* (*GCM.gyronorm a*) < *1*
⟨*proof*⟩

lemma *gyrocarrier′-norms-M* [*simp*]:
shows *gyrocarrier′.norms to-complex-M* = *GCM.norms*
⟨*proof*⟩

lemma *gyrocarrier-norms-embed′-reals-M* [*simp*]:
shows *gyrocarrier-norms-embed′.reals to-complex-M* = *GCM.reals*
⟨*proof*⟩

lemma *gyrocarrier′E*:
shows *gyrocarrier′ to-complex-E*
⟨*proof*⟩

lemma *gyrocarrier-norms-embed′E*:
shows *gyrocarrier-norms-embed′ to-complex-E*
⟨*proof*⟩

lemma *of-carrier-E*:
assumes *cmod z* < *1*
shows *gyrocarrier′.of-carrier to-complex-E z* = *Abs-PoincareDiscE* (*PoincareDisc.of-complex z*)
⟨*proof*⟩

global-interpretation *GCE*: *gyrocarrier-norms-embed′ to-complex-E*
rewrites *GCE.norms* = {*x. abs x* < *1*} **and**
GCE.reals = *Abs-PoincareDiscE* ‘ *PoincareDisc.of-complex* ‘ *cor* ‘ {*x.* |*x*| < *1*}
defines *of-complex-E* = *gyrocarrier′.of-carrier to-complex-E*
⟨*proof*⟩

lemma *of-real′-E*:
assumes *abs x* < *1*
shows *GCE.of-real′ x* = *Abs-PoincareDiscE* (*PoincareDisc.of-complex* (*cor x*))
⟨*proof*⟩

lemma *to-real′-E*:
assumes *z* ∈ *GCE.reals*
shows *GCE.to-real′ z* = *Re* (*to-complex-E z*)
⟨*proof*⟩

lemma *gyronorm-E-lt-1* [*simp*]:
shows *abs* (*GCE.gyronorm a*) < *1*
⟨*proof*⟩

lemma *gyrocarrier′-norms-E* [*simp*]:
shows *gyrocarrier′.norms to-complex-E* = *GCE.norms*
⟨*proof*⟩

lemma *gyrocarrier-norms-embed′-reals-E* [*simp*]:
shows *gyrocarrier-norms-embed′.reals to-complex-E* = *GCE.reals*
⟨*proof*⟩

lemma *φreals-to-reals*:
shows φ_{ME} ‘ *gyrocarrier-norms-embed′.reals to-complex-M* = *gyrocarrier-norms-embed′.reals to-complex-E*
⟨*proof*⟩

interpretation *gyrocarrier-norms-embed-M*: *gyrocarrier-norms-embed to-complex-M scale-M*
⟨*proof*⟩

interpretation *pre-gyrovector-space-M*: *pre-gyrovector-space to-complex-M scale-M*
⟨*proof*⟩

interpretation *gyrovector-space-norms-embed-M*: *gyrovector-space-norms-embed scale-M to-complex-M*
⟨*proof*⟩

lemmas *bij*φ_{ME} = *Einstein-gyrogroup-iso.φbij*

lemma *oplus*φ_{ME}:
shows φ_{ME} $(u \oplus v) = \varphi_{ME}\ u \oplus \varphi_{ME}\ v$
⟨*proof*⟩

lemma *scale*φ_{ME}:
shows φ_{ME} (*scale-M r u*) = *scale-E r* (φ_{ME} *u*)
⟨*proof*⟩

lemma *GCEoinvRMinus*:
assumes $a \in$ *gyrocarrier′.norms to-complex-E*
shows *GCE.oinvR a* = − *a*
⟨*proof*⟩

lemma *gyronorm*φ_{ME}:
shows φ_{ME} (*GCM.of-real′* (*GCM.gyronorm a*)) = *GCE.of-real′* (*GCE.gyronorm* (φ_{ME} *a*))
⟨*proof*⟩

interpretation *isoME″*: *gyrocarrier-isomorphism′ to-complex-M to-complex-E* φ_{ME}
⟨*proof*⟩

interpretation *isoME′*: *gyrocarrier-isomorphism to-complex-M to-complex-E* φ_{ME}
⟨*proof*⟩

interpretation *PGVME*: *pre-gyrovector-space-isomorphism to-complex-M to-complex-E scale-M scale-E* φ_{ME}
⟨*proof*⟩

interpretation *isoME-norms-embed'*: *gyrocarrier-isomorphism-norms-embed' to-complex-M to-complex-E scale-M scale-E* φ_{ME}
⟨*proof*⟩

interpretation *isoME-norms-embed*: *gyrocarrier-isomorphism-norms-embed to-complex-M to-complex-E scale-M scale-E* φ_{ME}
⟨*proof*⟩

interpretation *isoME'*: *gyrovector-space-isomorphism' to-complex-M to-complex-E scale-M scale-E* φ_{ME}
⟨*proof*⟩

interpretation *isoME*: *gyrovector-space-isomorphism to-complex-M to-complex-E scale-M scale-E* φ_{ME}
⟨*proof*⟩

end
theory *GyroVectorSpaceTrivial*
imports *GyroVectorSpace*
begin

Every group is a gyrogroup with identity gyration

sublocale *group-add* ⊆ *groupGyrogroupoid*: *gyrogroupoid 0* (+)
⟨*proof*⟩

sublocale *group-add* ⊆ *groupGyrogroup*: *gyrogroup 0* (+) λ *x*. − *x* λ *u v x*. *x*
⟨*proof*⟩

locale *gyrocarrier-trivial* = *gyrocarrier' to-carrier* **for**
to-carrier :: *'a*::{*gyrocommutative-gyrogroup*, *real-inner*, *real-normed-algebra-1*} ⇒ *'a* +
assumes *gyr-id*: ⋀ *u v x*. (*gyr*::*'a* ⇒ *'a* ⇒ *'a* ⇒ *'a*) *u v x* = *x*
assumes *to-carrier-id*: ⋀ *x*. *to-carrier x* = *x*
assumes *oplus*: ⋀ *x y*::*'a* . *x* ⊕ *y* = *x* + *y*
assumes *ominus*: ⋀ *x*::*'a* . ⊖ *x* = − *x*

sublocale *gyrocarrier-trivial* ⊆ *gyrocarrier to-carrier*
⟨*proof*⟩

sublocale *gyrocarrier-trivial* ⊆ *pre-gyrovector-space to-carrier* ($*_R$)
⟨*proof*⟩

sublocale *gyrocarrier-trivial* ⊆ *TG′*: *gyrocarrier-norms-embed′ to-carrier*
⟨*proof*⟩

context *gyrocarrier-trivial*
begin

lemma *norms-UNIV*:
  shows *norms* = *UNIV*
  ⟨*proof*⟩

lemma *reals-UNIV*:
  shows *TG′.reals* = *of-real* ' *UNIV*
  ⟨*proof*⟩

lemma *of-real′*:
  shows *TG′.of-real′* = *of-real*
  ⟨*proof*⟩

end

sublocale *gyrocarrier-trivial* ⊆ *TG*: *gyrocarrier-norms-embed to-carrier* $(*_R)$
⟨*proof*⟩

sublocale *gyrocarrier-trivial* ⊆ *gyrovector-space-norms-embed* $(*_R)$ *to-carrier*
⟨*proof*⟩

end
theory *hDistance*
  imports *MobiusGyroVectorSpace*
begin

abbreviation *distance-m-expr* :: *complex* ⇒ *complex* ⇒ *real* **where**
  distance-m-expr u v ≡ *1* + *2* ∗ (*cmod* (*u* − *v*))^2 / ((*1* − (*cmod u*)^2) ∗ (*1* − (*cmod v*)^2))

definition *distance-m* :: *complex* ⇒ *complex* ⇒ *real* **where**
  distance-m u v = *arcosh* (*distance-m-expr u v*)

lemma *arcosh-artanh-lemma*:
  shows (*cmod* (*1* − *cnj u* ∗ *v*))^2 − (*cmod* (*u* − *v*))^2 = (*1* − (*cmod u*)^2) ∗ (*1* − (*cmod v*)^2)
⟨*proof*⟩

lemma *distance-m-expr-ge-1*:
  fixes *u v* :: *complex*
  assumes *cmod u* < *1 cmod v* < *1*
  shows *distance-m-expr u v* ≥ *1*
⟨*proof*⟩

lemma *arcosh-artanh*:
  fixes *u v* :: *complex*
  assumes *cmod u <1 cmod v < 1*
  shows *arcosh* (*distance-m-expr u v*) =
        2 ∗ *artanh* (*cmod* ((*u*−*v*) / (*1* − (*cnj u*)∗*v*)))
⟨*proof*⟩

definition *distance-m-gyro* :: *PoincareDisc* ⇒ *PoincareDisc* ⇒ *real* **where**
 distance-m-gyro u v = *2* ∗ *artanh* (*Mobius-pre-gyrovector-space.distance u v*)

lemma *distance-equiv*:
  shows *distance-m-gyro u v* = *distance-m* (*to-complex u*) (*to-complex v*)
⟨*proof*⟩

definition *blaschke* **where**
  blaschke a z = (*z* − *a*) / (*1* − *cnj a* ∗ *z*)

lemma
  fixes *a z* :: *complex*
  shows *blaschke a z* = *oplus-m′* (*ominus-m′ a*) *z*
  ⟨*proof*⟩

end
theory *MobiusCollinear*
  imports *MobiusGyroVectorSpace*
begin

lemma *collinear-0-proportional′*:
  assumes $v \neq 0_m$
  shows *Mobius-pre-gyrovector-space.collinear x* 0_m *v* ⟷ (∃ *k*::*real*. *to-complex x* = *k* ∗ (*to-complex v*))
  ⟨*proof*⟩

lemma
  assumes $v \neq 0_m$
   shows *Mobius-pre-gyrovector-space.collinear x* 0_m *v* ⟷ *to-complex x* ∗ *cnj* (*to-complex v*) = *cnj* (*to-complex x*) ∗ *to-complex v*
  ⟨*proof*⟩

lemma *collinear-0-proportional*:
  shows *Mobius-pre-gyrovector-space.collinear x* 0_m *v* ⟷ *v* = 0_m ∨ (∃ *k*::*real*. *to-complex x* = *k* ∗ (*to-complex v*))
  ⟨*proof*⟩

lemma *to-complex-0* [*simp*]:
  shows *to-complex* 0_m = *0*
  ⟨*proof*⟩

lemma *to-complex-0-iff* [*iff*]:

shows *to-complex* $x = 0 \longleftrightarrow x = 0_m$
⟨*proof*⟩

lemma *mobius-between-0xy*:
shows *Mobius-pre-gyrovector-space.between* 0_m *x y* $\longleftrightarrow$
($\exists$ *k::real.* $0 \leq k \wedge k \leq 1 \wedge$ *to-complex x* $= k *$ *to-complex y*)
⟨*proof*⟩

end
theory *MobiusGeometry*
imports *MobiusGyroVectorSpace*
begin

lemma *mobius-collinear-u0v'*:
assumes $v \neq 0_m$
shows *Mobius-pre-gyrovector-space.collinear u* 0_m *v* $\longleftrightarrow$ ($\exists$ *k::real. to-complex u* $= k *$ (*to-complex v*))
⟨*proof*⟩

lemma *mobius-collinear-u0v*:
shows *Mobius-pre-gyrovector-space.collinear x* 0_m *v* $\longleftrightarrow$
$v = 0_m \vee$ ($\exists$ *k::real. to-complex x* $= k *$ (*to-complex v*))
⟨*proof*⟩

lemma *mobius-between-0uv*:
shows *Mobius-pre-gyrovector-space.between* 0_m *u v* $\longleftrightarrow$
($\exists$ *k::real.* $0 \leq k \wedge k \leq 1 \wedge$ *to-complex u* $= k *$ *to-complex v*)
⟨*proof*⟩

abbreviation *distance-m-expr* :: *complex* $\Rightarrow$ *complex* $\Rightarrow$ *real* **where**
distance-m-expr u v $\equiv 1 + 2 * (cmod\ (u - v))^2 / ((1 - (cmod\ u)^2) * (1 - (cmod\ v)^2))$

definition *distance-m* :: *complex* $\Rightarrow$ *complex* $\Rightarrow$ *real* **where**
distance-m u v = *arcosh* (*distance-m-expr u v*)

lemma *arcosh-artanh-lemma*:
shows $(cmod\ (1 - cnj\ u * v))^2 - (cmod\ (u - v))^2 = (1 - (cmod\ u)^2) * (1 - (cmod\ v)^2)$
⟨*proof*⟩

lemma *distance-m-expr-ge-1*:
fixes *u v* :: *complex*
assumes *cmod u* < 1 *cmod v* < 1
shows *distance-m-expr u v* ≥ 1
⟨*proof*⟩

lemma *arcosh-artanh*:
  fixes *u v* :: *complex*
  assumes *cmod u* <*1 cmod v* < *1*
  shows *arcosh* (*distance-m-expr u v*) =
        2 ∗ *artanh* (*cmod* ((*u*−*v*) / (*1* − (*cnj u*)∗*v*)))
⟨*proof*⟩

definition $distance_m$ :: *PoincareDisc* ⇒ *PoincareDisc* ⇒ *real* **where**
 $distance_m$ *u v* = *2* ∗ *artanh* (*Mobius-pre-gyrovector-space.distance u v*)

lemma $distance_m$*-equiv*:
  shows $distance_m$ *u v* = *distance-m* (*to-complex u*) (*to-complex v*)
⟨*proof*⟩

definition $cong_m$ :: *PoincareDisc* ⇒ *PoincareDisc* ⇒ *PoincareDisc* ⇒ *PoincareDisc* ⇒ *bool* **where**
  $cong_m$ *a b c d* ⟷ $distance_m$ *a b* = $distance_m$ *c d*

end
theory *TarskiIsomorphism*
  imports *Poincare-Disc.Tarski*
begin

locale *TarskiAbsoluteIso* = *TarskiAbsolute* +
  fixes *φ* :: *'a* ⇒ *'b*
  fixes *cong'* :: *'b* ⇒ *'b* ⇒ *'b* ⇒ *'b* ⇒ *bool*
  fixes *betw'* :: *'b* ⇒ *'b* ⇒ *'b* ⇒ *bool*
  assumes *φbij*: *bij φ*
  assumes *φcong*: ⋀ *x y z w*. *cong'* (*φ x*) (*φ y*) (*φ z*) (*φ w*) ⟷ *cong x y z w*
  assumes *φbetw*: ⋀ *x y z*. *betw'* (*φ x*) (*φ y*) (*φ z*) ⟷ *betw x y z*

sublocale *TarskiAbsoluteIso* ⊆ *TA*: *TarskiAbsolute cong' betw'*
⟨*proof*⟩

context *TarskiAbsoluteIso*
begin
lemma *φon-line*:
  shows *TA.on-line* (*φ p*) (*φ a*) (*φ b*) ⟷ *on-line p a b*
  ⟨*proof*⟩

lemma *φon-ray*:
  shows *TA.on-ray* (*φ p*) (*φ a*) (*φ b*) ⟷ *on-ray p a b*
  ⟨*proof*⟩

lemma *φin-angle*:
  shows *TA.in-angle* (*φ p*) (*φ a*) (*φ b*) (*φ c*) ⟷ *in-angle p a b c*
⟨*proof*⟩

lemma *φray-meets-line*:

shows *TA.ray-meets-line* (φ *ra*) (φ *rb*) (φ *la*) (φ *lb*) $\longleftrightarrow$
ray-meets-line ra rb la lb
⟨*proof*⟩

end

locale *TarskiHyperbolicIso* = *TarskiHyperbolic* +
fixes φ :: *'a* ⇒ *'b*
fixes *cong'* :: *'b* ⇒ *'b* ⇒ *'b* ⇒ *'b* ⇒ *bool*
fixes *betw'* :: *'b* ⇒ *'b* ⇒ *'b* ⇒ *bool*
assumes φ*bij*: *bij* φ
assumes φ*cong*: $\bigwedge$ *x y z w. cong'* (φ *x*) (φ *y*) (φ *z*) (φ *w*) $\longleftrightarrow$ *cong x y z w*
assumes φ*betw*: $\bigwedge$ *x y z. betw'* (φ *x*) (φ *y*) (φ *z*) $\longleftrightarrow$ *betw x y z*

sublocale *TarskiHyperbolicIso* ⊆ *TAI*: *TarskiAbsoluteIso*
⟨*proof*⟩

sublocale *TarskiHyperbolicIso* ⊆ *TarskiHyperbolic cong' betw'*
⟨*proof*⟩

locale *ElementaryTarskiHyperbolicIso* = *ElementaryTarskiHyperbolic* +
fixes φ :: *'a* ⇒ *'b*
fixes *cong'* :: *'b* ⇒ *'b* ⇒ *'b* ⇒ *'b* ⇒ *bool*
fixes *betw'* :: *'b* ⇒ *'b* ⇒ *'b* ⇒ *bool*
assumes φ*bij*: *bij* φ
assumes φ*cong*: $\bigwedge$ *x y z w. cong'* (φ *x*) (φ *y*) (φ *z*) (φ *w*) $\longleftrightarrow$ *cong x y z w*
assumes φ*betw*: $\bigwedge$ *x y z. betw'* (φ *x*) (φ *y*) (φ *z*) $\longleftrightarrow$ *betw x y z*

sublocale *ElementaryTarskiHyperbolicIso* ⊆ *THI*: *TarskiHyperbolicIso*
⟨*proof*⟩

sublocale *ElementaryTarskiHyperbolicIso* ⊆ *ElementaryTarskiHyperbolic cong' betw'*
⟨*proof*⟩

end
theory *MobiusGyroTarski*
imports *MobiusGeometry TarskiIsomorphism Poincare-Disc.Poincare-Tarski*
begin

This theory depends on the following AFP entries:
https://www.isa-afp.org/entries/Poincare_Disc.html https://www.isa-afp.org/entries/Complex_Geometry.html

They must be downloaded in order to check this theory.

The following lemmas can be moved to the cited AFP entries.

lemma *eqArgLessCmod*:
assumes $u \neq 0$ $v \neq 0$
shows *Arg* u = *Arg* v ∧ *cmod* u ≤ *cmod* v $\longleftrightarrow$ (∃ k. $k \geq 0$ ∧ $k \leq 1$ ∧ u = *cor* $k * v$)

⟨*proof*⟩

lift-definition *p-blaschke* :: *p-point* ⇒ *p-isometry* **is** λ *a*. (*moebius-pt* (*blaschke* (*to-complex a*)))
⟨*proof*⟩

lemma *p-between-p-isometry-pt* [*simp*]:
shows *p-between* (*p-isometry-pt f a*) (*p-isometry-pt f b*) (*p-isometry-pt f c*) ⟷ *p-between a b c*
⟨*proof*⟩

lemma *p-blaschke-id* [*simp*]:
shows *p-isometry-pt* (*p-blaschke x*) *x* = *p-zero*
⟨*proof*⟩

lemma *p-between-0uv*:
shows *p-between p-zero u v* ⟷
(∃ *k*≥*0*. *k* ≤ *1* ∧ *to-complex* (*Rep-p-point u*) = *cor k* ∗ *to-complex* (*Rep-p-point v*))
⟨*proof*⟩

A bijection between AFP type representing the Poincare disc (based on complex homogenous coordinates) and our type for poincare disc (based on ordinary complex numbers)

lift-definition φ :: *p-point* ⇒ *PoincareDisc* **is** *to-complex*
⟨*proof*⟩

lemma *distance-m-p-dist*:
shows *distance-m* (*PoincareDisc.to-complex* (φ *x*)) (*PoincareDisc.to-complex* (φ *y*)) = *p-dist x y*
⟨*proof*⟩

definition *blaschke′* :: *complex* ⇒ *complex* ⇒ *complex* **where**
blaschke′ a z = (*z* − *a*) / (*1* − *cnj a* ∗ *z*)

lemma *blaschke′-translation*:
fixes *a z* :: *complex*
shows *blaschke′ a z* = *oplus-m′* (*ominus-m′ a*) *z*
⟨*proof*⟩

lift-definition *blaschke-g* :: *PoincareDisc* ⇒ *PoincareDisc* ⇒ *PoincareDisc* **is** *blaschke′*
⟨*proof*⟩

lemma *blaschke-translation*:
blaschke-g a z = ($\ominus_m$ *a*) $\oplus_m$ *z*
⟨*proof*⟩

Isomorphism between hyperbolic geometry of Poincare disc defined in AFP entry, and hyperbolic geometry in Mobius gyrovector space. Since these two are isomorphic, the geometry of Mobius gyrovector space satisfies Tarski axioms.

interpretation *MobiusGyroTarskiIso*: *ElementaryTarskiHyperbolicIso p-congruent p-between* φ *cong*$_m$ *Mobius-pre-gyrovector-space.between*
⟨*proof*⟩

interpretation *MobiusGyroTarski*: *ElementaryTarskiHyperbolic cong*$_m$ *Mobius-pre-gyrovector-space.between*
⟨*proof*⟩

end
theory *MobiusGyrotrigonometry*
imports *Main GammaFactor PoincareDisc MobiusGyroVectorSpace MoreComplex*
begin

lemma *m-gamma-h1*:
shows $\ominus_m\ a \oplus_m\ b$ = *of-complex* ((*to-complex b* − *to-complex a*) / (*1* − *cnj* (*to-complex a*) ∗ *to-complex b*))
⟨*proof*⟩

lemma *m-gamma-h2*:
shows $(\langle\langle\ominus_m\ a \oplus_m\ b\rangle\rangle)^2$ =
$((\langle\langle b\rangle\rangle)^2 + (\langle\langle a\rangle\rangle)^2$ − (*to-complex a*) ∗ *cnj* (*to-complex b*) − *cnj* (*to-complex a*) ∗ (*to-complex b*)) /
(*1* − (*to-complex a*) ∗ *cnj* (*to-complex b*) − *cnj*(*to-complex a*) ∗ (*to-complex b*) + $(\langle\langle a\rangle\rangle)^2 * (\langle\langle b\rangle\rangle)^2$)
⟨*proof*⟩

lemma *m-gamma-h3*:
shows *1* − $(\langle\langle\ominus_m\ a \oplus_m\ b\rangle\rangle)^2$ =
(*1* − $(\langle\langle b\rangle\rangle)^2 - (\langle\langle a\rangle\rangle)^2 + (\langle\langle a\rangle\rangle)^2 * (\langle\langle b\rangle\rangle)^2$) /
(*1* − (*to-complex a*) ∗ *cnj* (*to-complex b*) − *cnj* (*to-complex a*) ∗ (*to-complex b*) + $(\langle\langle a\rangle\rangle)^2 * (\langle\langle b\rangle\rangle)^2$) (**is** *?lhs* = *?rhs*)
⟨*proof*⟩

lift-definition *gammma-factor-m* :: *PoincareDisc* ⇒ *real* (γ_m) **is** *gamma-factor*
⟨*proof*⟩

lemma *m-gamma-h4*:
shows $(\gamma_m\ (\ominus_m\ a \oplus_m\ b))^2$ =
(*1* − (*to-complex a*) ∗ *cnj* (*to-complex b*) − *cnj* (*to-complex a*) ∗ (*to-complex b*) + $(\langle\langle a\rangle\rangle)^2 * (\langle\langle b\rangle\rangle)^2$) /
(*1* − $(\langle\langle b\rangle\rangle)^2 - (\langle\langle a\rangle\rangle)^2 + (\langle\langle a\rangle\rangle)^2 * (\langle\langle b\rangle\rangle)^2$)
⟨*proof*⟩

lemma *m-gamma-equation*:

shows $(\gamma_m\ (\ominus_m\ a \oplus_m\ b))^2 = (\gamma_m\ a)^2 * (\gamma_m\ b)^2 * (1 - 2 * a \cdot b + (\langle\langle a\rangle\rangle)^2 * (\langle\langle b\rangle\rangle)^2)$
⟨*proof*⟩

lemma *T8-25-help1*:
assumes $A\ t \neq B\ t\ A\ t \neq C\ t\ C\ t \neq B\ t$
$a = (\langle\langle \textit{Mobius-pre-gyrovector-space.get-a}\ t\rangle\rangle)^2$ $b = (\langle\langle \textit{Mobius-pre-gyrovector-space.get-b}\ t\rangle\rangle)^2$ $c = (\langle\langle \textit{Mobius-pre-gyrovector-space.get-c}\ t\rangle\rangle)^2$
shows $\textit{to-complex}\ ((\textit{of-complex}\ a) \oplus_m (\textit{of-complex}\ b) \oplus_m (\ominus_m\ (\textit{of-complex}\ c))) =$
$(a + b - c - a*b*c) / (1 + a*b - a*c - b*c)$ (**is** $?lhs = ?rhs$)
⟨*proof*⟩

lemma *T8-25-help2*:
fixes t :: *PoincareDisc otriangle*
assumes $(A\ t) \neq (B\ t)\ (A\ t) \neq (C\ t)\ (C\ t) \neq (B\ t)$
$a = \langle\langle \textit{Mobius-pre-gyrovector-space.get-a}\ t\rangle\rangle$ $b = \langle\langle \textit{Mobius-pre-gyrovector-space.get-b}\ t\rangle\rangle$ $c = \langle\langle \textit{Mobius-pre-gyrovector-space.get-c}\ t\rangle\rangle$
$\textit{gamma} = \textit{Mobius-pre-gyrovector-space.get-gamma}\ t$
shows $\cos\ \textit{gamma} = (a^2 + b^2 - c^2 - (a*b*c)^2) / (2 * a * b * (1 - c^2))$
⟨*proof*⟩

lemma *T8-25-help3*:
fixes t :: *PoincareDisc otriangle*
assumes $(A\ t) \neq (B\ t)\ (A\ t) \neq (C\ t)\ (C\ t) \neq (B\ t)$
$a = \langle\langle \textit{Mobius-pre-gyrovector-space.get-a}\ t\rangle\rangle$ $b = \langle\langle \textit{Mobius-pre-gyrovector-space.get-b}\ t\rangle\rangle$ $c = \langle\langle \textit{Mobius-pre-gyrovector-space.get-c}\ t\rangle\rangle$
$\textit{gamma} = \textit{Mobius-pre-gyrovector-space.get-gamma}\ t$
$\textit{beta-a} = 1 / \textit{sqrt}\ (1 + a^2)$ $\textit{beta-b} = 1 / \textit{sqrt}\ (1+b^2)$
shows $2 * \textit{beta-a}^2 * a * \textit{beta-b}^2 * b * \cos\ \textit{gamma} = (a^2 + b^2 - c^2 - (a*b*c)^2) / ((1 + a^2) * (1 + b^2) * (1-c^2))$
⟨*proof*⟩

lemma *T8-25-help4*:
fixes t :: *PoincareDisc otriangle*
assumes $(A\ t) \neq (B\ t)\ (A\ t) \neq (C\ t)\ (C\ t) \neq (B\ t)$
$a = \langle\langle \textit{Mobius-pre-gyrovector-space.get-a}\ t\rangle\rangle$ $b = \langle\langle \textit{Mobius-pre-gyrovector-space.get-b}\ t\rangle\rangle$ $c = \langle\langle \textit{Mobius-pre-gyrovector-space.get-c}\ t\rangle\rangle$
$\textit{gamma} = \textit{Mobius-pre-gyrovector-space.get-gamma}\ t$
$\textit{beta-a} = 1 / \textit{sqrt}\ (1 + a^2)$ $\textit{beta-b} = 1 / \textit{sqrt}\ (1+b^2)$
shows $1 - 2 * \textit{beta-a}^2 * a * \textit{beta-b}^2 * b * \cos\ \textit{gamma} =$
$(1 + (a*b)^2 - (a*c)^2 - (b*c)^2) / ((1 + a^2) * (1 + b^2) * (1-c^2))$
⟨*proof*⟩

lemma *T25-help5*:
fixes t :: *PoincareDisc otriangle*
assumes $(A\ t) \neq (B\ t)\ (A\ t) \neq (C\ t)\ (C\ t) \neq (B\ t)$
$a = \langle\langle \textit{Mobius-pre-gyrovector-space.get-a}\ t\rangle\rangle$ $b = \langle\langle \textit{Mobius-pre-gyrovector-space.get-b}\ t\rangle\rangle$

t⟫ *c* = ⟪*Mobius-pre-gyrovector-space.get-c t*⟫
gamma = *Mobius-pre-gyrovector-space.get-gamma t*
beta-a = *1* / *sqrt* (*1* + a^2) *beta-b* = *1* / *sqrt* (*1*+b^2)
shows (*2* ∗ *beta-*a^2 ∗ *a* ∗ *beta-*b^2 ∗ *b* ∗ *cos gamma*) / (*1* − *2* ∗ *beta-*a^2 ∗ *a* ∗ *beta-*b^2 ∗ *b* ∗ *cos gamma*) =
to-complex ((*of-complex* (a^2)) $\oplus_m$ (*of-complex* (b^2)) $\oplus_m$ ($\ominus_m$ (*of-complex* (c^2)))) (**is** *?lhs* = *?rhs*)
⟨*proof*⟩

lemma *T25-MobiusCosineLaw*:
fixes *t* :: *PoincareDisc otriangle*
assumes (*A t*) ≠ (*B t*) (*A t*) ≠ (*C t*) (*C t*) ≠ (*B t*)
a = ⟪*Mobius-pre-gyrovector-space.get-a t*⟫ *b* = ⟪*Mobius-pre-gyrovector-space.get-b t*⟫ *c* = ⟪*Mobius-pre-gyrovector-space.get-c t*⟫
gamma = *Mobius-pre-gyrovector-space.get-gamma t*
beta-a = *1* / *sqrt* (*1* + a^2) *beta-b* = *1* / *sqrt* (*1*+b^2)
shows c^2 = *to-complex* ((*of-complex* (a^2)) $\oplus_m$ (*of-complex* (b^2)) $\oplus_m$ ($\ominus_m$ (*of-complex*
(*2* ∗ *beta-*a^2 ∗ *a* ∗ *beta-*b^2 ∗ *b* ∗ *cos*(*gamma*) /
(*1* − *2* ∗ *beta-*a^2 ∗ *a* ∗ *beta-*b^2 ∗ *b* ∗ *cos gamma*)))))
⟨*proof*⟩

abbreviation *add-complex* (**infixl** $\oplus_{mc}$ *100*) **where**
add-complex c1 c2 ≡ *to-complex* (*of-complex c1* $\oplus_m$ *of-complex c2*)

lemma *T-MobiusPythagorean*:
fixes *t* :: *PoincareDisc otriangle*
assumes (*A t*) ≠ (*B t*) (*A t*) ≠ (*C t*) (*C t*) ≠ (*B t*)
a = ⟪*Mobius-pre-gyrovector-space.get-a t*⟫ *b* = ⟪*Mobius-pre-gyrovector-space.get-b t*⟫ *c* = ⟪*Mobius-pre-gyrovector-space.get-c t*⟫
gamma = *Mobius-pre-gyrovector-space.get-gamma t gamma* = *pi* / *2*
shows c^2 = a^2 $\oplus_{mc}$ b^2
⟨*proof*⟩

end
theory *Poincare*
imports *Complex-Main HOL−Analysis.Inner-Product GammaFactor*
begin

typedef *PoincareDisc* = {*z*::*complex*. *cmod z* < *1*}
⟨*proof*⟩

setup-lifting *type-definition-PoincareDisc*

abbreviation *to-complex* :: *PoincareDisc* ⇒ *complex* **where**
to-complex ≡ *Rep-PoincareDisc*
abbreviation *of-complex* :: *complex* ⇒ *PoincareDisc* **where**
of-complex ≡ *Abs-PoincareDisc*

lemma *poincare-disc-two-elems*:
shows ∃ *z1 z2*::*PoincareDisc*. *z1* ≠ *z2*
⟨*proof*⟩

lift-definition *inner-p* :: *PoincareDisc* ⇒ *PoincareDisc* ⇒ *real* (**infixl** · *100*) **is** *inner* ⟨*proof*⟩

lift-definition *norm-p* :: *PoincareDisc* ⇒ *real* (⟪-⟫ [*100*] *101*) **is** *norm* ⟨*proof*⟩

lemma *norm-lt-one*:
shows ⟪*u*⟫ < *1*
⟨*proof*⟩

lemma *norm-geq-zero*:
shows ⟪*u*⟫ ≥ *0*
⟨*proof*⟩

lemma *square-norm-inner*:
shows (⟪*u*⟫)^2 = *u* · *u*
⟨*proof*⟩

lift-definition *gammma-factor-p* :: *PoincareDisc* ⇒ *real* (γ_p) **is** *gamma-factor* ⟨*proof*⟩

lemma *gamma-factor-p-nonzero* [*simp*]:
shows γ_p *u* ≠ *0*
⟨*proof*⟩

lemma *gamma-factor-p-positive* [*simp*]:
shows γ_p *u* > *0*
⟨*proof*⟩

lemma *norm-square-gamma-factor-p*:
shows (⟪*u*⟫)^*2* = *1* − *1* / (γ_p *u*)^*2*
⟨*proof*⟩

lemma *norm-square-gamma-factor-p'*:
shows (⟪*u*⟫)^*2* = ((γ_p *u*)^*2* − *1*) / (γ_p *u*)^*2*
⟨*proof*⟩

lemma *gamma-factor-p-square-norm*:
shows (γ_p *u*)^2 = *1* / (*1* − (⟪*u*⟫)^2)
⟨*proof*⟩

end

References

[1] A. A. Ungar. *Analytic Hyperbolic Geometry: Mathematical Foundations and Applications*. World Scientific, Singapore, 2005.