Feuerbach's Theorem

Lawrence C Paulson

June 25, 2026

Abstract

Feuerbach's theorem is a classical result in Euclidean geometry about certain circles associated with a triangle. It states that the nine-point circle is tangent internally to the triangle's incircle and tangent externally to each of the three excircles, where the *incircle* is the circle tangent to all three sides of the triangle; the *excircles* are circles tangent to one side and the extensions of the other two; the *nine-point circle* is the circle passing through nine special points of the triangle.

This development is an experiment in automatic translation from one proof assistant to another using AI. The HOL Light file `feuerbach.ml` was given, one proof at a time, to Claude 4.6 using the Isabelle Assistant plug-in.[1] Despite coaxing and hints, the autoformalisation process transformed a 214-line HOL Light file to nearly 1500 lines of Isabelle/HOL, though one that could be processed in five seconds. Part of the excess length comes from spelling out two transformations (translation of a point to zero, followed by rotation of a side to the vector $(1, 0)$) that in HOL Light is done silently by WLOG tactics. The current proof has been heavily simplified, in part using Claude but mostly manually, and is now under 750 lines. It is notable that Claude often relied on the **algebra** proof method, despite its relative obscurity.

Contents

[1] Part of AutoCorrode: `https://github.com/awslabs/AutoCorrode`

```
theory Feuerbach
  imports HOL−Analysis.Analysis
begin
```

1 Algebraic condition for two circles to be tangent to each other

```
lemma inner-real2: (x::real^2) · y = x$1 ∗ y$1 + x$2 ∗ y$2
  by (simp add: inner-vec-def UNIV-2)

lemma norm-real2-sq: (norm (x::real^2))^2 = (x$1)^2 + (x$2)^2
 by (simp add: power2-norm-eq-inner inner-real2 power2-eq-square[of x$1] power2-eq-square[of
x$2])

lemma dist-real2-sq: (dist (a::real^2) b)^2 = (a$1 − b$1)^2 + (a$2 − b$2)^2
  by (simp add: dist-norm norm-real2-sq)

lemma collinear-3-2d:
  collinear {(a::real^2), b, c} ⟷
   (b$1 − a$1) ∗ (c$2 − a$2) = (c$1 − a$1) ∗ (b$2 − a$2)
proof −
  have collinear {a, b, c} = (a = b ∨ c = b ∨ (∃ k. c − b = k ∗R (a − b)))
    by (simp add: collinear-3 collinear-lemma)
  also have ... = ((b$1 − a$1) ∗ (c$2 − a$2) = (c$1 − a$1) ∗ (b$2 − a$2))
    (is ?L = ?R)
  proof
    assume ?L
    then consider a = b | c = b | k where c − b = k ∗R (a − b)
      by auto
    then show ?R
    proof cases
      case (3 k)
      then have ⋀i. (c − b)$i = (k ∗R (a − b))$i by (simp add: vec-eq-iff)
      from this [of 1] this [of 2]
      show ?thesis
        by (simp add: vec-eq-iff) algebra
    qed auto
  next
    assume ?R
    then have det: (a$1 − b$1) ∗ (c$2 − b$2) = (c$1 − b$1) ∗ (a$2 − b$2)
      by (simp add: algebra-simps)
    show ?L
    proof (cases a = b)
      case True then show ?thesis by simp
    next
      case False
      then obtain i :: 2 where i-ne: (a − b) $ i ≠ 0
        by (auto simp: vec-eq-iff)
```

```
      define k where k = (c − b) $ i / (a − b) $ i
      have c − b = k *R (a − b)
      proof (rule vec-eq-iff [THEN iffD2], rule allI)
        fix j :: 2
        have (c$j − b$j) ∗ (a$i − b$i) = (c$i − b$i) ∗ (a$j − b$j)
          using exhaust-2 [of i] using exhaust-2 [of j] det
          by (auto simp: algebra-simps)
        then show (c − b) $ j = (k *R (a − b)) $ j
          using i-ne by (simp add: eq-divide-imp k-def)
      qed
      then show ?thesis by auto
    qed
  qed
  finally show ?thesis .
qed
```

2 Geometric invariance lemmas

```
lemma collinear-linear-image:
  fixes f :: 'a::euclidean-space ⇒ 'b::euclidean-space
  assumes linear f inj f
  shows collinear (f ‘ S) = collinear S
  by (simp add: assms collinear-aff-dim)

lemma collinear-orthogonal-transformation:
  fixes f :: 'a::euclidean-space ⇒ 'b::euclidean-space
  assumes orthogonal-transformation f
  shows collinear {f a, f b, f c} = collinear {a, b, c}
  using assms collinear-linear-image orthogonal-transformation-inj
    orthogonal-transformation-linear by force

lemma rotation-to-x-axis:
  fixes v :: real^2
  assumes v ≠ 0
  obtains R :: real^2 ⇒ real^2 where orthogonal-transformation R (R v)$1 =
norm v (R v)$2 = 0
proof −
  define p where p = v$1
  define q where q = v$2
  define r where r = norm v
  have r-pos: r > 0 using assms by (simp add: r-def)
  have r-ne: r ≠ 0 using r-pos by simp
  have r-sq: r^2 = p^2 + q^2
    unfolding r-def p-def q-def by (simp add: norm-real2-sq)
  define R :: real^2 ⇒ real^2 where
    R x = (χ i. if i = 1 then (p ∗ x$1 + q ∗ x$2) / r else (−q ∗ x$1 + p ∗ x$2)
/ r) for x
  have R1: ⋀x. (R x)$1 = (p ∗ x$1 + q ∗ x$2) / r
    unfolding R-def by simp
```

```
  have R2: ⋀x. (R x)$2 = (−q ∗ x$1 + p ∗ x$2) / r
    unfolding R-def by simp
  have lin: linear R
  proof (rule linearI)
    fix x y :: real^2
    show R (x + y) = R x + R y
       by (simp add: vec-eq-iff R1 R2 forall-2 add-divide-distrib diff-divide-distrib
algebra-simps)
  next
    fix c :: real and x :: real^2
    show R (c ∗R x) = c ∗R R x
      by (simp add: vec-eq-iff R1 R2 forall-2 algebra-simps)
  qed
  have inner-pres: R x · R y = x · y for x y :: real^2
    using r-sq r-ne by (simp add: inner-real2 R1 R2 field-simps) algebra
  have orth: orthogonal-transformation R
    unfolding orthogonal-transformation-def using lin inner-pres by auto
  have rv1: (R v)$1 = norm v
     by (metis R1 nonzero-mult-div-cancel-left p-def power2-eq-square q-def r-def
r-ne r-sq)
  have rv2: (R v)$2 = 0
    by (simp add: R2 p-def q-def)
  from orth rv1 rv2 show ?thesis by (rule that[where R=R])
qed

lemma collinear-translate:
  fixes x :: 'a::euclidean-space
  shows collinear {x, y, z} = collinear {x − a, y − a, z − a}
  by (simp add: between-norm collinear-between-cases)

lemma int-case-lemma:
  fixes c1 c2 :: (real,2) vec
  assumes 0 ≤ r1 r1 < r2 and d: dist c1 c2 = |r1 − r2| and int: dist c1 c2 =
|r1 − r2|
  shows c1 = c2 ∧ r1 = r2 ∨ (∃!x. dist c1 x = r1 ∧ dist c2 x = r2)
proof −
  have r-diff: r2 − r1 > 0 using assms by linarith
  have abs-eq: dist c1 c2 = r2 − r1
    using int assms by simp
  define x where x = c1 + (r1 / (r2 − r1)) ∗R (c1 − c2)
  have dist-c1c2: norm (c1 − c2) = r2 − r1
    using abs-eq by (simp add: dist-norm)
  have dist1: dist c1 x = r1
    using assms r-diff
    by (simp add: dist-norm x-def divide-simps norm-minus-commute)
  have dist2: dist c2 x = r2
  proof −
    have c2 − x = (1 + r1 / (r2 − r1)) ∗R (c2 − c1)
      by (simp add: x-def scaleR-add-left scaleR-diff-right)
```

```
    also have 1 + r1 / (r2 − r1) = r2 / (r2 − r1)
      using r-diff by (simp add: field-simps)
    finally show ?thesis
      using assms r-diff
      by (simp add: dist-norm divide-simps norm-minus-commute)
  qed
  have unique: y = x if dist c1 y = r1 dist c2 y = r2 for y
  proof −
    have nor: norm (c1 − y) = r1 norm (c2 − y) = r2
      using that by (auto simp add: dist-norm)
    have dist c2 y = dist c2 c1 + dist c1 y
      using that abs-eq by (simp add: dist-commute)
    then have norm (c2 − c1) *R (c1 − y) = norm (c1 − y) *R (c2 − c1)
      by (simp add: dist-triangle-eq)
    then have eq-y: (r2 − r1) *R y = r2 *R c1 − r1 *R c2
      using dist-c1c2 nor by (simp add: norm-minus-commute algebra-simps)
    have eq-x: (r2 − r1) *R x = r2 *R c1 − r1 *R c2
      using r-diff
      by (simp add: x-def scaleR-add-right scaleR-diff-left scaleR-diff-right)
    show y = x using eq-y eq-x r-diff
      by (metis scaleR-cancel-left less-irrefl)
  qed
  show ?thesis
    using dist1 dist2 unique by blast
qed

theorem circles-tangent:
  fixes r1 r2 :: real and c1 c2 :: real^2
  assumes 0 ≤ r1 0 ≤ r2
    and dist c1 c2 = r1 + r2 ∨ dist c1 c2 = |r1 − r2|
  shows c1 = c2 ∧ r1 = r2 ∨ (∃!x::real^2. dist c1 x = r1 ∧ dist c2 x = r2)
  using assms
proof −
  consider (ext) dist c1 c2 = r1 + r2 | (int) dist c1 c2 = |r1 − r2|
    using assms(3) by blast
  then show ?thesis
  proof cases
    case ext
    show ?thesis
    proof (cases r1 + r2 = 0)
      case True then show ?thesis
        using assms(1,2) local.ext by auto
    next
      case False
      then have r-pos: r1 + r2 > 0 using assms(1,2) by linarith
      define x where x = c1 + (r1 / (r1 + r2)) *R (c2 − c1)
      have x-sub-c1: x − c1 = (r1 / (r1 + r2)) *R (c2 − c1)
        by (simp add: x-def)
      have x-sub-c2: x − c2 = − (r2 / (r1 + r2)) *R (c2 − c1)
```

```
    proof -
      have x - c2 = (r1 / (r1 + r2) - 1) *R (c2 - c1)
        by (simp add: x-def scaleR-diff-left)
      also have r1 / (r1 + r2) - 1 = - (r2 / (r1 + r2))
        using r-pos by (simp add: field-simps)
      finally show ?thesis by simp
    qed
    have dist-c1c2: norm (c2 - c1) = r1 + r2
      using ext by (simp add: dist-norm norm-minus-commute)
    have dist1: dist c1 x = r1
      using dist-c1c2 assms(1) r-pos
      by (simp add: dist-norm x-def divide-simps)
    have dist2: dist c2 x = r2
    proof -
      have c2 - x = (r2 / (r1 + r2)) *R (c2 - c1)
      using x-sub-c2 by (metis minus-diff-eq neg-equal-iff-equal scaleR-minus-left)
      then have norm (c2 - x) = r2 / (r1 + r2) * norm (c2 - c1)
        using assms(2) r-pos by (simp add: divide-simps)
      then show ?thesis
        using dist-c1c2 r-pos by (simp add: dist-norm)
    qed
    have unique: y = x if dist c1 y = r1 dist c2 y = r2 for y
    proof -
    have norm-c1y: norm (c1 - y) = r1 using that(1) by (simp add: dist-norm)
      have norm-yc2: norm (y - c2) = r2
        using that(2) by (simp add: dist-norm norm-minus-commute)
      have dist c1 c2 = dist c1 y + dist y c2
        using that ext by (simp add: dist-commute)
      then have norm (c1 - y) *R (y - c2) = norm (y - c2) *R (c1 - y)
        by (simp add: dist-triangle-eq)
      then have (r1 + r2) *R y = r2 *R c1 + r1 *R c2
        using norm-c1y norm-yc2 by (simp add: algebra-simps)
      moreover have (r1 + r2) *R x = r2 *R c1 + r1 *R c2
        using r-pos
        by (simp add: x-def scaleR-add-left scaleR-add-right scaleR-diff-right)
      ultimately show y = x
        using r-pos by (metis scaleR-cancel-left less-irrefl)
    qed
    show ?thesis
      using dist1 dist2 unique by blast
  qed
next
  case int
  show ?thesis
  proof (cases r1 = r2)
    case True
    with int show ?thesis by auto
  next
    case False
```

```
      show ?thesis
      proof (cases r1 ≤ r2)
        case True
        then show ?thesis
          using int-case-lemma False ‹0 ≤ r1› int less-eq-real-def by blast
      next
        case False
        then have c1 = c2 ∧ r2 = r1 ∨ (∃!x. dist c2 x = r2 ∧ dist c1 x = r1)
              using int-case-lemma[of r2 r1 c2 c1] ‹0 ≤ r2› int by (simp add:
dist-commute)
        then show ?thesis
          by blast
      qed
    qed
  qed
qed
```

3 Feuerbach's theorem

Given a non-degenerate triangle *abc*, let the circle passing through the midpoints of its sides (the "9 point circle") have center *ncenter* and radius *nradius*. Now suppose the circle with center *icenter* and radius *iradius* is tangent to three sides (either internally or externally to one side and two produced sides), meaning that it's either the inscribed circle or one of the three escribed circles. Then the two circles are tangent to each other, i.e. either they are identical or they touch at exactly one point.

```
locale Feuerbach =
  fixes a b c :: real^2
    and mbc mac mab :: real^2
    and pbc pac pab :: real^2
    and ncenter icenter :: real^2
    and nradius iradius :: real
  assumes mab-def: mab = midpoint a b
      and mbc-def: mbc = midpoint b c
      and mac-def: mac = midpoint c a
  assumes nr1: dist ncenter mbc = nradius
      and nr2: dist ncenter mac = nradius
      and nr3: dist ncenter mab = nradius
      and ir1: dist icenter pbc = iradius
      and ir2: dist icenter pac = iradius
      and ir3: dist icenter pab = iradius
      and col-ab: collinear {a, b, pab} and orth-ab: orthogonal (a − b) (icenter −
pab)
      and col-bc: collinear {b, c, pbc} and orth-bc: orthogonal (b − c) (icenter −
pbc)
      and col-ac: collinear {a, c, pac} and orth-ac: orthogonal (a − c) (icenter −
pac)
```

begin

lemma *special*:
  assumes *bx*: *b$2 = 0* **and** *bne*: *b$1 ≠ 0* **and** *cne*: *c$2 ≠ 0* **and** [*simp*]: *a=0*
  shows *ncenter = icenter* ∧ *nradius = iradius* ∨
    (∃!*x*::*real^2*. *dist ncenter x = nradius* ∧ *dist icenter x = iradius*)
proof −
  — First establish non-negativity of radii
  obtain *0 ≤ nradius 0 ≤ iradius* **using** *nr1 ir1 zero-le-dist* **by** *metis*
    — Non-collinearity: since a=0, b on x-axis (*b $ 2 = 0*, *b $ 1 ≠ 0*), and *c $ 2 ≠ 0*
  have *ncol*: ¬ *collinear* {*0*, *b*, *c*}
    by (*simp add*: *assms collinear-3-2d*)
    — Introduce coordinate names
  define *bx* **where** *bx = b$1*
  define *cx* **where** *cx = c$1*
  define *cy* **where** *cy = c$2*
  — Expand midpoint coordinates
  have *mab-eq*: *mab* = (*1/2*) $*_R$ *b* **and** *mbc-eq*: *mbc* = (*1/2*) $*_R$ (*b* + *c*) **and** *mac-eq*: *mac* = (*1/2*) $*_R$ *c*
    by (*auto simp add*: *midpoint-def mab-def mbc-def mac-def*)
  have *mab1*: *mab$1 = bx / 2* **and** *mab2*: *mab$2 = 0*
    using *mab-eq bx* **by** (*simp-all add*: *bx-def*)
  have *mbc1*: *mbc$1* = (*bx + cx*) / *2* **and** *mbc2*: *mbc$2 = cy / 2*
    using *mbc-eq bx* **by** (*simp-all add*: *bx-def cx-def cy-def add-divide-distrib*)
  have *mac1*: *mac$1 = cx / 2* **and** *mac2*: *mac$2 = cy / 2*
    using *mac-eq* **by** (*simp-all add*: *cx-def cy-def*)
    — Extract collinearity conditions
  have *pab2*: *pab$2 = 0*
    using *bne bx col-ab collinear-3-2d* **by** *force*
  have *col-bc-eq*: (*c$1* − *b$1*) ∗ (*pbc$2* − *b$2*) = (*pbc$1* − *b$1*) ∗ (*c$2* − *b$2*)
    using *col-bc collinear-3-2d*[*of b c pbc*] **by** *simp*
  have *pbc-col*: (*cx* − *bx*) ∗ *pbc$2* = (*pbc$1* − *bx*) ∗ *cy*
    using *col-bc-eq bx* **by** (*simp add*: *bx-def cx-def cy-def*)
  have *col-ac-eq*: *c$1* ∗ *pac$2* = *pac$1* ∗ *c$2*
    using *col-ac collinear-3-2d*[*of 0 c pac*] **by** *simp*
  have *pac-col*: *cx* ∗ *pac$2* = *pac$1* ∗ *cy*
    using *col-ac-eq* **by** (*simp add*: *cx-def cy-def*)
    — Extract orthogonality conditions
  have *orth-ab-eq*: *icenter$1 = pab$1*
    using *orth-ab* **by** (*simp add*: *orthogonal-def inner-real2 bx bne*)
  have *orth-bc-eq*: (*bx* − *cx*) ∗ (*icenter$1* − *pbc$1*) − *cy* ∗ (*icenter$2* − *pbc$2*) = *0*
    using *orth-bc* **by** (*simp add*: *orthogonal-def inner-real2 cx-def cy-def bx bx-def*)
  have *orth-ac-eq*: *cx* ∗ (*icenter$1* − *pac$1*) + *cy* ∗ (*icenter$2* − *pac$2*) = *0*
    using *orth-ac* **by** (*simp add*: *orthogonal-def inner-real2 cx-def cy-def*)
    — Approach: use theorem *circles-tangent*. We need to show the distance condition.

```
  — We work entirely in coordinates with a = 0, b = (bx, 0), c = (cx, cy).
  — All geometric conditions become polynomial equations.
  — Strategy: solve for ncenter, nradius, icenter, iradius from constraints,
  — then verify the tangency condition.

— Solve for ncenter from equidistance to midpoints
— n1 = ncenter $ 1, n2 = ncenter $ 2
have n1-val: ncenter$1 = (bx + 2*cx) / 4
  using dist-real2-sq[of ncenter mbc] mbc1 mbc2
  using dist-real2-sq[of ncenter mac] mac1 mac2 bne
  unfolding nr1 nr2 bx-def by algebra

have n2-val: ncenter$2 * cy = (cy^2 + bx*cx − cx^2) / 4
  using dist-real2-sq[of ncenter mbc] mbc1 mbc2
  using dist-real2-sq[of ncenter mab] mab1 mab2
  by (simp add: nr1 nr3 n1-val power2-eq-square field-simps)

have nradius-sq: nradius^2 = (ncenter$1 − bx/2)^2 + ncenter$2^2
  using dist-real2-sq mab1 mab2 nr3 by force

— Extract incircle foot point and center coordinates
— From pab $ 2 = 0 and icenter $ 1 = pab $ 1, we get iradius = |icenter $ 2|
have irad-sq: iradius^2 = (icenter$2)^2
  using dist-real2-sq ir3 orth-ab-eq pab2 by fastforce

— Derive incircle squared-distance to each side
— Key idea: pbc is foot of perpendicular from icenter to line bc.
— From orthogonality (bx-cx)*u = cy*v where u=icenter $ 1-pbc $ 1, v=icenter
$ 2-pbc $ 2.
— From collinearity, cy*(icenter $ 1-bx) - (cx-bx)*icenter $ 2 = cy*u - (cx-bx)*v.
— Then (uš+vš)*D = (cy*u-(cx-bx)*v)š because (cx-bx)*u+cy*v=0 (Lagrange
identity).

have irad-sq-bc: iradius^2 * ((cx−bx)^2 + cy^2) = (cy*(icenter$1−bx) −
(cx−bx)*icenter$2)^2
proof −
  define u where u = icenter$1 − pbc$1
  define v where v = icenter$2 − pbc$2
  have uv-orth: (cx − bx) * u + cy * v = 0
    using orth-bc-eq by (simp add: u-def v-def algebra-simps)
  have irad-uv: iradius^2 = u^2 + v^2
    using dist-real2-sq ir1 u-def v-def by blast
  have cy * (icenter$1 − bx) − (cx − bx) * icenter$2 = cy * u − (cx − bx) * v
    by (simp add: mult.commute pbc-col right-diff-distrib' u-def v-def)
  then show ?thesis using irad-uv uv-orth by algebra
qed

have irad-sq-ac: iradius^2 * (cx^2 + cy^2) = (cy*icenter$1 − cx*icenter$2)^2
proof −
```

define *u* **where** *u = icenter$1 − pac$1*
define *v* **where** *v = icenter$2 − pac$2*
have *uv-orth*: *cx ∗ u + cy ∗ v = 0*
using *orth-ac-eq* **by** (*simp add*: *u-def v-def*)
have *irad-uv*: *iradius^2 = u^2 + v^2*
using *dist-real2-sq ir2 u-def v-def* **by** *blast*
have *cy ∗ icenter$1 − cx ∗ icenter$2 = cy ∗ u − cx ∗ v*
by (*simp add*: *pac-col right-diff-distrib′ u-def v-def*)
then show *?thesis* **using** *irad-uv uv-orth* **by** *algebra*
qed

have *tangent-cond*: *dist ncenter icenter = nradius + iradius ∨ dist ncenter icenter = |nradius − iradius|*
proof −
— Introduce short names for coordinates
define *i1* **where** *i1 = icenter$1*
define *i2* **where** *i2 = icenter$2*
define *n1* **where** *n1 = ncenter$1*
define *n2* **where** *n2 = ncenter$2*
— Restate all equations in terms of i1, i2, n1, n2, bx, cx, cy
from *n1-val* **have** *n1-eq*: *n1 = (bx + 2∗cx) / 4* **by** (*simp add*: *n1-def*)
from *n2-val* **have** *n2-eq*: *n2 ∗ cy = (cy^2 + bx∗cx − cx^2) / 4* **by** (*simp add*: *n2-def*)
from *nradius-sq* **have** *nr-sq*: *nradius^2 = (n1 − bx/2)^2 + n2^2* **by** (*simp add*: *n1-def n2-def*)
from *irad-sq* **have** *ir-sq*: *iradius^2 = i2^2* **by** (*simp add*: *i2-def*)
from *irad-sq-bc irad-sq* **have** *ir-bc*: *i2^2 ∗ ((cx−bx)^2 + cy^2) = (cy∗(i1−bx) − (cx−bx)∗i2)^2*
by (*simp add*: *i1-def i2-def*)
from *irad-sq-ac irad-sq* **have** *ir-ac*: *i2^2 ∗ (cx^2 + cy^2) = (cy∗i1 − cx∗i2)^2*
by (*simp add*: *i1-def i2-def*)
have *key-linear*: *cy ∗ (2∗i1 − bx) = 2 ∗ (i1 + cx − bx) ∗ i2*
using *ir-ac ir-bc bne cne*
by (*simp add*: *bx-def cy-def field-simps*) *algebra*
have *dist-sq*: (*dist ncenter icenter*)*^2 = (n1 − i1)^2 + (n2 − i2)^2*
using *dist-real2-sq*[*of ncenter icenter*] **by** (*simp add*: *n1-def n2-def i1-def i2-def*)
have *cy-ne*: *cy ≠ 0* **using** *cne* **by** (*simp add*: *cy-def*)
have *bx-ne*: *bx ≠ 0* **using** *bne* **by** (*simp add*: *bx-def*)
have *nrad-nn*: *0 ≤ nradius* **using** *nr1 zero-le-dist* **by** *metis*
have *irad-nn*: *0 ≤ iradius* **using** *ir1 zero-le-dist* **by** *metis*
have *dist-nn*: *0 ≤ dist ncenter icenter* **by** *simp*
have *key-sq*: $((dist\ ncenter\ icenter)^2 - nradius^2 - iradius^2)^2 = 4 * nradius^2 * iradius^2$
proof −
— Substitute all definitions to get a purely algebraic statement
have *lhs*: $(dist\ ncenter\ icenter)^2 - nradius^2 - iradius^2 =$
$i1^2 - (bx+2*cx)*i1/2 - 2*n2*i2 + bx*cx/2$
using *dist-sq nr-sq ir-sq* **by** (*simp add*: *n1-eq field-simps power2-eq-square*)

```
      have rhs-eq: 4 * nradius^2 * iradius^2 = ((2*cx−bx)^2/4 + 4*n2^2)*i2^2
        using nr-sq ir-sq key-linear by (simp add: n1-eq field-simps) algebra
      have 4 * cy^2 * (i1^2 − (bx+2*cx)*i1/2 − 2*n2*i2 + bx*cx/2)^2 =
          4 * cy^2 * ((2*cx−bx)^2/4 + 4*n2^2)*i2^2
        using ir-ac n2-eq key-linear by (simp add: field-simps) algebra
      then show ?thesis using lhs rhs-eq cy-ne by simp
    qed
    then have prod-zero: ((dist ncenter icenter)^2 − (nradius + iradius)^2) * ((dist
ncenter icenter)^2 − (nradius − iradius)^2) = 0
      by (simp add: power2-eq-square algebra-simps)
    then show ?thesis
      by (smt (verit) irad-nn mult-eq-0-iff nrad-nn real-abs-dist real-sqrt-abs)
  qed
  show ?thesis
    using circles-tangent[OF ‹0 ≤ nradius› ‹0 ≤ iradius›] tangent-cond by blast
qed

theorem main:
  assumes ¬ collinear {a, b, c}
  shows ncenter = icenter ∧ nradius = iradius ∨
      (∃!x::real^2. dist ncenter x = nradius ∧ dist icenter x = iradius)
proof −
  — Step 1: Translate so that a becomes the origin
  define a' where a' = (0::real^2)
  define b' where b' = b − a
  define c' where c' = c − a
  define mab' where mab' = mab − a
  define mbc' where mbc' = mbc − a
  define mac' where mac' = mac − a
  define pab' where pab' = pab − a
  define pbc' where pbc' = pbc − a
  define pac' where pac' = pac − a
  define ncenter' where ncenter' = ncenter − a
  define icenter' where icenter' = icenter − a
  — Translation preserves all geometric properties
  have ncol': ¬ collinear {a', b', c'}
   using assms(1) collinear-translate[of a b c a] by (simp add: a'-def b'-def c'-def)
  have mid-ab': midpoint a' b' = mab'
   by (simp add: a'-def add-diff-eq b'-def diff-add-eq mab'-def mab-def midpoint-eq-iff)
  have mid-bc': midpoint b' c' = mbc'
   by (simp add: add-diff-eq b'-def c'-def diff-add-eq mbc'-def mbc-def midpoint-eq-iff)
  have mid-ca': midpoint c' a' = mac'
   by (simp add: add-diff-eq a'-def c'-def diff-add-eq mac'-def mac-def midpoint-eq-iff)
  — Now a' = 0, so we have the origin case
  — Step 2: Rotate so that b' lands on the x-axis
  have b'-ne: b' ≠ 0
    using a'-def ncol' by force
  obtain R :: real^2 ⇒ real^2 where
    R-orth: orthogonal-transformation R and R-b1: R b' $1 = norm b' and R-b2:
```

R b′ $2 = 0
  using *rotation-to-x-axis*[*OF b′-ne*] **by** *blast*
 have *R-lin*: *linear R*
  using *R-orth orthogonal-transformation-linear* **by** *blast*
 have *R-inner*: $\bigwedge v\ w.\ R\ v \cdot R\ w = v \cdot w$
  using *R-orth* **by** (*simp add*: *orthogonal-transformation-def*)
 have *R-norm*: $\bigwedge v.$ *norm* (*R v*) = *norm v*
  using *R-orth orthogonal-transformation* **by** *blast*
 have *R-dist*: $\bigwedge x\ y.$ *dist* (*R x*) (*R y*) = *dist x y*
  by (*meson R-orth orthogonal-transformation-isometry*)
 have *R-0*: *R 0 = 0* **using** *R-lin* **by** (*simp add*: *linear-0*)
 have *R-midpoint*: $\bigwedge x\ y.$ *R* (*midpoint x y*) = *midpoint* (*R x*) (*R y*)
  using *R-lin* **by** (*simp add*: *midpoint-def linear-add linear-cmul*)
 have *R-orth-pres*: $\bigwedge u\ v.$ *orthogonal* (*R u*) (*R v*) = *orthogonal u v*
  by (*simp add*: *orthogonal-def R-inner*)
 — Rotated (R b') is on the x-axis with positive first component
 have *b′′-x*: *R b′ $ 2 = 0* **using** *R-b2* **by** *simp*
 have *b′′-ne*: *R b′ $ 1* $\neq$ *0*
  by (*simp add*: *R-b1 b′-ne*)
 — (R c') has nonzero y-component (from non-collinearity)
 have *c′′-y*: *R c′ $ 2* $\neq$ *0*
 proof
  assume *c2-0*: *R c′ $ 2 = 0*
  — If *R c′ $ 2 = 0* then 0, (R b'), (R c') are all on x-axis, hence collinear
  have *collinear* {(*0*::*real^2*), (*R b′*), (*R c′*)}
   by (*simp add*: *R-b2 c2-0 collinear-3-2d*)
  — But collinear is preserved under orthogonal transformation
  then have *collinear* {*R 0*, *R b′*, *R c′*}
   by (*simp add*: *R-0*)
  then have *collinear* {*0*, *b′*, *c′*}
   using *collinear-orthogonal-transformation*[*OF R-orth*, *of 0 b′ c′*] **by** (*simp add*: *R-0*)
  then have *collinear* {*a′*, *b′*, *c′*} **by** (*simp add*: *a′-def*)
  then show *False* **using** *ncol′* **by** *contradiction*
 qed
 — Transfer all properties to rotated coordinates and apply the special case theorem
 interpret *F0*: *Feuerbach 0 R b′ R c′ R mbc′ R mac′ R mab′ R pbc′ R pac′ R pab′ R ncenter′ R icenter′*
 proof
  show *R mab′ = midpoint 0* (*R b′*)
   using *R-0 R-midpoint a′-def mid-ab′* **by** *force*
  show *R mbc′ = midpoint* (*R b′*) (*R c′*)
   using *R-midpoint mid-bc′* **by** *force*
  show *R mac′ = midpoint* (*R c′*) *0*
   using *R-0 R-midpoint a′-def mid-ca′* **by** *force*
  obtain *dist ncenter′ mbc′ = nradius dist ncenter′ mac′ = nradius dist ncenter′ mab′ = nradius*
   by (*metis diff-add-cancel dist-add-cancel2 mab′-def mac′-def mbc′-def ncenter′-def nr1 nr2 nr3*)

then show *dist* (*R ncenter′*) (*R mbc′*) = *nradius dist* (*R ncenter′*) (*R mac′*) = *nradius dist* (*R ncenter′*) (*R mab′*) = *nradius*
using *R-dist* **by** *auto*
obtain *dist icenter′ pbc′* = *iradius dist icenter′ pac′* = *iradius dist icenter′ pab′* = *iradius*
by (*metis* (*no-types*, *opaque-lifting*) *add.commute diff-add-cancel dist-add-cancel icenter′-def ir1*
ir2 ir3 pab′-def pac′-def pbc′-def)
then show *dist* (*R icenter′*) (*R pbc′*) = *iradius dist* (*R icenter′*) (*R pac′*) = *iradius dist* (*R icenter′*) (*R pab′*) = *iradius*
using *R-dist* **by** *auto*
have *collinear* {*a′*, *b′*, *pab′*}
by (*metis a′-def b′-def col-ab collinear-translate diff-self pab′-def*)
then show *collinear* {*0*, (*R b′*), *R pab′*}
using *collinear-orthogonal-transformation*[*OF R-orth*, *of 0 b′ pab′*]
by (*simp add*: *a′-def R-0*)
have *orthogonal* (*a′* − *b′*) (*icenter′* − *pab′*)
by (*simp add*: *a′-def b′-def icenter′-def orth-ab pab′-def*)
then show *orthogonal* (*0* − (*R b′*)) ((*R icenter′*) − *R pab′*)
by (*simp add*: *a′-def R-inner inner-diff-right orthogonal-def*)
have *collinear* {*b′*, *c′*, *pbc′*}
using *b′-def c′-def col-bc collinear-translate pbc′-def* **by** *blast*
then show *collinear* {(*R b′*), (*R c′*), *R pbc′*}
using *collinear-orthogonal-transformation*[*OF R-orth*, *of b′ c′ pbc′*] **by** *simp*
have *orthogonal* (*b′* − *c′*) (*icenter′* − *pbc′*)
by (*simp add*: *b′-def c′-def icenter′-def orth-bc pbc′-def*)
then show *orthogonal* ((*R b′*) − (*R c′*)) ((*R icenter′*) − *R pbc′*)
by (*simp add*: *R-inner inner-diff-left inner-diff-right orthogonal-def*)
have *collinear* {*a′*, *c′*, *pac′*}
by (*metis a′-def c′-def col-ac collinear-translate diff-self pac′-def*)
then show *collinear* {*0*, (*R c′*), *R pac′*}
using *collinear-orthogonal-transformation*[*OF R-orth*, *of 0 c′ pac′*]
by (*simp add*: *a′-def R-0*)
have *orthogonal* (*a′* − *c′*) (*icenter′* − *pac′*)
by (*simp add*: *a′-def c′-def icenter′-def orth-ac pac′-def*)
then show *orthogonal* (*0* − (*R c′*)) ((*R icenter′*) − *R pac′*)
by (*simp add*: *a′-def R-inner inner-diff-right orthogonal-def*)
qed
have *special*: (*R ncenter′*) = (*R icenter′*) ∧ *nradius* = *iradius* ∨
(∃!*x*. *dist* (*R ncenter′*) *x* = *nradius* ∧ *dist* (*R icenter′*) *x* = *iradius*)
using *F0.special* **by** (*simp add*: *b′′-ne b′′-x c′′-y*)
— Step 4: Transfer conclusion back to original coordinates
show *?thesis*
proof (*cases R ncenter′* = *R icenter′* ∧ *nradius* = *iradius*)
case *True*
then show *?thesis*
by (*metis R-dist diff-add-cancel dist-eq-0-iff icenter′-def ncenter′-def*)
next
case *False*

with *special* **obtain** *x* **where**
x-uniq: *dist* (*R ncenter′*) *x* = *nradius* ∧ *dist* (*R icenter′*) *x* = *iradius*
⋀*y*. *dist* (*R ncenter′*) *y* = *nradius* ∧ *dist* (*R icenter′*) *y* = *iradius* ⟹ *y* = *x*
by *blast*
— Map x back through inverse transformations
have *inj-R*: *inj R*
using *R-orth orthogonal-transformation-inj* **by** *blast*
then obtain *Rinv* **where** *Rinv-lr*: ⋀*v*. *Rinv* (*R v*) = *v* **and** *Rinv-rl*: ⋀*v*. *R* (*Rinv v*) = *v*
by (*metis linear-injective-imp-surjective surjE injD R-lin*)
define *x′* **where** *x′* = *Rinv x* + *a*
have *Rx′*: *R* (*x′* − *a*) = *x* **by** (*simp add*: *x′-def Rinv-lr Rinv-rl*)
obtain *dist ncenter x′* = *nradius dist icenter x′* = *iradius*
using *x-uniq* **unfolding** *icenter′-def ncenter′-def*
by (*metis R-dist Rinv-rl diff-add-cancel dist-add-cancel2 x′-def*)
moreover have *y* = *x′* **if** *dist ncenter y* = *nradius dist icenter y* = *iradius* **for** *y*
proof −
have *dist* (*R ncenter′*) (*R* (*y* − *a*)) = *nradius dist* (*R icenter′*) (*R* (*y* − *a*)) = *iradius*
using *that R-dist* **by** (*auto simp add*: *ncenter′-def icenter′-def dist-norm*)
then show *y* = *x′*
using *Rinv-lr x′-def x-uniq*(*2*) **by** *force*
qed
ultimately show *?thesis* **by** *blast*
qed
qed

end

4 Nine-point circle

As a little bonus, verify that the circle passing through the midpoints of the sides is indeed a 9-point circle, i.e. it passes through the feet of the altitudes and the midpoints of the lines joining the vertices to the orthocenter (where the altitudes intersect).

lemma *nine-point-circle-1-scalar*:
fixes *bv cx cy n1 n2 f1 f2 g1 g2 h1 h2 nr* :: *real*
assumes *cy-ne*: *cy* ≠ *0* **and** *bv-ne*: *bv* ≠ *0*
— Nine-point center equidistant to midpoints of sides
— midpoint(0,b) = (bv/2, 0), midpoint(b,c) = ((bv+cx)/2, cy/2), midpoint(c,0) = (cx/2, cy/2)
and *nr1*: (*n1* − (*bv*+*cx*)/*2*)^*2* + (*n2* − *cy*/*2*)^*2* = *nr*^*2*
and *nr2*: (*n1* − *cx*/*2*)^*2* + (*n2* − *cy*/*2*)^*2* = *nr*^*2*
and *nr3*: (*n1* − *bv*/*2*)^*2* + *n2*^*2* = *nr*^*2*
— fab = (f1, f2): foot of altitude from c to ab (the x-axis)
— collinear 0, (bv,0), (f1,f2): bv * f2 = f1 * 0, so f2 = 0
and *fab-col*: *bv* ∗ *f2* = *0*

```
  — orthogonal (0-(bv,0)) ((cx,cy)-(f1,f2)): -bv*(cx-f1) + 0*(cy-f2) = 0
  and fab-orth: bv * (cx − f1) = 0
  — fbc = (g1, g2): foot of altitude from 0 to bc
  — collinear (bv,0), (cx,cy), (g1,g2): (cx-bv)*(g2-0) = (g1-bv)*(cy-0)
  and fbc-col: (cx − bv) * g2 = (g1 − bv) * cy
  — orthogonal ((bv,0)-(cx,cy)) ((0,0)-(g1,g2)): (bv-cx)*(-g1) + (-cy)*(-g2) = 0
  and fbc-orth: (bv − cx) * g1 = cy * g2
  — fac = (h1, h2): foot of altitude from b to ca
  — collinear (cx,cy), 0, (h1,h2): (-cx)*(h2-cy) = (h1-cx)*(-cy)
  and fac-col: cx * h2 = h1 * cy
  — orthogonal ((cx,cy)-0) ((bv,0)-(h1,h2)): cx*(bv-h1) + cy*(0-h2) = 0
  and fac-orth: cx * (bv − h1) = cy * h2
 shows (n1 − f1)^2 + (n2 − f2)^2 = nr^2 ∧ (n1 − g1)^2 + (n2 − g2)^2 =
nr^2 ∧ (n1 − h1)^2 + (n2 − h2)^2 = nr^2
 using assms by algebra

theorem nine-point-circle-1:
 fixes a b c :: real^2
  and mbc mac mab :: real^2
  and fbc fac fab :: real^2
  and ncenter :: real^2
  and nradius :: real
 assumes ¬ collinear {a, b, c}
  and midpoint a b = mab
  and midpoint b c = mbc
  and midpoint c a = mac
  and dist ncenter mbc = nradius
  and dist ncenter mac = nradius
  and dist ncenter mab = nradius
  and collinear {a, b, fab} orthogonal (a − b) (c − fab)
  and collinear {b, c, fbc} orthogonal (b − c) (a − fbc)
  and collinear {c, a, fac} orthogonal (c − a) (b − fac)
 shows dist ncenter fab = nradius ∧ dist ncenter fbc = nradius ∧ dist ncenter fac
= nradius
proof −
 have nr-pos: 0 ≤ nradius using assms zero-le-dist by metis
 define a1 a2 b1 b2 c1 c2 n1 n2 f1 f2 g1 g2 h1 h2
  where a1 = a$1 and a2 = a$2
   and b1 = b$1 and b2 = b$2
   and c1 = c$1 and c2 = c$2
   and n1 = ncenter$1 and n2 = ncenter$2
   and f1 = fab$1 and f2 = fab$2
   and g1 = fbc$1 and g2 = fbc$2
   and h1 = fac$1 and h2 = fac$2
 note defs = a1-def a2-def b1-def b2-def c1-def c2-def
       n1-def n2-def f1-def f2-def g1-def g2-def h1-def h2-def
 have ncol: (b1 − a1) * (c2 − a2) ≠ (c1 − a1) * (b2 − a2)
  using assms(1) collinear-3-2d by (auto simp: defs)
 have dist1: (n1 − (b1+c1)/2)^2 + (n2 − (b2+c2)/2)^2 = nradius^2
```

```
    using assms(3,5)[symmetric] by (simp add: midpoint-def dist-real2-sq defs)
  have dist2: (n1 − (c1+a1)/2)^2 + (n2 − (c2+a2)/2)^2 = nradius^2
    using assms(4,6)[symmetric] by (simp add: midpoint-def dist-real2-sq defs)
  have dist3: (n1 − (a1+b1)/2)^2 + (n2 − (a2+b2)/2)^2 = nradius^2
    using assms(2,7)[symmetric] by (simp add: midpoint-def dist-real2-sq defs)
  have col: (b1 − a1) ∗ (f2 − a2) = (f1 − a1) ∗ (b2 − a2)
          (c1 − b1) ∗ (g2 − b2) = (g1 − b1) ∗ (c2 − b2)
          (a1 − c1) ∗ (h2 − c2) = (h1 − c1) ∗ (a2 − c2)
    using assms collinear-3-2d by (auto simp: defs)+
  have orth: (a1 − b1) ∗ (c1 − f1) + (a2 − b2) ∗ (c2 − f2) = 0
           (b1 − c1) ∗ (a1 − g1) + (b2 − c2) ∗ (a2 − g2) = 0
           (c1 − a1) ∗ (b1 − h1) + (c2 − a2) ∗ (b2 − h2) = 0
    using assms
    by (simp-all add: orthogonal-def inner-real2 defs)
  have (n1 − f1)^2 + (n2 − f2)^2 = nradius^2 ∧
      (n1 − g1)^2 + (n2 − g2)^2 = nradius^2 ∧
      (n1 − h1)^2 + (n2 − h2)^2 = nradius^2
    using dist1 dist2 dist3 col orth ncol
    by algebra
  then show ?thesis
    using nr-pos
   by (simp add: dist-real2-sq f1-def f2-def g1-def g2-def h1-def h2-def n1-def n2-def
power2-eq-imp-eq)
qed


theorem nine-point-circle-2:
  fixes a b c :: real^2
    and mbc mac mab :: real^2
    and fbc fac fab :: real^2
    and ncenter oc :: real^2
    and nradius :: real
  assumes ¬ collinear {a, b, c}
    and midpoint a b = mab
    and midpoint b c = mbc
    and midpoint c a = mac
    and dist ncenter mbc = nradius
    and dist ncenter mac = nradius
    and dist ncenter mab = nradius
    and collinear {a, b, fab} orthogonal (a − b) (c − fab)
    and collinear {b, c, fbc} orthogonal (b − c) (a − fbc)
    and collinear {c, a, fac} orthogonal (c − a) (b − fac)
    and collinear {oc, a, fbc} collinear {oc, b, fac} collinear {oc, c, fab}
  shows dist ncenter (midpoint oc a) = nradius ∧
        dist ncenter (midpoint oc b) = nradius ∧
        dist ncenter (midpoint oc c) = nradius
proof −
  have nr-pos: 0 ≤ nradius using assms(5) zero-le-dist by metis
  define a1 a2 b1 b2 c1 c2 n1 n2 f1 f2 g1 g2 h1 h2 o1 o2
```

```
  where a1 = a$1 and a2 = a$2
    and b1 = b$1 and b2 = b$2
    and c1 = c$1 and c2 = c$2
    and n1 = ncenter$1 and n2 = ncenter$2
    and f1 = fab$1 and f2 = fab$2
    and g1 = fbc$1 and g2 = fbc$2
    and h1 = fac$1 and h2 = fac$2
    and o1 = oc$1 and o2 = oc$2
  note defs = a1-def a2-def b1-def b2-def c1-def c2-def
          n1-def n2-def f1-def f2-def g1-def g2-def h1-def h2-def o1-def o2-def
  have ncol: (b1 − a1) ∗ (c2 − a2) ≠ (c1 − a1) ∗ (b2 − a2)
    using assms(1) collinear-3-2d by (auto simp: defs)
  have dist1: (n1 − (b1+c1)/2)^2 + (n2 − (b2+c2)/2)^2 = nradius^2
    using assms(3,5)[symmetric] by (simp add: midpoint-def dist-real2-sq defs)
  have dist2: (n1 − (c1+a1)/2)^2 + (n2 − (c2+a2)/2)^2 = nradius^2
    using assms(4,6)[symmetric] by (simp add: midpoint-def dist-real2-sq defs)
  have dist3: (n1 − (a1+b1)/2)^2 + (n2 − (a2+b2)/2)^2 = nradius^2
    using assms(2,7)[symmetric] by (simp add: midpoint-def dist-real2-sq defs)
  have col: (b1 − a1) ∗ (f2 − a2) = (f1 − a1) ∗ (b2 − a2)
        (c1 − b1) ∗ (g2 − b2) = (g1 − b1) ∗ (c2 − b2)
        (a1 − c1) ∗ (h2 − c2) = (h1 − c1) ∗ (a2 − c2)
    using assms collinear-3-2d by (auto simp: defs)+
  have orth: (a1 − b1) ∗ (c1 − f1) + (a2 − b2) ∗ (c2 − f2) = 0
         (b1 − c1) ∗ (a1 − g1) + (b2 − c2) ∗ (a2 − g2) = 0
         (c1 − a1) ∗ (b1 − h1) + (c2 − a2) ∗ (b2 − h2) = 0
    using assms by (simp-all add: orthogonal-def inner-real2 defs)
  have oc-col: (a1 − o1) ∗ (g2 − o2) = (g1 − o1) ∗ (a2 − o2)
          (b1 − o1) ∗ (h2 − o2) = (h1 − o1) ∗ (b2 − o2)
          (c1 − o1) ∗ (f2 − o2) = (f1 − o1) ∗ (c2 − o2)
    using assms collinear-3-2d by (auto simp: defs)+
  — Derive direct orthogonality conditions for the orthocenter, eliminating the
altitude-foot variables from the final algebra calls.
  have oc-orth: (b1 − c1) ∗ (o1 − a1) + (b2 − c2) ∗ (o2 − a2) = 0
           (c1 − a1) ∗ (o1 − b1) + (c2 − a2) ∗ (o2 − b2) = 0
           (a1 − b1) ∗ (o1 − c1) + (a2 − b2) ∗ (o2 − c2) = 0
    using oc-col col orth ncol by algebra+
  have (n1 − (o1+a1)/2)^2 + (n2 − (o2+a2)/2)^2 = nradius^2
    using dist1 dist2 dist3 oc-orth ncol by algebra
  moreover have (n1 − (o1+b1)/2)^2 + (n2 − (o2+b2)/2)^2 = nradius^2
    using dist1 dist2 dist3 oc-orth ncol by algebra
  moreover have (n1 − (o1+c1)/2)^2 + (n2 − (o2+c2)/2)^2 = nradius^2
    using dist1 dist2 dist3 oc-orth ncol by algebra
  ultimately show ?thesis
    using nr-pos by (simp add: dist-real2-sq defs midpoint-def power2-eq-imp-eq)
qed

end
```