The Five Platonic Solids

Evan Finken

September 11, 2026

Abstract

This entry formalizes that there are exactly five Platonic solids: the tetrahedron, cube, octahedron, dodecahedron, and icosahedron. A Platonic solid is a convex polyhedron whose faces are congruent regular polygons, with the same number of edges meeting at each vertex. Euler's polyhedron formula is used to show that there are at most five Platonic solids, and each of the five, defined as the convex hull of its vertices, is then shown to satisfy the required properties. The formal proof is computational (slow!) and is ported from John Harrison's formalization in HOL Light [1]. Large language models were used to explain the intermediate steps of the HOL Light formalization and to write parts of the more complicated proofs.

Contents

1 Convexity **3**

2 Determinants **7**

3 Polytopes **8**
- 3.1 Invariance under isometries 8
- 3.2 Distinctness of facets 10
- 3.3 Ridges lie in two facets 12
- 3.4 Lower bounds on face counts 22
  - 3.4.1 Vertices 22
  - 3.4.2 Facets 23
- 3.5 Double counting 26
  - 3.5.1 $mF = 2E$ 27
  - 3.5.2 $nV = 2E$ 28
- 3.6 Face containment 30

4 Computational helpers **31**
- 4.1 Operations in $\mathbb{Q}(\sqrt{5})$ 31
- 4.2 Operations in $\mathbb{R}^3$ 35
- 4.3 Computation of faces 36

4.3.1 2-faces 36
4.3.2 1-faces 44
4.4 Full-dimensionality 53
4.5 Relations between faces of different dimensions 54
4.6 Congruence 56
4.6.1 Congruence of segments 57
4.6.2 Congruence of 2-faces 58
4.7 Equiangularity 71

5 Tetrahedron **74**
5.1 Definition 74
5.2 Facets, edges, and vertices 75
5.3 Regularity 78

6 Cube **80**
6.1 Definition 80
6.2 Facets, edges, and vertices 81
6.3 Regularity 86

7 Octahedron **88**
7.1 Definition 88
7.2 Facets, edges, and vertices 89
7.3 Regularity 94

8 Dodecahedron **96**
8.1 Definition 96
8.2 Facets, edges, and vertices 98
8.3 Regularity 108

9 Icosahedron **114**
9.1 Definition 114
9.2 Facets, edges, and vertices 115
9.3 Regularity 128

10 Main result **132**
10.1 Possible Schläfli symbols 132
10.2 Exactly five Platonic solids 135

1 Convexity

```
theory Convex-Euclidean-Space-More
  imports HOL−Analysis.Starlike
begin

lemma connected-Int-rel-frontier:
  assumes connected S
      and S ⊆ affine hull T
      and S ∩ T ≠ {}
      and S − T ≠ {}
    shows S ∩ rel-frontier T ≠ {}
proof
  assume ∗: S ∩ rel-frontier T = {}
  let ?E1 = S ∩ rel-interior T
  let ?E2 = S − closure T
  have openin (top-of-set S) ?E1
    by (meson assms(2) openin-Int openin-rel-interior
        openin-subtopology-Int-subset openin-subtopology-self)
  moreover have openin (top-of-set S) ?E2
    by (meson closed-closedin closed-closure closedin-self
        closedin-subtopology-refl openin-subtopology-diff-closed)
  moreover have S ⊆ ?E1 ∪ ?E2
    using ∗ rel-frontier-def by fastforce
  moreover have ?E1 ∩ ?E2 = {}
    using rel-interior-subset-closure by fastforce
  moreover have ?E1 ≠ {}
    using assms(3) calculation(3) closure-subset by fastforce
  moreover have ?E2 ≠ {}
    using assms(4) calculation(3) rel-interior-subset by fastforce
  ultimately show False
    using connected-openin[of S] assms(1) by blast
qed

end
theory Convex-More
  imports HOL−Analysis.Starlike
begin

lemma convex-hull-union-explicit:
  fixes S T :: 'a::euclidean-space set
  assumes convex S
      and convex T
    shows convex hull (S ∪ T) =
          S ∪ T ∪
          {(1 − u) ∗R x + u ∗R y | x y u. x ∈ S ∧ y ∈ T ∧ 0 ≤ u ∧ u ≤ 1}
  (is ?lhs = ?rhs)
proof
  show ?lhs ⊆ ?rhs
```

proof
fix y
assume $y \in$ *convex hull* $(S \cup T)$
then obtain $U\ f$ **where** *U-facts*:
finite U $U \subseteq (S \cup T)$ $(\forall x{\in}U.\ 0 \leq f\ x)$ *sum* $f\ U = 1$ $(\sum v{\in}U.\ f\ v *_R v) = y$
unfolding *convex-hull-explicit* **by** *auto*
have *U-partition*: $U = (U \cap S) \cup (U - S)$
by *auto*
have *disjoint*: $(U \cap S) \cap (U - S) = \{\}$
by *auto*
have *finite* $(U \cap S)$ *finite* $(U - S)$
using *U-facts*(1) **by** *auto*

let $?a =$ *sum* $f\ (U \cap S)$
let $?b =$ *sum* $f\ (U - S)$
have $?a + ?b = 1$
using *sum.union-disjoint*[*of* $U \cap S$ $U - S$ f] *U-facts*(1,4) *U-partition disjoint*
by *auto*
have *vsum-eq*: $y = (\sum v{\in}U \cap S.\ f\ v *_R v) + (\sum v{\in}U - S.\ f\ v *_R v)$
using *sum.union-disjoint*[*of* $U \cap S$ $U - S$ $\lambda v.\ f\ v *_R v$]
using *U-facts*(1,5) *U-partition disjoint*
by *auto*
have $0 \leq ?a$ $0 \leq ?b$
using *U-facts*(3)
by (*auto intro*: *sum-nonneg*)

consider
(*a-zero*) $?a = 0$ |
(*b-zero*) $?a \neq 0$ $?b = 0$ |
(*both-pos*) $?a \neq 0$ $?b \neq 0$
by *auto*
then show $y \in ?rhs$
proof *cases*
case *a-zero*
then have *zero-on-S*: $\forall x{\in}U \cap S.\ f\ x = 0$
using *U-facts*(3) *sum-nonneg-eq-0-iff*[*of* $U \cap S$ f]
by (*simp add*: ‹*finite* $(U \cap S)$›)
then have $y = (\sum v{\in}U - S.\ f\ v *_R v)$
using *vsum-eq* **by** *simp*
moreover have *sum* $f\ (U - S) = 1$
using ‹$?a + ?b = 1$› *a-zero*
by *simp*
moreover have $U - S \subseteq T$
using *U-facts*(2) **by** *auto*
moreover have $\forall x{\in}U - S.\ 0 \leq f\ x$
using *U-facts*(3) **by** *auto*
ultimately have $y \in T$
using *assms*(2) ‹*finite* $(U - S)$› **unfolding** *convex-explicit*
by *blast*

```
  then show ?thesis by auto
next
  case b-zero
  then have zero-on-T: ∀ x∈U − S. f x = 0
    using U-facts(3) sum-nonneg-eq-0-iff[of U − S f]
    by (simp add: ‹finite (U − S)›)
  then have y = (∑ v∈U ∩ S. f v *R v)
    using vsum-eq by simp
  moreover have sum f (U ∩ S) = 1
    using ‹?a + ?b = 1› b-zero
    by simp
  moreover have U ∩ S ⊆ S
    by auto
  moreover have ∀ x∈U ∩ S. 0 ≤ f x
    using U-facts(3) by auto
  ultimately have y ∈ S
    using assms(1) ‹finite (U ∩ S)› unfolding convex-explicit
    by blast
  then show ?thesis by auto
next
  case both-pos
  then have 0 < ?a 0 < ?b
    using ‹0 ≤ ?a› ‹0 ≤ ?b› by auto

  let ?x = ∑ v∈U ∩ S. (f v / ?a) *R v
  let ?y' = ∑ v∈U − S. (f v / ?b) *R v

  have sum (λv. f v / ?a) (U ∩ S) = 1
    using ‹0 < ?a›
    by (metis both-pos(1) div-self sum-divide-distrib)
  moreover have ∀ v∈U ∩ S. 0 ≤ f v / ?a
    using U-facts(3) ‹0 < ?a› by auto
  ultimately have ?x ∈ S
    using assms(1) ‹finite (U ∩ S)› unfolding convex-explicit
    by simp

  have sum (λv. f v / ?b) (U − S) = 1
    using ‹0 < ?b›
    by (metis both-pos(2) div-self sum-divide-distrib)
  moreover have ∀ v∈U − S. 0 ≤ f v / ?b
    using U-facts(3) ‹0 < ?b› by auto
  moreover have U − S ⊆ T
    using U-facts(2) by auto
  ultimately have ?y' ∈ T
    using assms(2) ‹finite (U − S)› unfolding convex-explicit
    by presburger

  have (∑ v∈U ∩ S. f v *R v) = ?a *R ?x
   by (smt (verit) both-pos(1) eq-vector-fraction-iff scaleR-right.sum sum.cong)
```

moreover have $(\sum v \in U - S. f\ v *_R v) = ?b *_R ?y'$
by $(smt\ (verit)\ both\text{-}pos(2)\ eq\text{-}vector\text{-}fraction\text{-}iff\ scaleR\text{-}right.sum\ sum.cong)$
ultimately have $y = (1 - ?b) *_R ?x + ?b *_R ?y'$
using *vsum-eq* ‹$?a + ?b = 1$› **by** *auto*

moreover have $?b \leq 1$
using ‹$0 \leq ?a$› ‹$0 \leq ?b$› ‹$?a + ?b = 1$›
by *auto*

ultimately show *?thesis*
using ‹$0 < ?b$› ‹$?x \in S$› ‹$?y' \in T$› **by** *auto*
qed
qed
next
have $S \cup T \subseteq convex\ hull\ (S \cup T)$
using *hull-subset* **by** *fast*
then show $?rhs \subseteq ?lhs$
by $(smt\ (verit,\ ccfv\text{-}threshold)\ assms(1,2)\ convex\text{-}hull\text{-}union\text{-}two\ empty\text{-}iff$ $le\text{-}sup\text{-}iff$
$mem\text{-}Collect\text{-}eq\ subsetI)$
qed

lemma *convex-hull-union-nonempty-explicit*:
fixes $S\ T :: 'a::euclidean\text{-}space\ set$
assumes *convex S*
and $S \neq \{\}$
and *convex T*
and $T \neq \{\}$
shows $convex\ hull\ (S \cup T) =$
$\{(1 - u) *_R x + u *_R y \mid x\ y\ u.\ x \in S \wedge y \in T \wedge 0 \leq u \wedge u \leq 1\}$
(**is** - = *?rhs*)
proof −
have $S \subseteq ?rhs$
using *assms*(4) **by** *fastforce*
moreover have $T \subseteq ?rhs$
by $(smt\ (verit,\ ccfv\text{-}threshold)\ add\text{-}0\ assms(2)\ equals0I\ mem\text{-}Collect\text{-}eq\ scaleR\text{-}one$ $scaleR\text{-}zero\text{-}left\ subset\text{-}iff)$
ultimately show *?thesis*
using *convex-hull-union-explicit*[*OF assms*(1,3)]
by *blast*
qed

lemma *convex-hull-isometry*:
assumes *orthogonal-transformation f*
shows $(\lambda x.\ c + f\ x)\ `\ (convex\ hull\ S) = convex\ hull\ ((\lambda x.\ c + f\ x)\ `\ S)$
using *orthogonal-transformation-linear*[*OF assms*] *convex-hull-linear-image*[*of f S*]
convex-hull-translation[*of c f ‘ S*]
by (*metis image-image*)

end

2 Determinants

```
theory Determinants-More
  imports HOL−Analysis.Starlike
begin

lemma hyperplane-subset-imp-scaled:
  fixes a :: 'a::euclidean-space
  assumes {x. a · x = b} ⊆ {x. a' · x = b'}
      and a ≠ 0
      and a' ≠ 0
  shows ∃ c. a' = c *R a ∧ b' = c * b ∧ c ≠ 0
proof −
  let ?c = (a · a') / (a · a)
  let ?v = a' − ?c *R a
  have a · ?v = 0
    by (simp add: inner-diff)
  let ?w = (b / (a · a)) *R a
  have a · ?w = b
    using assms(2) by simp
  then have a' · ?w = b'
    using assms(1) by blast
  have a · (?w + ?v) = b
    using ‹a · ?w = b› ‹a · ?v = 0›
    by (simp add: inner-add-right)
  then have a' · (?w + ?v) = b'
    using assms(1) by blast
  then have a' · ?v = 0
    using ‹a' · ?w = b'›
    by (simp add: inner-add-right)
  have ?v · ?v = 0
    using ‹a' · ?v = 0› ‹a · ?v = 0›
    by (simp add: inner-diff-left)
  then have ?v = 0
    by simp
  then have a' = ?c *R a
    by auto
  moreover have b' = ?c * b
   by (metis (mono-tags, lifting) assms(1) Collect-mono-iff ‹a' = ?c *R a› hyper-
plane-eq-Ex
        inner-scaleR-left assms(2))
  moreover have ?c ≠ 0
    using ‹a' = ?c *R a› assms(3) by fastforce
  ultimately show ?thesis
    by blast
qed
```

lemma *subset-hyperplanes*:
  fixes a :: $'a$::*euclidean-space*
  shows $(\{x.\ a \cdot x = b\} \subseteq \{x.\ a' \cdot x = b'\}) =$
      $(\{x.\ a \cdot x = b\} = \{\} \vee \{x.\ a' \cdot x = b'\} = UNIV \vee$
       $\{x.\ a \cdot x = b\} = \{x.\ a' \cdot x = b'\})$
  (**is** *?lhs* = *?rhs*)
proof
  assume ∗: *?lhs*
  consider
    (*a-zero*) $a = 0$ |
    (a'*-zero*) $a' = 0$ |
    (*nonzero*) $a \neq 0$ $a' \neq 0$
    by *blast*
  then show *?rhs*
  proof (*cases*)
    case *a-zero*
    then show *?thesis* **using** ∗ **by** *auto*
  next
    case a'*-zero*
    then show *?thesis* **using** ∗ **by** *force*
  next
    case *nonzero*
    then obtain c **where** ‹$a' = c *_R a$› ‹$b' = c * b$›
      using ∗ *hyperplane-subset-imp-scaled* **by** *blast*
    then show *?thesis*
      by (*smt* (*verit*, *ccfv-threshold*) *Collect-cong inner-scaleR-left*
          nonzero(*2*) *scaleR-eq-0-iff vector-space-over-itself.scale-left-imp-eq*)
  qed
next
  assume *?rhs*
  then show *?lhs* **by** *blast*
qed

end

3 Polytopes

theory *Polytope-More*
  imports
    Convex-Euclidean-Space-More
    HOL−Analysis.Analysis
begin

3.1 Invariance under isometries

lemma *facet-of-translation-eq*:
    ((+) a ‘ T *facet-of* (+) a ‘ S) = (T *facet-of* S)
  by (*simp add*: *aff-dim-translation-eq facet-of-def*)

lemma *facets-of-translation*:
shows $\{F.\ F$ *facet-of* $(+)\ a$ ‘ $S\} = (‘)\ ((+)\ a)$ ‘ $\{F.\ F$ *facet-of* $S\}$
proof −
have $\bigwedge F.\ F$ *facet-of* $(+)\ a$ ‘ $S \Longrightarrow \exists\, G.\ G$ *facet-of* $S \wedge F = (+)\ a$ ‘ G
by (*metis facet-of-imp-subset facet-of-translation-eq subset-imageE*)
then show *?thesis*
by (*auto simp*: *image-iff facet-of-translation-eq*)
qed

lemma *polyhedron-translation-eq*:
shows *polyhedron* S = *polyhedron* $((+)\ x$ ‘ $S)$
proof −
have *translation*: $\bigwedge S\ x.$ *polyhedron* $S \Longrightarrow$ *polyhedron* $((+)\ x$ ‘ $S)$
proof −
fix S::$'b$ *set* **and** x::$'b$
assume *polyhedron* S
then obtain F **where** *F-props*:
finite F
$S = \bigcap F$
$\bigwedge h.\ h \in F \Longrightarrow \exists\, a\ b.\ a \neq 0 \wedge h = \{y.\ a \cdot y \leq b\}$
unfolding *polyhedron-def* **by** *force*
let $?F' = ((‘)\ ((+)\ x))$ ‘ F
have *finite* $?F'$
using *F-props*(*1*) **by** *blast*
moreover have $((+)\ x$ ‘ $S) = \bigcap ?F'$
by (*metis* (*no-types*, *lifting*) *F-props*(*1*,*2*) *Inf-fin.hom-commute*
Inf-fin-Inf Inf-top-conv(*2*) *calculation empty-iff empty-is-image*
surj-plus translation-Int)
moreover have $\bigwedge h'.\ h' \in\ ?F' \Longrightarrow \exists\, a\ b.\ a \neq 0 \wedge h' = \{y.\ a \cdot y \leq b\}$
proof −
fix h'
assume $h' \in\ ?F'$
then obtain $h\ a\ b$ **where**
h'-*eq*: $h' = ((+)\ x)$ ‘ h **and**
h-props: $h \in F\ a \neq 0\ h = \{y.\ a \cdot y \leq b\}$
using *F-props*(*3*) **by** *blast*
have $h' = \{y + x \mid y.\ a \cdot y \leq b\}$
using h'-*eq h-props*(*3*) **by** *auto*
also have $\ldots = \{y.\ a \cdot (y - x) \leq b\}$
by *force*
finally have $h' = \{y.\ a \cdot y \leq b + a \cdot x\}$
by (*auto simp*: *inner-diff-right*)
then show $\exists\, a\ b.\ a \neq 0 \wedge h' = \{y.\ a \cdot y \leq b\}$
using *h-props*(*2*) **by** *blast*
qed
ultimately show *polyhedron* $((+)\ x$ ‘ $S)$
unfolding *polyhedron-def* **by** *meson*

qed
show *?thesis*
using *translation*[*of S x*] *translation*[*of* $((+)\ x$ ‘ $S)$ $-x$]
by (*metis translation-galois*)
qed

lemma *face-of-isometry-eq*:
assumes *orthogonal-transformation f*
shows $((\lambda x.\ c + f\ x)$ ‘ T *face-of* $(\lambda x.\ c + f\ x)$ ‘ $S) \longleftrightarrow T$ *face-of S*
by (*metis assms face-of-translation-eq face-of-linear-image image-image orthogonal-transformation-inj orthogonal-transformation-linear*)

lemma *extreme-point-of-isometry-eq*:
assumes *orthogonal-transformation f*
shows $(c + f\ T$ *extreme-point-of* $(\lambda x.\ c + f\ x)$ ‘ $S) \longleftrightarrow T$ *extreme-point-of S*
using *face-of-isometry-eq*[*OF assms*, *of c* $\{T\}$ *S*]
by (*simp add*: *face-of-singleton*)

lemma *aff-dim-isometry-eq*:
assumes *orthogonal-transformation f*
shows *aff-dim* $S =$ *aff-dim* $((\lambda x.\ c + f\ x)$ ‘ $S)$
by (*metis assms aff-dim-injective-linear-image aff-dim-translation-eq image-image orthogonal-transformation-def orthogonal-transformation-inj*)

3.2 Distinctness of facets

lemma *facets-of-polyhedron-explicit-distinct*:
fixes S :: *'a* :: *euclidean-space set*
assumes *finite*: *finite F*
and *seq*: $S =$ *affine hull* $S \cap \bigcap F$
and *faceq*: $\bigwedge h.\ h \in F \Longrightarrow a\ h \neq 0 \wedge h = \{x.\ a\ h \cdot x \leq b\ h\}$
and *psub*: $\bigwedge F'.\ F' \subset F \Longrightarrow S \subset$ *affine hull* $S \cap \bigcap F'$
and *h1-in*: $h1 \in F$
and *h2-in*: $h2 \in F$
and *eq*: $S \cap \{x.\ a\ h1 \cdot x = b\ h1\} = S \cap \{x.\ a\ h2 \cdot x = b\ h2\}$
shows $h1 = h2$
proof (*rule ccontr*)
assume $*$: $h1 \neq h2$
have $S \neq \{\}$
using *h1-in psub* **by** *force*
have *polyhedron S*
by (*meson polyhedron-Int-affine*[*of S*] *seq faceq finite*)
have *rel-interior* $S \neq \{\}$
using *rel-interior-polyhedron-explicit*
by (*simp add*: ‹$S \neq \{\}$› ‹*polyhedron S*› *polyhedron-imp-convex rel-interior-eq-empty*)
then obtain *z* **where** *z-props*: $z \in S$ $\bigwedge h.\ h \in F \Longrightarrow a\ h \cdot z < b\ h$
using *rel-interior-polyhedron-explicit*[*OF finite seq*] *faceq psub*
by *blast*

have $S \subset$ *affine hull* $S \cap \bigcap(F - \{h2\})$
  using *h2-in psub*[*of* $F - \{h2\}$]
  by *blast*
then obtain x **where** *x-props*: $x \in$ *affine hull* S $x \in \bigcap(F - \{h2\})$ $x \notin S$
  by *auto*
then have $a\ h2 \cdot x > b\ h2$
  using *Diff-iff empty-iff faceq insert-iff seq*
  by *fastforce*
have $\bigwedge h.\ h \in (F - \{h2\}) \Longrightarrow a\ h \cdot x \leq b\ h$
  using *x-props*(*2*) *faceq* **by** *auto*
have *closed-segment* x z $\cap$ *rel-frontier* $S \neq \{\}$
  by (*metis Diff-iff Int-iff closed-segment-subset connected-Int-rel-frontier*
    connected-segment convex-affine-hull empty-iff ends-in-segment(*1*,*2*) *seq*
    x-props(*1*,*3*) *z-props*(*1*))
then have *closed-segment* x z $\cap$ (S $-$ *rel-interior* S) $\neq \{\}$
  by (*simp add*: ‹*polyhedron* S› *rel-boundary-of-polyhedron*
    rel-frontier-of-polyhedron)
then obtain y **where** *y-props*:
  $y \in$ *closed-segment* x z
  $y \in S$ $-$ *rel-interior* S
  $y \in S$
  by *blast*
have S $-$ *rel-interior* $S = \bigcup\{S \cap \{x.\ a\ h \cdot x = b\ h\} \mid h.\ h \in F\}$
  using *facet-of-polyhedron-explicit*[*OF finite seq faceq psub*]
    rel-boundary-of-polyhedron[*OF* ‹*polyhedron* S›]
  by *auto*
then obtain h **where** *h-props*: $h \in F$ $a\ h \cdot y = b\ h$
  using *y-props*(*2*) **by** *auto*
have $\exists k.\ k \in F \wedge k \neq h2 \wedge a\ k \cdot y = b\ k$
proof (*cases* $h = h2$)
  case *True*
  then have $y \in S \cap \{x.\ a\ h2 \cdot x = b\ h2\}$
    using *h-props*(*2*) *y-props*(*3*) **by** *blast*
  then have $a\ h1 \cdot y = b\ h1$
    using *eq* **by** *blast*
  then show *?thesis*
    using *h1-in* $*$ **by** *blast*
next
  case *False*
  then show *?thesis*
    using *h-props* **by** *blast*
qed
then obtain k **where** *k-props*: $k \in F$ $k \neq h2$ $a\ k \cdot y = b\ k$
  by *blast*
have $y \in$ *open-segment* x z
  by (*metis DiffI* ‹$\exists k.\ k \in F \wedge k \neq h2 \wedge a\ k \cdot y = b\ k$›
    all-not-in-conv insertE open-segment-def order-less-irrefl
    x-props(*3*) *y-props*(*1*,*3*) *z-props*(*2*))
then obtain u **where** *u-props*: $y = (1 - u) *_R x + u *_R z$ $0 < u$ $u < 1$

using *in-segment*(*2*)[*of y x z*] **by** *blast*
have $(1 - u) * (a\ k \cdot x) + u * (a\ k \cdot z) = b\ k$
using *k-props*(*3*) **unfolding** *u-props*(*1*)
by (*simp add*: *inner-right-distrib*)
moreover have $(1 - u) * (a\ k \cdot x) \leq (1 - u) * b\ k$
by (*simp add*: ‹$\bigwedge ha.\ ha \in F - \{h2\} \Longrightarrow a\ ha \cdot x \leq b\ ha$› *k-props*(*1*,*2*) *u-props*(*3*))
moreover have $u * (a\ k \cdot z) < u * b\ k$
by (*simp add*: *k-props*(*1*) *u-props*(*2*) *z-props*(*2*))
ultimately show *False*
by *argo*
qed

3.3 Ridges lie in two facets

lemma *polyhedron-ridge-two-facets-0*:
fixes *p* :: *'a* :: *euclidean-space set*
assumes *polyhedron p*
and *r face-of p*
and $0 \in r$
and *aff-dim r = aff-dim p − 2*
shows $\exists f1\ f2.$ *f1 face-of p* $\wedge$ *aff-dim f1 = aff-dim p − 1* $\wedge$
f2 face-of p $\wedge$ *aff-dim f2 = aff-dim p − 1* $\wedge$
$f1 \neq f2 \wedge r \subseteq f1 \wedge r \subseteq f2 \wedge f1 \cap f2 = r \wedge$
$(\forall f.$ *f face-of p* $\wedge$ *aff-dim f = aff-dim p − 1* $\wedge r \subseteq f$
$\longrightarrow f = f1 \vee f = f2)$
proof −
let *?S* = {*f*. *f face-of p* $\wedge$ *aff-dim f = aff-dim p − 1* $\wedge r \subseteq f$}
have *finite ?S*
by (*simp add*: *assms*(*1*) *finite-polyhedron-faces*)
have $r = \bigcap$ {*f*. *f facet-of p* $\wedge r \subseteq f$}
using *face-of-polyhedron*[*of p r*] *assms* **by** *fastforce*
then have *r-eq*: $r = \bigcap$ *?S*
using *assms*(*3*) *facet-of-def* **by** *blast*
have $0 \in p$
using *assms*(*2*,*3*) *face-of-imp-subset* **by** *auto*

have *card ?S* = 0 $\vee$ *card ?S* = 1 $\vee$ *card ?S* = 2 $\vee$ *card ?S* $\geq$ 3
by *linarith*
then consider
(*card0*) *card ?S* = 0 |
(*card1*) *card ?S* = 1 |
(*card2*) *card ?S* = 2 |
(*card3*) *card ?S* $\geq$ 3
by *argo*

then show *?thesis*
proof (*cases*)
case *card0*

then show *?thesis*
  by (*smt* (*verit*, *best*) *Inf-empty* ‹*finite ?S*› *aff-dim-UNIV*
    aff-dim-le-DIM assms(*4*) *card-gt-0-iff nless-le r-eq*)

next
  case *card1*
  then obtain *f1* **where**
    S-eq: *?S* = {*f1*} **and**
    aff-dim-f1: *aff-dim f1* = *aff-dim p* − *1*
    by (*metis* (*mono-tags*, *lifting*) *all-not-in-conv card-1-singletonE*
      insert-not-empty mem-Collect-eq singletonD)
  show *?thesis*
    using *r-eq aff-dim-f1 assms*(*4*) **unfolding** *S-eq*
    by *simp*

next
  case *card2*
  then obtain *f1 f2* **where**
    S-eq: *?S* = {*f1*, *f2*} **and**
    f1 face-of p aff-dim f1 = *aff-dim p* − *1 r* ⊆ *f1* **and**
    f2 face-of p aff-dim f2 = *aff-dim p* − *1 r* ⊆ *f2* **and**
    f1 ≠ *f2*
    by (*smt* (*verit*) *card-2-iff insert-iff mem-Collect-eq*)
  moreover have *f1* ∩ *f2* = *r*
    using *S-eq r-eq* **by** *auto*
  moreover have (∀*f*. *f face-of p* ∧ *aff-dim f* = *aff-dim p* − *1* ∧ *r* ⊆ *f*
      ⟶ *f* = *f1* ∨ *f* = *f2*)
    using *S-eq* **by** *auto*
  ultimately show *?thesis*
    by *blast*

next
  case *card3*
  then have *card ?S* ≥ *Suc* (*Suc* (*Suc 0*))
    by *presburger*
  then obtain *f1 f2 f3 S2* **where**
    S-eq: *?S* = *insert f3* (*insert f2* (*insert f1 S2*)) **and**
    fs-neq: *f1* ≠ *f2 f1* ≠ *f3 f2* ≠ *f3*
    by (*smt* (*verit*) *card-le-Suc-iff insertCI*)
  have *f1-props*: *f1 face-of p aff-dim f1* = *aff-dim p* − *1 r* ⊆ *f1* **and**
    f2-props: *f2 face-of p aff-dim f2* = *aff-dim p* − *1 r* ⊆ *f2* **and**
    f3-props: *f3 face-of p aff-dim f3* = *aff-dim p* − *1 r* ⊆ *f3*
    using *S-eq*
    by *auto*

  obtain *F a b* **where**
    finite F **and**
    p-eq: *p* = *affine hull p* ∩ ⋂ *F* **and**
    halfspaces: ⋀*h*. *h*∈*F* ⟹ *a h* ≠ *0* ∧ *h* = {*x*. *a h* · *x* ≤ *b h*} ∧

$(\forall\, x \in \mathit{affine\ hull\ p}.\ x + a\ h \in \mathit{affine\ hull\ p})$ **and**
minimal: $\bigwedge F'.\ F' \subset F \longrightarrow p \subset \mathit{affine\ hull}\ p \cap \bigcap F'$
using *polyhedron-Int-affine-parallel-minimal*[*of p*] *assms*(*1*)
by *metis*
obtain *h1* **where** $h1 \in F$ $a\ h1 \neq 0$ **and** *f1-eq*: $f1 = p \cap \{x.\ a\ h1 \cdot x = b\ h1\}$
using *facet-of-polyhedron-explicit*[*OF* ‹*finite F*› *p-eq*, *of a b f1*]
halfspaces minimal f1-props **unfolding** *facet-of-def*
using *assms*(*3*) **by** *blast*
obtain *h2* **where** $h2 \in F$ $a\ h2 \neq 0$ **and** *f2-eq*: $f2 = p \cap \{x.\ a\ h2 \cdot x = b\ h2\}$
using *facet-of-polyhedron-explicit*[*OF* ‹*finite F*› *p-eq*, *of a b f2*]
halfspaces minimal f2-props **unfolding** *facet-of-def*
using *assms*(*3*) **by** *blast*
obtain *h3* **where** $h3 \in F$ $a\ h3 \neq 0$ **and** *f3-eq*: $f3 = p \cap \{x.\ a\ h3 \cdot x = b\ h3\}$
using *facet-of-polyhedron-explicit*[*OF* ‹*finite F*› *p-eq*, *of a b f3*]
halfspaces minimal f3-props **unfolding** *facet-of-def*
using *assms*(*3*) **by** *blast*

have *all-x-in-r*: $\bigwedge x.\ x \in r \Longrightarrow a\ h1 \cdot x = b\ h1 \wedge a\ h2 \cdot x = b\ h2 \wedge a\ h3 \cdot x = b\ h3$
using *f1-props*(*3*) *f2-props*(*3*) *f3-props*(*3*) *f1-eq f2-eq f3-eq*
by *blast*
then have *b-hs-eq-0*: $b\ h1 = 0$ $b\ h2 = 0$ $b\ h3 = 0$
using ‹$0 \in r$› **by** *force+*
then have *all-x-in-r-0*: $\bigwedge x.\ x \in r \Longrightarrow a\ h1 \cdot x = 0 \wedge a\ h2 \cdot x = 0 \wedge a\ h3 \cdot x = 0$
using *all-x-in-r* **by** *auto*
have *a-hs-neq*: $a\ h1 \neq a\ h2$ $a\ h1 \neq a\ h3$ $a\ h2 \neq a\ h3$
using *b-hs-eq-0 f1-eq f2-eq f3-eq fs-neq* **by** *force+*
have *hs-neq*: $h1 \neq h2$ $h1 \neq h3$ $h2 \neq h3$
using *fs-neq f1-eq f2-eq f3-eq* **by** *auto*

have *aff-dim* $r = \mathit{dim}\ r$
using *aff-dim-zero*[*OF hull-inc*[*OF* ‹$0 \in r$›]] **.**
moreover have *aff-dim* $p = \mathit{dim}\ p$
using *aff-dim-zero*[*OF hull-inc*[*OF* ‹$0 \in p$›]] **.**
ultimately have $\mathit{dim}\ r = \mathit{dim}\ p - 2$
using *assms*(*4*) **by** *presburger*

have *not-contained-xy-z*:
$\neg (\{x.\ a\ hx \cdot x \le 0\} \cap \{x.\ a\ hy \cdot x \le 0\} \subseteq \{x.\ a\ hz \cdot x \le 0\})$
if $hx \in F$ $hy \in F$ $hz \in F$ $b\ hx = 0$ $b\ hy = 0$ $b\ hz = 0$ $hx \neq hz$ $hy \neq hz$ **for** *hx hy hz*
proof
assume $\{x.\ a\ hx \cdot x \le 0\} \cap \{x.\ a\ hy \cdot x \le 0\} \subseteq \{x.\ a\ hz \cdot x \le 0\}$
then have $*$: $hx \cap hy \subseteq hz$
using *that halfspaces*
by *metis*
have $F - \{hz\} \subset F$
using ‹$hz \in F$› ‹*finite F*› **by** *blast*

then have *p-psubset*: $p \subset \mathit{affine\ hull}\ p \cap \bigcap(F - \{hz\})$
using *minimal* **by** *presburger*
have $hx \in F - \{hz\}$ $hy \in F - \{hz\}$
using *that* **by** *auto*
then show *False*
using $*$ *p-eq* *p-psubset* **by** *blast*
qed
have *not-contained-12-3*:
$\neg\ (\{x.\ a\ h1 \cdot x \le 0\} \cap \{x.\ a\ h2 \cdot x \le 0\} \subseteq \{x.\ a\ h3 \cdot x \le 0\})$
using *not-contained-xy-z*[*OF* ‹$h1 \in F$› ‹$h2 \in F$› ‹$h3 \in F$›] *b-hs-eq-0* *hs-neq*
by *blast*
have *not-contained-13-2*:
$\neg\ (\{x.\ a\ h1 \cdot x \le 0\} \cap \{x.\ a\ h3 \cdot x \le 0\} \subseteq \{x.\ a\ h2 \cdot x \le 0\})$
using *not-contained-xy-z*[*OF* ‹$h1 \in F$› ‹$h3 \in F$› ‹$h2 \in F$›] *b-hs-eq-0* *hs-neq*
by *blast*
have *not-contained-23-1*:
$\neg\ (\{x.\ a\ h2 \cdot x \le 0\} \cap \{x.\ a\ h3 \cdot x \le 0\} \subseteq \{x.\ a\ h1 \cdot x \le 0\})$
using *not-contained-xy-z*[*OF* ‹$h2 \in F$› ‹$h3 \in F$› ‹$h1 \in F$›] *b-hs-eq-0* *hs-neq*
by *blast*

obtain w **where** $w \in \mathit{rel\text{-}interior}\ p$
using ‹$0 \in p$› *polyhedron-imp-convex* *rel-interior-eq-empty* *assms*(*1*)
by *auto*
moreover have $\mathit{rel\text{-}interior}\ p = \{x \in p.\ \forall h{\in}F.\ a\ h \cdot x < b\ h\}$
using *rel-interior-polyhedron-explicit*[*OF* ‹*finite F*› *p-eq*, *of a b*]
halfspaces minimal
by *blast*
ultimately have *w-props*: $a\ h1 \cdot w < 0$ $a\ h2 \cdot w < 0$ $a\ h3 \cdot w < 0$
using *b-hs-eq-0* ‹$h1 \in F$› ‹$h2 \in F$› ‹$h3 \in F$›
by *auto*

have $r \subseteq \mathit{span}\ p$
using *face-of-imp-subset*[*OF assms*(*2*)] *span-superset*[*of p*]
by *blast*
have $a\ h1 \in \mathit{span}\ p$ $a\ h2 \in \mathit{span}\ p$ $a\ h3 \in \mathit{span}\ p$
using *halfspaces* ‹$h1 \in F$› ‹$h2 \in F$› ‹$h3 \in F$› ‹$0 \in p$›
by (*metis add-0 affine-hull-span-0 hull-inc*)+
then have $\mathit{span}\ \{a\ h1,\ a\ h2,\ a\ h3\} \subseteq \mathit{span}\ p$
using *subspace-span*[*of p*]
by (*simp add*: *span-minimal*)
then have $r \cup \mathit{span}\ \{a\ h1,\ a\ h2,\ a\ h3\} \subseteq \mathit{span}\ p$
using ‹$r \subseteq \mathit{span}\ p$› *subspace-span*[*of p*]
by (*simp add*: *subspace-sum-minimal*)
have $\dim\ (\mathit{span}\ p) = \dim\ p$
by *simp*
have $(\bigwedge x\ y.\ x \in r \Longrightarrow y \in \mathit{span}\ \{a\ h1,\ a\ h2,\ a\ h3\} \Longrightarrow x \cdot y = 0)$
proof $-$
fix $x\ y$
assume *x-in-r*: $x \in r$

```
      assume y ∈ span {a h1, a h2, a h3}
      then show x · y = 0
      proof (induct y rule: span-induct)
        case base
        then show ?case
          using subspace-hyperplane by blast
      next
        case (step y)
        then show ?case
          using all-x-in-r-0[OF x-in-r]
          by (metis inner-commute empty-iff insertE)
      qed
    qed
    then have dim (r ∪ span {a h1, a h2, a h3}) = dim r + dim (span {a h1, a
h2, a h3})
      using dim-orthogonal-sum[of r span {a h1, a h2, a h3}]
      by blast
    moreover have dim (r ∪ span {a h1, a h2, a h3}) ≤ dim p
      using dim-subset[OF ‹r ∪ span {a h1, a h2, a h3} ⊆ span p›]
      by auto
    ultimately have dim (span {a h1, a h2, a h3}) ≤ 2
      using ‹dim r = dim p − 2›
      by linarith

    have independent {a h1, a h2, a h3}
    proof
      assume dependent {a h1, a h2, a h3}
      have finite {a h1, a h2, a h3} by simp
      obtain c where c-sum: (∑ v∈{a h1, a h2, a h3}. c v *R v) = 0
               and nonzero: ∃ v∈{a h1, a h2, a h3}. c v ≠ 0
        using ‹dependent {a h1, a h2, a h3}› dependent-finite[OF ‹finite {a h1, a
h2, a h3}›]
        by auto

      let ?c1 = c (a h1) and ?c2 = c (a h2) and ?c3 = c (a h3)
      have sum-0: ?c1 *R a h1 + ?c2 *R a h2 + ?c3 *R a h3 = 0
        using c-sum a-hs-neq
        by (simp add: group-cancel.add1)
      have ?c1 ≠ 0
      proof
        assume ?c1 = 0
        then have ?c2 *R a h2 + ?c3 *R a h3 = 0 using sum-0 by simp
        then have ?c2 ≠ 0 ?c3 ≠ 0
          using nonzero ‹?c1 = 0›
          using ‹a h2 ≠ 0› ‹a h3 ≠ 0› by auto

        define k where k = − (?c3 / ?c2)
        have a h2 = k *R a h3
          using ‹?c2 *R a h2 + ?c3 *R a h3 = 0› ‹?c2 ≠ 0›
```

unfolding *k-def*
by (*smt* (*verit*, *ccfv-SIG*) *eq-vector-fraction-iff neg-eq-iff-add-eq-0 scaleR-left.minus scaleR-minus-right*)

show *False*
proof (*cases* $k > 0$)
case *True*
then have $\{x.\ a\ h3 \cdot x \leq 0\} = \{x.\ a\ h2 \cdot x \leq 0\}$
unfolding ‹$a\ h2 = k *_R a\ h3$› **by** (*auto simp*: *mult-le-0-iff*)
then have $\{x.\ a\ h1 \cdot x \leq 0\} \cap \{x.\ a\ h2 \cdot x \leq 0\} \subseteq \{x.\ a\ h3 \cdot x \leq 0\}$
by *blast*
then show *False* **using** *not-contained-12-3* **by** *simp*
next
case *False*
then have $k < 0$
using ‹$a\ h2 = k *_R a\ h3$› *a-hs-neq*(*3*) ‹$a\ h2 \neq 0$› **by** *fastforce*
have $a\ h2 \cdot w = k * (a\ h3 \cdot w)$ **using** ‹$a\ h2 = k *_R a\ h3$› ‹$a\ h2 \neq 0$› **by** *auto*
moreover have $a\ h2 \cdot w < 0$ **using** *w-props*(*2*) **by** *simp*
moreover have $k * (a\ h3 \cdot w) > 0$ **using** ‹$k < 0$› *w-props*(*3*) *mult-neg-neg* **by** *blast*
ultimately show *False* **by** *simp*
qed
qed

define u **where** $u = -(?c2\ /\ ?c1)$
define v **where** $v = -(?c3\ /\ ?c1)$
have $?c1 *_R a\ h1 = - ?c2 *_R a\ h2 - ?c3 *_R a\ h3$
by (*metis* (*no-types*, *lifting*) *sum-0 add.left-commute add.right-neutral add-minus-cancel scaleR-left.minus*
uminus-add-conv-diff)
then have $(1/?c1) *_R (?c1 *_R a\ h1) = (1/?c1) *_R (-?c2 *_R a\ h2 - ?c3 *_R a\ h3)$
by *auto*
then have *a1-eq*: $a\ h1 = u *_R a\ h2 + v *_R a\ h3$
using ‹$?c1 \neq 0$› **unfolding** *u-def v-def*
by (*simp add*: *scaleR-right-diff-distrib*)

have $u \neq 0$
proof
assume $u = 0$
then have $a\ h1 = v *_R a\ h3$
using *a1-eq* **by** *simp*
show *False*
proof (*cases* $v > 0$)
case *True*
then have $\{x.\ a\ h3 \cdot x \leq 0\} = \{x.\ a\ h1 \cdot x \leq 0\}$
unfolding ‹$a\ h1 = v *_R a\ h3$›
by (*auto simp*: *mult-le-0-iff*)

```
    then have {x. a h2 · x ≤ 0} ∩ {x. a h3 · x ≤ 0} ⊆ {x. a h1 · x ≤ 0}
      by blast
    then show False
      using not-contained-23-1
      by simp
  next
    case False
    then have v < 0
      using ‹a h1 = v *R a h3› a-hs-neq(2) ‹a h1 ≠ 0›
      by auto
    have a h1 · w = v * (a h3 · w)
      using ‹a h1 = v *R a h3›
      by simp
    moreover have a h1 · w < 0
      using w-props(1) by simp
    moreover have v * (a h3 · w) > 0
      using ‹v < 0› w-props(3) mult-neg-neg
      by blast
    ultimately show False by simp
  qed
qed

have v ≠ 0
proof
  assume v = 0
  then have a h1 = u *R a h2
    using a1-eq by simp
  show False
  proof (cases u > 0)
    case True
    then have {x. a h2 · x ≤ 0} = {x. a h1 · x ≤ 0}
      unfolding ‹a h1 = u *R a h2›
      by (auto simp: mult-le-0-iff)
    then have {x. a h2 · x ≤ 0} ∩ {x. a h3 · x ≤ 0} ⊆ {x. a h1 · x ≤ 0}
      by blast
    then show False
      using not-contained-23-1
      by simp
  next
    case False
    then have u < 0
      using ‹a h1 = u *R a h2› a-hs-neq(1) ‹a h1 ≠ 0›
      by auto
    have a h1 · w = u * (a h2 · w)
      using ‹a h1 = u *R a h2›
      by simp
    moreover have a h1 · w < 0
      using w-props(1) by simp
    moreover have u * (a h2 · w) > 0
```

```
      using ‹u < 0› w-props(2) mult-neg-neg
      by blast
    ultimately show False by simp
  qed
qed

consider (pos-pos) u > 0 v > 0 |
        (pos-neg) u > 0 v < 0 |
        (neg-pos) u < 0 v > 0 |
        (neg-neg) u < 0 v < 0
  using ‹u ≠ 0› ‹v ≠ 0› by linarith
then show False
proof cases
  case pos-pos
  have {x. a h2 · x ≤ 0} ∩ {x. a h3 · x ≤ 0} ⊆ {x. a h1 · x ≤ 0}
  proof
    fix x
    assume x-in: x ∈ {x. a h2 · x ≤ 0} ∩ {x. a h3 · x ≤ 0}
    have a h1 · x = u ∗ (a h2 · x) + v ∗ (a h3 · x)
      using a1-eq by (simp add: inner-add)
    also have ... ≤ 0
      using x-in pos-pos
      by (simp add: add-decreasing2 split-mult-neg-le)
    finally show x ∈ {x. a h1 · x ≤ 0}
      by blast
  qed
  then show False
    using not-contained-23-1 by simp
next
  case pos-neg
  have u ∗R a h2 = a h1 + (−v) ∗R a h3
    using a1-eq
    by (simp add: algebra-simps)
  have {x. a h1 · x ≤ 0} ∩ {x. a h3 · x ≤ 0} ⊆ {x. a h2 · x ≤ 0}
  proof
    fix x assume x-in: x ∈ {x. a h1 · x ≤ 0} ∩ {x. a h3 · x ≤ 0}
    have u ∗ (a h2 · x) = (a h1 · x) + (−v) ∗ (a h3 · x)
      using ‹u ∗R a h2 = a h1 + (−v) ∗R a h3›
      by (metis inner-left-distrib inner-scaleR-left)
    also have ... ≤ 0
      using x-in pos-neg
      by (smt (verit, best) IntE mem-Collect-eq mult-le-0-iff)
    finally have u ∗ (a h2 · x) ≤ 0 .
    then show x ∈ {x. a h2 · x ≤ 0}
      using ‹u > 0›
      by (simp add: mult-le-0-iff)
  qed
  then show False
    using not-contained-13-2
```

```
        by simp
      next
        case neg-pos
        have v *R a h3 = a h1 + (−u) *R a h2
          using a1-eq
          by (simp add: algebra-simps)
        have {x. a h1 · x ≤ 0} ∩ {x. a h2 · x ≤ 0} ⊆ {x. a h3 · x ≤ 0}
        proof
          fix x
          assume x-in: x ∈ {x. a h1 · x ≤ 0} ∩ {x. a h2 · x ≤ 0}
          have v * (a h3 · x) = (a h1 · x) + (−u) * (a h2 · x)
            using ‹v *R a h3 = a h1 + (−u) *R a h2›
            by (metis inner-left-distrib inner-scaleR-left)
          also have ... ≤ 0
            using x-in neg-pos
            by (smt (verit, best) IntE mem-Collect-eq mult-le-0-iff)
          finally show x ∈ {x. a h3 · x ≤ 0}
            using ‹v > 0›
            by (simp add: mult-le-0-iff)
        qed
        then show False
          using not-contained-12-3
          by simp
      next
        case neg-neg
        have a h1 · w = u * (a h2 · w) + v * (a h3 · w)
          using a1-eq
          by (simp add: inner-add)
        moreover have u * (a h2 · w) > 0 v * (a h3 · w) > 0
          using neg-neg w-props(2,3) mult-neg-neg
          by blast+
        ultimately have a h1 · w > 0
          by simp
        then show False
          using w-props(1) by simp
      qed
    qed

    then have dim (span {a h1, a h2, a h3}) = 3
      using a-hs-neq(1,2,3) dim-span-eq-card-independent
      by fastforce
    then show ?thesis
      using ‹dim (span {a h1, a h2, a h3}) ≤ 2›
      by linarith
  qed
qed

lemma polyhedron-ridge-two-facets:
  fixes p :: 'a :: euclidean-space set
```

```
  assumes polyhedron p
    and r face-of p
    and r ≠ {}
    and aff-dim r = aff-dim p − 2
  shows ∃f1 f2. f1 face-of p ∧ aff-dim f1 = aff-dim p − 1 ∧
               f2 face-of p ∧ aff-dim f2 = aff-dim p − 1 ∧
               f1 ≠ f2 ∧ r ⊆ f1 ∧ r ⊆ f2 ∧ r = f1 ∩ f2 ∧
               (∀f. f face-of p ∧ aff-dim f = aff-dim p − 1 ∧ r ⊆ f
                 ⟶ f = f1 ∨ f = f2)
proof −
  obtain z where z ∈ r
    using assms(3) by blast
  then have z ∈ p
    using assms(2) face-of-imp-subset by auto
  let ?r' = ((+) (−z)) ‘ r
  let ?p' = ((+) (−z)) ‘ p
  have origin-r': 0 ∈ ?r'
    using ‹z ∈ r› by simp
  have origin-p': 0 ∈ ?p'
    using ‹z ∈ p› by simp
  have polyhedron: polyhedron ?p'
    using assms(1) polyhedron-translation-eq[of p −z] by blast
  have face-of: ?r' face-of ?p'
    using assms(2) face-of-translation-eq by blast
  have nonempty: ?r' ≠ {}
    using assms(3) by blast
  have aff-dims: aff-dim ?r' = aff-dim ?p' − 2
    by (metis aff-dim-translation-eq assms(4))
  obtain f1 f2 where
    f1-face-of: f1 face-of ?p' and
    aff-dim-f1: aff-dim f1 = aff-dim ?p' − 1 and
    f2-face-of: f2 face-of ?p' and
    aff-dim-f2: aff-dim f2 = aff-dim ?p' − 1 and
    fs-neq: f1 ≠ f2 and
    sub-f1: ?r' ⊆ f1 and
    sub-f2: ?r' ⊆ f2 and
    fs-inter-eq: f1 ∩ f2 = ?r' and
    only-fs: (∀f. f face-of ?p' ∧
                  aff-dim f = aff-dim ?p' − 1 ∧ ?r' ⊆ f ⟶
                  f = f1 ∨ f = f2)
    using polyhedron-ridge-two-facets-0[OF polyhedron face-of origin-r' aff-dims]
    by auto
  let ?f1' = ((+) z) ‘ f1
  let ?f2' = ((+) z) ‘ f2
  have ?f1' face-of p
    by (metis f1-face-of face-of-translation-eq translation-galois)
  moreover have aff-dim ?f1' = aff-dim p − 1
    by (metis aff-dim-f1 aff-dim-translation-eq)
  moreover have ?f2' face-of p
```

by (*metis f2-face-of face-of-translation-eq translation-galois*)
moreover have *aff-dim ?f2'* = *aff-dim p* − *1*
by (*metis aff-dim-f2 aff-dim-translation-eq*)
moreover have *?f1'* ≠ *?f2'*
using *fs-neq translation-invert*
by *auto*
moreover have *r* ⊆ *?f1'*
by (*metis sub-f1 subset-image-iff translation-galois*)
moreover have *r* ⊆ *?f2'*
by (*metis sub-f2 subset-image-iff translation-galois*)
moreover have *?f1'* ∩ *?f2'* = *r*
by (*metis fs-inter-eq translation-Int translation-galois*)
moreover have (∀ *f*. *f face-of p* ∧
aff-dim f = *aff-dim p* − *1* ∧ *r* ⊆ *f* ⟶
f = *?f1'* ∨ *f* = *?f2'*)
by (*smt* (*verit*, *best*) *aff-dim-translation-eq face-of-translation-eq image-mono*
only-fs translation-galois)
ultimately show *?thesis*
by *blast*
qed

3.4 Lower bounds on face counts

3.4.1 Vertices

lemma *polytope-vertex-lower-bound*:
assumes *polytope p*
shows *aff-dim p* + *1* ≤ *card* {*v*. *v face-of p* ∧ *aff-dim v* = *0*}
proof −
let *?V* = {*v*. *v extreme-point-of p*}
have *sets-eq*: {*v*. *v face-of p* ∧ *aff-dim v* = *0*} = (λ*v*. {*v*}) ‘ *?V*
using *face-of-singleton aff-dim-sing aff-dim-eq-0*
by *auto*
then have *cards-eq*: *card* {*v*. *v face-of p* ∧ *aff-dim v* = *0*} = *card ?V*
unfolding *sets-eq* **using** *card-image inj-singleton*
by *blast*
have *aff-dim p* + *1* ≤ *aff-dim* (*convex hull ?V*) + *1*
using *Krein-Milman-Minkowski*[*OF polytope-imp-compact*[*OF assms*] *polytope-imp-convex*[*OF assms*]]
by *simp*
moreover have *aff-dim* (*convex hull ?V*) + *1* ≤ *card ?V*
using *finite-polyhedron-extreme-points*[*OF polytope-imp-polyhedron*[*OF assms*]]
aff-dim-le-card[*of ?V*] *aff-dim-convex-hull*[*of ?V*]
by *simp*
ultimately show *?thesis*
unfolding *cards-eq* **by** *auto*
qed

3.4.2 Facets

lemma *polytope-facet-lower-bound-0*:
  assumes *polytope p*
  assumes *aff-dim p* $\neq$ *0*
  assumes *origin-in-p*: *0* $\in$ *p*
  shows *aff-dim p* + *1* $\leq$ *card* {*f*. *f facet-of p*}
proof −
  have *aff-dim-eq-dim*: *aff-dim p* = *dim p*
    using *origin-in-p aff-dim-zero*[*of p*]
    by (*simp add*: *hull-inc*)

  obtain *H a b* **where**
    finite H **and**
    p-eq: *p* = *affine hull p* $\cap$ $\bigcap$ *H* **and**
    halfspaces: $\bigwedge$*h*. *h*$\in$*H* $\Longrightarrow$ *a h* $\neq$ *0* $\wedge$ *h* = {*x*. *a h* $\cdot$ *x* $\leq$ *b h*} $\wedge$
         ($\forall$ *x* $\in$ *affine hull p*. *x* + *a h* $\in$ *affine hull p*) **and**
    minimal: $\bigwedge$*F'*. *F'* $\subset$ *H* $\longrightarrow$ *p* $\subset$ *affine hull p* $\cap$ $\bigcap$ *F'*
    using *polyhedron-Int-affine-parallel-minimal*[*of p*]
      polytope-imp-polyhedron[*OF assms*(*1*)]
    by *metis*
  have *p-eq-span*: *p* = *span p* $\cap$ $\bigcap$ *H*
    using *p-eq* ‹*0* $\in$ *p*› *affine-hull-span-0*
    by *blast*
  have $\bigwedge$*h*. *h*$\in$*H* $\Longrightarrow$ *0*$\in$*h*
    using ‹*0* $\in$ *p*› *p-eq*
    by *blast*
  then have *b-nonneg*: $\bigwedge$*h*. *h*$\in$*H* $\Longrightarrow$ *0* $\leq$ *b h*
    by (*metis halfspaces inner-zero-right mem-Collect-eq*)

  have *ineq1*: *dim p* + *1* $\leq$ *card H*
  proof (*rule ccontr*)
    assume ¬(*dim p* + *1* $\leq$ *card H*)
    then have $*$: *card H* $\leq$ *dim p*
      by *auto*
    have *H* $\neq$ {}
      by (*smt* (*verit*, *ccfv-threshold*) *IntE Int-lower2 affine-affine-hull*
        affine-bounded-eq-lowdim closure-Inter-convex-open
        convex-rel-interior-finite-Inter emptyE finite.intros(*1*) *image-is-empty*
        inf.orderE local.aff-dim-eq-dim of-nat-0-le-iff origin-in-p p-eq
        polytope-eq-bounded-polyhedron rel-interior-eq-closure subset-hull
        assms(*1*,*2*))
    then obtain *h* **where** *h*$\in$*H*
      by *blast*
    have $\forall$ *h'*$\in$*H*. *a h'* $\in$ *span p*
      by (*metis add-0 affine-hull-span-0 halfspaces hull-inc origin-in-p*)
    then have *span* (*a* ‘ (*H* − {*h*})) $\subseteq$ *span p*
      using *span-mono*[*of a* ‘ (*H* − {*h*}) *span p*] *span-span*[*of p*]
      by *blast*
    moreover have *span* (*a* ‘ (*H* − {*h*})) $\neq$ *span p*

```
proof
  assume span (a ‘ (H − {h})) = span p
  then have dim (span p) = dim (span (a ‘ (H − {h})))
    by argo
  also have ... ≤ card (a ‘ (H − {h}))
    by (simp add: ‹finite H› dim-le-card')
  finally show False
    by (metis (no-types, lifting) ∗ ‹finite H› ‹h ∈ H› card-Diff1-less-iff
        card-image-le dim-span finite-Diff linorder-not-le order-trans)
qed
ultimately have span-psubset: span (a ‘ (H − {h})) ⊂ span p
  by blast
obtain n where
  n ≠ 0
  n ∈ span p and
  orthogonal: ⋀x. x ∈ span (a ‘ (H − {h})) ⟹ orthogonal n x
  using orthogonal-to-subspace-exists-gen[OF span-psubset]
  by blast
obtain B where bounded: ⋀x. x∈p ⟹ norm x ≤ B
  using polytope-imp-bounded[OF assms(1)] bounded-pos[of p]
  by blast
{assume a h · n ≥ 0
  let ?x = −((B + 1) / (norm n)) ∗R n
  have ?x ∈ span p
    using ‹n ∈ span p› span-mul
    by blast
  moreover have ⋀k. k∈H ⟹ (a k) · ?x ≤ (b k)
  proof −
    fix k
    assume k∈H
    {assume k = h
      have a h · ?x ≤ b h
        using ‹0 ≤ a h · n› ‹h ∈ H› b-nonneg bounded divide-nonneg-nonneg
          inner-simps(6) origin-in-p real-0-le-add-iff
        by fastforce
    } moreover {assume k ≠ h
      have (a k) · ?x ≤ (b k)
       by (smt (verit, best) ‹k ∈ H› ‹k ≠ h› b-nonneg image-eqI inner-commute
           inner-simps(6) insert-Diff-single insert-absorb insert-iff mult-eq-0-iff
           orthogonal orthogonal-def span-superset subsetD)
    } ultimately show a k · ?x ≤ b k
      by blast
  qed
  ultimately have ?x ∈ p
    using p-eq-span halfspaces
    by blast
  have norm ?x > B
    by (simp add: ‹n ≠ 0›)
  then have False
```

```
      using bounded[OF ‹?x ∈ p›]
      by linarith
  } moreover {assume a h · n ≤ 0
    let ?x = ((B + 1) / (norm n)) *R n
    have ?x ∈ span p
      using ‹n ∈ span p› span-mul
      by blast
    moreover have ⋀k. k∈H ⟹ a k · ?x ≤ b k
    proof −
      fix k
      assume k∈H
      {assume k = h
        have a h · ?x ≤ b h
       by (smt (verit, best) ‹a h · n ≤ 0› ‹h ∈ H› b-nonneg bounded inner-simps(6)
             norm-ge-zero norm-zero origin-in-p real-scaleR-def split-scaleR-neg-le
             zero-le-divide-iff zero-less-one-class.zero-le-one)
      } moreover {assume k ≠ h
        have (a k) · ?x ≤ (b k)
         by (smt (verit, best) ‹k ∈ H› ‹k ≠ h› b-nonneg image-eqI inner-commute
             inner-simps(6) insert-Diff-single insert-absorb insert-iff mult-eq-0-iff
             orthogonal orthogonal-def span-superset subsetD)
      } ultimately show a k · ?x ≤ b k
        by blast
    qed
    ultimately have ?x ∈ p
      using p-eq-span halfspaces
      by blast
    have norm ?x > B
      by (simp add: ‹n ≠ 0›)
    then have False
      using bounded[OF ‹?x ∈ p›]
      by linarith
  } ultimately show False
    by linarith
qed

have ineq2: card H = card {f. f facet-of p}
proof −
  let ?g = λh. p ∩ {x. a h · x = b h}
  have halfspaces-simple: ⋀h'. h' ∈ H ⟹ a h' ≠ 0 ∧ h' = {x. a h' · x ≤ b h'}
    using halfspaces by blast
  have {f. f facet-of p} = ?g ‘ H
    using facet-of-polyhedron-explicit[OF ‹finite H› p-eq halfspaces-simple] mini-
mal
    by auto
  moreover have inj-on ?g H
   using facets-of-polyhedron-explicit-distinct[OF ‹finite H› p-eq halfspaces-simple]
    by (auto simp: minimal inj-on-def)
  ultimately show ?thesis
```

using *card-image* **by** *fastforce*
qed

show *?thesis*
unfolding *aff-dim-eq-dim*
using *ineq1 ineq2*
by *linarith*
qed

lemma *polytope-facet-lower-bound*:
assumes *polytope p*
assumes *aff-dim p* $\neq$ *0*
shows *aff-dim p* + *1* $\leq$ *card* {*f*. *f facet-of p*}
proof (*cases p* = {})
case *True*
then show *?thesis*
using *aff-dim-empty*[*of p*] *facet-of-empty*
by *simp*
next
case *False*
then obtain *z* **where** *z* $\in$ *p*
by *blast*
have *has-origin*: *0* $\in$ ((+) (−*z*) ‘ *p*)
using ‹*z* $\in$ *p*› **by** *simp*
have *is-polytope*: *polytope* ((+) (−*z*) ‘ *p*)
using *polytope-translation-eq*[*of* −*z p*] *assms*(*1*)
by *blast*
have *nonzero-aff-dim*: *aff-dim* ((+) (−*z*) ‘ *p*) $\neq$ *0*
using *aff-dim-translation-eq*[*of* −*z p*] *assms*(*2*)
by *argo*
have *h1*: *card* {*f*. *f facet-of* (+) (− *z*) ‘ *p*} = *card* ((‘) ((+) (− *z*)) ‘ {*f*. *f facet-of p*})
using *facets-of-translation*[*of* −*z p*]
by *argo*
moreover have *inj-on* ((‘) ((+) (− *z*))) {*f*. *f facet-of p*}
by (*simp add*: *inj-on-image*)
ultimately have *card* {*f*. *f facet-of* (+) (− *z*) ‘ *p*} = *card* {*f*. *f facet-of p*}
by (*metis card-image*)
then show *?thesis*
using *polytope-facet-lower-bound-0*[*OF is-polytope nonzero-aff-dim has-origin*]
unfolding *aff-dim-translation-eq*[*of* −*z p*]
by *argo*
qed

3.5 Double counting

lemma *multiple-counting*:
fixes *R* :: ′*a* ⇒ ′*b* ⇒ *bool*
assumes *finite s*

```
  assumes finite t
  assumes ⋀x. x∈s ⟹ card {y∈t. R x y} = m
  assumes ⋀y. y∈t ⟹ card {x∈s. R x y} = n
  shows m ∗ card s = n ∗ card t
proof −
  have lhs: (∑x∈s. ∑y∈{y∈t. R x y}. 1) = m ∗ card s
    using assms(1,3) by simp
  have rhs: (∑y∈t. ∑x∈{x∈s. R x y}. 1) = n ∗ card t
    using assms(2,4) by simp
  show ?thesis
    using sum.swap-restrict[OF assms(1) assms(2), of λx y. (1::nat) R]
    unfolding lhs rhs .
qed
```

3.5.1 $mF = 2E$

Let F be the number of facets and E be the number of ridges. If there are m ridges per facet, then m*F = 2*E.

```
lemma facet-ridge-relation:
  assumes polytope p
    and aff-dim p ≥ 2
    and edges-per-face: ∀f. f face-of p ∧ aff-dim f = aff-dim p − 1 ⟶
          card {e. e face-of p ∧ aff-dim e = aff-dim p − 2 ∧ e ⊆ f} = m
  shows m ∗ card {f. f face-of p ∧ aff-dim f = aff-dim p − 1} =
      2 ∗ card {e. e face-of p ∧ aff-dim e = aff-dim p − 2}
proof −
  let ?faces-set = {f. f face-of p ∧ aff-dim f = aff-dim p − 1}
  let ?edges-set = {e. e face-of p ∧ aff-dim e = aff-dim p − 2}
  have finite-faces: finite ?faces-set
    using finite-polytope-faces[OF ‹polytope p›]
    by fast
  have finite-edges: finite ?edges-set
    using finite-polytope-faces[OF ‹polytope p›]
    by fast
  have lhs-m: ⋀f. f∈?faces-set ⟹ card {e∈?edges-set. e ⊆ f} = m
    using edges-per-face by auto
  have rhs-2: ⋀e. e∈?edges-set ⟹ card {f∈?faces-set. e ⊆ f} = 2
  proof −
    fix e
    assume ∗: e∈?edges-set
    have e face-of p
      using ∗
      by blast
    have aff-dim-e: aff-dim e = aff-dim p − 2
      using ∗ by auto
    then have e ≠ {}
      using assms(2) by auto
    have polyhedron p
      using polytope-imp-polyhedron[OF ‹polytope p›] .
```

obtain *f1 f2* **where**
f1-props: *f1 face-of p aff-dim f1* = *aff-dim p* − *1 e* ⊆ *f1* **and**
f2-props: *f2 face-of p aff-dim f2* = *aff-dim p* − *1 e* ⊆ *f2* **and**
distinct: *f1* ≠ *f2* **and**
characterization: ∀ *f*. *f face-of p* ∧ *aff-dim f* = *aff-dim p* − *1* ∧ *e* ⊆ *f* ⟶ *f* = *f1* ∨ *f* = *f2*
using *polyhedron-ridge-two-facets*[*OF* ‹*polyhedron p*› ‹*e face-of p*› ‹*e* ≠ {}› *aff-dim-e*]
by *auto*
have {*f*∈*?faces-set*. *e* ⊆ *f*} = {*f1*, *f2*}
using *characterization f1-props f2-props*
by *auto*
then show *card* {*f*∈*?faces-set*. *e* ⊆ *f*} = *2*
using *distinct*
by *simp*
qed
show *?thesis*
using *multiple-counting*[*OF finite-faces finite-edges lhs-m rhs-2*] **.**
qed

3.5.2 $nV = 2E$

Let V be the number of vertices and E be the number of edges. If there are n edges per vertex, then n*V = 2*E.

lemma *edge-vertex-relation*:
assumes *polytope p*
and *edges-per-vert*: ∀ *v*. *v face-of p* ∧ *aff-dim v* = *0* ⟶
card {*e*. *e face-of p* ∧ *aff-dim e* = *1* ∧ *v* ⊆ *e*} = *n*
shows *n* ∗ *card* {*v*. *v face-of p* ∧ *aff-dim v* = *0*} =
2 ∗ *card* {*e*. *e face-of p* ∧ *aff-dim e* = *1*}
proof −
let *?vertices-set* = {*v*. *v face-of p* ∧ *aff-dim v* = *0*}
let *?edges-set* = {*e*. *e face-of p* ∧ *aff-dim e* = *1*}
have *finite-vertices*: *finite ?vertices-set*
using *finite-polytope-faces*[*OF* ‹*polytope p*›]
by *fast*
have *finite-edges*: *finite ?edges-set*
using *finite-polytope-faces*[*OF* ‹*polytope p*›]
by *fast*
have *lhs-n*: ⋀*v*. *v*∈*?vertices-set* ⟹ *card* {*e*∈*?edges-set*. *v* ⊆ *e*} = *n*
using *edges-per-vert* **by** *auto*
have *rhs-2*: ⋀*e*. *e*∈*?edges-set* ⟹ *card* {*v*∈*?vertices-set*. *v* ⊆ *e*} = *2*
proof −
fix *e*
assume ∗: *e*∈*?edges-set*
have *aff-dim e* = *1*
using ∗ **by** *auto*
then have *e* ≠ {}
by *force*

have *compact e*
using ∗ *face-of-polytope-polytope*[*OF* ‹*polytope p*›] *polytope-imp-compact*[*of e*]
by *blast*
have *convex e*
using ∗ **unfolding** *face-of-def*
by *blast*
have *collinear e*
proof −
obtain *B* **where**
aff-hull-B-e: *affine hull B* = *affine hull e* **and**
¬ *affine-dependent B* **and**
int (*card B*) = *2*
using ‹*aff-dim e* = *1*› *aff-dim-basis-exists*[*of e*] **by** *auto*
then obtain *x y* **where** *B-eq*: *B* = {*x*, *y*} *x* ≠ *y*
by (*meson card-2-iff of-nat-eq-numeral-iff*)
have *e* ⊆ *affine hull B*
by (*simp add*: *aff-hull-B-e hull-subset*)
then show *?thesis*
unfolding *B-eq* **using** *collinear-affine-hull*
by *fast*
qed
then obtain *a b* **where** *e-eq*: *e* = *closed-segment a b*
using *compact-convex-collinear-segment*[*OF* ‹*e* ≠ {}› ‹*compact e*› ‹*convex e*› ‹*collinear e*›]
by *blast*
then have *a* ≠ *b*
using ‹*aff-dim e* = *1*›
by *force*

have {*v*. *v extreme-point-of e*} = {*v*. {*v*} *face-of e* ∧ *aff-dim* {*v*} = *0*}
unfolding *face-of-singleton*[*symmetric*] **using** *aff-dim-sing*
by *blast*
then have (λ*x*. {*x*}) ‘ {*v*. *v extreme-point-of e*} = {*v*. *v face-of e* ∧ *aff-dim v* = *0*}
using *aff-dim-eq-0* **by** *auto*
then have (λ*x*. {*x*}) ‘ {*v*. *v extreme-point-of e*} = {*v*. *v face-of p* ∧ *aff-dim v* = *0* ∧ *v* ⊆ *e*}
using *face-of-face* ∗ **by** *force*
then have *bij-betw* (λ*x*. {*x*}) {*v*. *v extreme-point-of e*} {*v*∈*?vertices-set*. *v* ⊆ *e*}
by (*simp add*: *bij-betw-imageI inj-on-def*)
then have *card* {*v*∈*?vertices-set*. *v* ⊆ *e*} = *card* {*v*. *v extreme-point-of e*}
using *bij-betw-same-card* **by** *force*
moreover have {*v*. *v extreme-point-of e*} = {*a*, *b*}
unfolding *e-eq* **using** *extreme-point-of-segment*[*of* - *a b*] ‹*a* ≠ *b*›
by *auto*
moreover have *card* {*a*, *b*} = *2*
using ‹*a* ≠ *b*› **by** *simp*
ultimately show *card* {*v*∈*?vertices-set*. *v* ⊆ *e*} = *2*

    by *argo*
  qed
  show *?thesis*
    using *multiple-counting*[*OF finite-vertices finite-edges lhs-n rhs-2*] **.**
qed

hide-fact *multiple-counting*

3.6 Face containment

lemma *ridge-belongs-to-facet*:
  assumes *e face-of p*
    and *aff-dim e* = *aff-dim p* − *2*
    and *aff-dim p* ≥ *2*
    and *polyhedron p*
  obtains *f* **where** *f face-of p aff-dim f* = *aff-dim p* − *1 e face-of f*
proof −
  have *e* ≠ {}
    using *assms*(*2*,*3*) **by** *auto*
  moreover have *e* ≠ *p*
    using *assms*(*2*,*3*) **by** *auto*
  ultimately obtain *f* **where** *f facet-of p e* ⊆ *f*
    using *assms*(*1*,*4*) *face-of-polyhedron-subset-facet*
    by *blast*
  then have *f face-of p aff-dim f* = *aff-dim p* − *1*
    unfolding *facet-of-def* **by** *simp+*
  moreover have *e face-of f*
    using ‹*f face-of p*› *assms*(*1*) ‹*e* ⊆ *f*› *face-of-face*
    by *blast*
  ultimately show (⋀*f*. *f face-of p* ⟹ *aff-dim f* = *aff-dim p* − *1* ⟹ *e face-of f* ⟹ *thesis*) ⟹ *thesis*
    by *blast*
qed

lemma *peak-belongs-to-ridge*:
  assumes *v face-of p*
    and *aff-dim v* = *aff-dim p* − *3*
    and *aff-dim p* ≥ *3*
    and *polyhedron p*
  obtains *e* **where** *e face-of p aff-dim e* = *aff-dim p* − *2 v face-of e*
proof −
  have *v* ≠ {}
    using *assms*(*2*,*3*) **by** *auto*
  moreover have *v* ≠ *p*
    using *assms*(*2*,*3*) **by** *auto*
  ultimately obtain *f* **where** *f facet-of p v* ⊆ *f*
    using *assms*(*1*,*4*) *face-of-polyhedron-subset-facet*
    by *blast*

then have *f face-of p aff-dim f* = *aff-dim p* − *1*
  unfolding *facet-of-def assms*(*3*)
  by *simp*+
moreover have *v face-of f*
  using ‹*f face-of p*› *assms*(*1*) ‹*v* ⊆ *f*› *face-of-face*
  by *blast*
obtain *e* **where** *e face-of f aff-dim e* = *aff-dim p* − *2 v face-of e*
  using *ridge-belongs-to-facet*[*OF* ‹*v face-of f*› - -
      face-of-polyhedron-polyhedron[*OF assms*(*4*) ‹*f face-of p*›]]
   assms(*2*,*3*) ‹*aff-dim f* = *aff-dim p* − *1*›
  by *auto*
then show (⋀*e*. *e face-of p* ⟹ *aff-dim e* = *aff-dim p* − *2* ⟹ *v face-of e* ⟹ *thesis*) ⟹ *thesis*
  using *face-of-trans*[*OF* ‹*e face-of f*› ‹*f face-of p*›]
  by *blast*
qed

end

4 Computational helpers

theory *Computation*
 imports
  Convex-More
  Determinants-More
  HOL−*Analysis.Cross3*
  HOL−*Analysis.Polytope*
  HOL−*Eisbach.Eisbach*
  Polytope-More
  HOL−*Examples.Sqrt*
begin

4.1 Operations in $\mathbb{Q}(\sqrt{5})$

Numbers of the form $a + b\sqrt{5}$, where $a, b \in \mathbb{Q}$

abbreviation *φ* :: *real*
 where *φ* ≡ (*1* + *sqrt 5*)/*2*

lemma *real-rat5-mul*:
 (*a1* + *b1* ∗ *sqrt 5*) ∗ (*a2* + *b2* ∗ *sqrt 5*)
  = (*a1* ∗ *a2* + *5* ∗ *b1* ∗ *b2*) + (*a1* ∗ *b2* + *a2* ∗ *b1*) ∗ *sqrt 5*
 by (*simp add*: *algebra-simps*)

lemma *real-rat5-inv*:
 assumes *a*^*2* ≠ *5* ∗ *b*^*2*
 shows *inverse* (*a* + *b* ∗ *sqrt 5*) = *a* / (*a*^*2* − *5* ∗ *b*^*2*) + −*b* / (*a*^*2* − *5* ∗ *b*^*2*) ∗ *sqrt 5*
proof −

```
  have (a + b * sqrt 5) * (a - b * sqrt 5) = a^2 - 5 * b^2
    by (simp add: algebra-simps power2-eq-square)
  moreover have a^2 - 5 * b^2 ≠ 0
    using assms by simp
  ultimately have inverse (a + b * sqrt 5) = (a - b * sqrt 5) / (a^2 - 5 * b^2)
    by (metis divide-inverse mult-1 mult-eq-0-iff
        nonzero-divide-mult-cancel-right)
  then show ?thesis
    by argo
qed

lemma real-rat5-div:
  assumes a2^2 ≠ 5 * b2^2
  shows (a1 + b1 * sqrt 5) / (a2 + b2 * sqrt 5) =
          ((a1 * a2 - 5 * b1 * b2) / (a2^2 - 5 * b2^2)) +
          ((a2 * b1 - a1 * b2) / (a2^2 - 5 * b2^2)) * sqrt 5
  unfolding divide-inverse[of (a1 + b1 * sqrt 5) (a2 + b2 * sqrt 5)]
    real-rat5-inv[OF assms]
  using real-rat5-mul[of a1 b1 a2 / (a2^2 - 5 * b2^2) -b2 / (a2^2 - 5 * b2^2)]
  by argo

lemma real-rat5-le-sqrt-cases:
  x ≤ y * sqrt 5 ⟷
   (x ≤ 0 ∧ 0 ≤ y) ∨
   (0 ≤ x ∧ 0 ≤ y ∧ x^2 ≤ 5 * y^2) ∨
   (x ≤ 0 ∧ y ≤ 0 ∧ 5 * y^2 ≤ x^2)
proof -
  have sqrt5-pos: sqrt 5 > 0 by simp
  have sqrt5-sq: (sqrt 5)^2 = 5 by simp

  consider
    (x-ge0-y-ge0) x ≥ 0 y ≥ 0 |
    (x-ge0-y-le0) x ≥ 0 y ≤ 0 |
    (x-le0-y-ge0) x ≤ 0 y ≥ 0 |
    (x-le0-y-le0) x ≤ 0 y ≤ 0
    by argo

  then show ?thesis
  proof (cases)
    case x-ge0-y-ge0
    then show ?thesis
      using power-mono-iff[of x y * sqrt 5 2]
      by (auto simp add: power-mult-distrib sqrt5-sq mult-ac)
  next
    case x-ge0-y-le0
    then show ?thesis
      by (smt (verit) mult.left-commute mult-le-0-iff mult-zero-left
          not-numeral-le-zero power2-eq-square power2-less-eq-zero-iff
          real-sqrt-le-0-iff zero-le-mult-iff)
```

next
case *x-le0-y-ge0*
then show *?thesis*
by (*smt* (*verit*, *del-insts*) *sqrt5-pos power-mult-distrib real-sqrt-mult real-sqrt-pow2-iff*)
next
case *x-le0-y-le0*
have $x \le y * sqrt\ 5 \longleftrightarrow -x \ge -y * sqrt\ 5$
by *linarith*
also have ... $\longleftrightarrow (-x)\hat{}2 \ge (-y * sqrt\ 5)\hat{}2$
by (*smt* (*verit*) *x-le0-y-le0 sqrt5-pos mult-nonneg-nonneg pos2 power-mono-iff*)
also have ... $\longleftrightarrow x\hat{}2 \ge 5 * y\hat{}2$
by (*simp add*: *power-mult-distrib sqrt5-sq mult-ac*)
finally show *?thesis* **using** ‹$x \le 0$› ‹$y \le 0$› **by** *auto*
qed
qed

lemma *real-rat5-le-comparison*:
$(a1 + b1 * sqrt\ 5) \le (a2 + b2 * sqrt\ 5) \longleftrightarrow$
$(a1 \le a2 \wedge b1 \le b2) \vee$
$(a2 \le a1 \wedge b1 \le b2 \wedge (a1 - a2)\hat{}2 \le 5 * (b2 - b1)\hat{}2) \vee$
$(a1 \le a2 \wedge b2 \le b1 \wedge 5 * (b2 - b1)\hat{}2 \le (a1 - a2)\hat{}2)$
using *real-rat5-le-sqrt-cases*[*of* $a1 - a2$ $b2 - b1$]
by *argo*

Wrapper definition to avoiding oversimplification when applying simp

definition *rat5* :: *rat* ⇒ *rat* ⇒ *real*
where *rat5 a b* = *real-of-rat a* + *real-of-rat b* ∗ *sqrt 5*

lemma *rat5-add*: *rat5 a1 b1* + *rat5 a2 b2* = *rat5* (*a1* + *a2*) (*b1* + *b2*)
unfolding *rat5-def*
by (*simp add*: *algebra-simps of-rat-add*)

lemma *rat5-sub*: *rat5 a1 b1* − *rat5 a2 b2* = *rat5* (*a1* − *a2*) (*b1* − *b2*)
unfolding *rat5-def*
by (*simp add*: *algebra-simps of-rat-diff*)

lemma *rat5-mul*: *rat5 a1 b1* ∗ *rat5 a2 b2* = *rat5* (*a1* ∗ *a2* + 5 ∗ *b1* ∗ *b2*) (*a1* ∗ *b2* + *b1* ∗ *a2*)
unfolding *rat5-def*
by (*simp add*: *algebra-simps of-rat-mult of-rat-add*)

lemma *rat5-div*:
assumes $a2 \neq 0 \vee b2 \neq 0$
shows *rat5 a1 b1* / *rat5 a2 b2* =
rat5 $((a1 * a2 - 5 * b1 * b2) / (a2\hat{}2 - 5 * b2\hat{}2))$
$((a2 * b1 - a1 * b2) / (a2\hat{}2 - 5 * b2\hat{}2))$
proof −
have *denom-nonzero*: $(real\text{-}of\text{-}rat\ a2)\hat{}2 \neq 5 * (real\text{-}of\text{-}rat\ b2)\hat{}2$

proof
  assume ∗: (*real-of-rat a2*)^2 = 5 ∗ (*real-of-rat b2*)^2
  show *False*
  proof (*cases b2 = 0*)
    case *True*
    then show *False*
      using ∗ *assms* **by** *simp*
  next
    case *False*
    have *5 = (real-of-rat a2 / real-of-rat b2)^2*
      using ∗ *False*
      by (*simp add*: *field-simps of-rat-divide*)
    then have *sqrt 5 = abs (real-of-rat (a2 / b2))*
      by (*metis of-rat-divide real-sqrt-abs*)
    moreover have *sqrt 5* ∉ ℚ
      using *sqrt-prime-irrational*[*of 5*] **by** *auto*
    moreover have *abs (real-of-rat (a2 / b2))* ∈ ℚ
      by *simp*
    ultimately show *False*
      by *auto*
  qed
qed
show *?thesis*
  using *rat5-def real-rat5-div*[*OF denom-nonzero*]
  by (*simp add*: *of-rat-diff of-rat-divide of-rat-mult of-rat-power*)
qed

lemma *rat5-le*:
  (*rat5 a1 b1* ≤ *rat5 a2 b2*) =
   ((*a1* ≤ *a2* ∧ *b1* ≤ *b2*) ∨
   (*a2* ≤ *a1* ∧ *b1* ≤ *b2* ∧ (*a1* − *a2*)^2 ≤ 5 ∗ (*b2* − *b1*)^2) ∨
   (*a1* ≤ *a2* ∧ *b2* ≤ *b1* ∧ 5 ∗ (*b2* − *b1*)^2 ≤ (*a1* − *a2*)^2))
  unfolding *rat5-def real-rat5-le-comparison*
 by (*metis* (*mono-tags*, *opaque-lifting*) *of-rat-diff of-rat-less-eq of-rat-mult of-rat-numeral-eq of-rat-power*)

lemma *rat5-eq*: *rat5 a1 b1 = rat5 a2 b2* ⟷ *a1 = a2* ∧ *b1 = b2*
proof −
 {
  assume ∗: *rat5 a1 b1 = rat5 a2 b2 b1* ≠ *b2*
  have *real-of-rat* (*a1* − *a2*) = *real-of-rat* (*b2* − *b1*) ∗ *sqrt 5*
    using ∗ *rat5-def*
    by (*simp add*: *left-diff-distrib of-rat-diff*)
  moreover have *real-of-rat* (*b2* − *b1*) ≠ *0*
    using ∗ **by** *simp*
  ultimately have *sqrt 5 = real-of-rat* ((*a1* − *a2*) / (*b2* − *b1*))
    by (*simp add*: *of-rat-divide*)
  then have *False*
    using *sqrt-prime-irrational*[*of 5*] **by** *auto*

} **then show** *?thesis*
  using *rat5-def* **by** *force*
qed

lemma *rat5-0-1*:
  shows *rat5 0 0 = 0 rat5 1 0 = 1* **unfolding** *rat5-def* **by** *auto*

lemma *rat5-eq-0*:
  $rat5\ a1\ b1 = 0 \longleftrightarrow a1 = 0 \wedge b1 = 0$
  using *rat5-eq*[*of a1 b1 0 0*] **unfolding** *rat5-def*
  by *simp*

lemma *rat5-eq-1*:
  $rat5\ a1\ b1 = 1 \longleftrightarrow a1 = 1 \wedge b1 = 0$
  using *rat5-eq*[*of a1 b1 1 0*] **unfolding** *rat5-def*
  by *simp*

4.2 Operations in $\mathbb{R}^3$

lemma *vector3-add*:
  fixes *x1 x2 x3 y1 y2 y3* :: *real*
  shows *vector* [*x1*, *x2*, *x3*] + *vector* [*y1*, *y2*, *y3*] =
        vector [*x1* + *y1*, *x2* + *y2*, *x3* + *y3*]
  by (*simp add*: *vec-eq-iff vector-def forall-3*)

lemma *vector3-sub*:
  fixes *x1 x2 x3 y1 y2 y3* :: *real*
  shows *vector* [*x1*, *x2*, *x3*] − *vector* [*y1*, *y2*, *y3*] =
        vector [*x1* − *y1*, *x2* − *y2*, *x3* − *y3*]
  by (*simp add*: *vec-eq-iff vector-def forall-3*)

unbundle *cross3-syntax*

lemma *vector3-cross*:
  fixes *x1 x2 x3 y1 y2 y3* :: *real*
  shows *vector* [*x1*, *x2*, *x3*] × *vector* [*y1*, *y2*, *y3*] =
        (*vector* [*x2* ∗ *y3* − *x3* ∗ *y2*, *x3* ∗ *y1* − *x1* ∗ *y3*, *x1* ∗ *y2* − *x2* ∗ *y1*] :: *real^3*)
  by (*simp add*: *cross3-def*)

lemma *vector3-eq-0*:
  fixes *x1 x2 x3* :: *real*
  shows (*vector* [*x1*, *x2*, *x3*] :: *real^3*) = 0 ⟷ *x1* = 0 ∧ *x2* = 0 ∧ *x3* = 0
  by (*simp add*: *vec-eq-iff vector-def forall-3*)

lemma *vector3-eq*:
  fixes *x1 x2 x3 y1 y2 y3* :: *real*
  shows (*vector* [*x1*, *x2*, *x3*] :: *real^3*) = *vector* [*y1*, *y2*, *y3*] ⟷ *x1* = *y1* ∧ *x2* = *y2* ∧ *x3* = *y3*

by (*simp add*: *vec-eq-iff vector-def forall-3*)

lemma *vector3-neq*:
shows $((vector\ [x1,\ x2,\ x3]::real\hat{}3) \neq vector\ [y1,\ y2,\ y3]) =$
$(x1 \neq y1 \lor x2 \neq y2 \lor x3 \neq y3)$
by (*simp add*: *vec-eq-iff forall-3*)

lemma *vector3-dot*:
fixes *x1 x2 x3 y1 y2 y3* :: *real*
shows $(vector\ [x1,\ x2,\ x3]\ ::\ real\hat{}3) \cdot vector\ [y1,\ y2,\ y3] =$
$x1{*}y1 + x2{*}y2 + x3{*}y3$
by (*simp add*: *inner-vec-def sum-3 vector-def*)

4.3 Computation of faces

4.3.1 2-faces

lemma *compute-faces-2*:
fixes *F S* :: (*real*^*3*) *set*
assumes *finite S*
shows (*F face-of* (*convex hull S*) ∧ *aff-dim F* = *2*) =
$(\exists x{\in}S.\ \exists y{\in}S.\ \exists z{\in}S.$
let $a = (z - x) \times (y - x)$ *in*
$a \neq 0 \land$
(*let* $b = a \cdot x$ *in*
$((\forall w{\in}S.\ a \cdot w \le b) \lor (\forall w{\in}S.\ a \cdot w \ge b)) \land$
F = *convex hull* $(S \cap \{x.\ a \cdot x = b\})))$
(**is** *?lhs* = *?rhs*)
proof
assume *?lhs*
then have ∗: *F face-of* (*convex hull S*) *aff-dim F* = *2* **by** *blast+*
then obtain *T* **where** *T-facts*: *T* ⊆ *S F* = *convex hull T*
by (*meson assms face-of-convex-hull-subset finite-imp-compact*)
then obtain *U* **where** *U-facts*: *U* ⊆ *T* ¬ *affine-dependent U affine hull T* = *affine hull U*
using *affine-basis-exists*[*of T*] **by** *blast*
then have *aff-dim U* = *2*
by (*metis* ∗(*2*) *T-facts*(*2*) *aff-dim-affine-hull aff-dim-convex-hull*)
then have *card U* = *3*
using *U-facts*(*2*) *aff-dim-affine-independent* **by** *fastforce*
then obtain *x y z* **where**
U-eq: $U = \{x,\ y,\ z\}$ **and**
xyz-neq: $x \neq y\ x \neq z\ y \neq z$
using *card-3-iff*[*of U*] **by** *blast*
then have *not-collinear*: ¬ *collinear* $\{x,\ y,\ z\}$
using *U-facts*(*2*) *exposed-face-of-polyhedron*
by (*simp add*: *collinear-3-eq-affine-dependent*)
have *xyz-in-S*: $x \in S\ y \in S\ z \in S$
using *U-eq U-facts*(*1*) *T-facts*(*1*) **by** *blast+*

```
let ?a = (z − x) × (y − x)
let ?b = ?a · x
let ?H = {w. ?a · w = ?b}
have ?a ≠ 0
  using not-collinear collinear-3 cross-eq-0[symmetric]
  by (metis (no-types, opaque-lifting) NO-MATCH-def insert-commute)
have ?a · y = ?b
   by (metis (no-types, opaque-lifting) add.commute diff-0-right diff-add-cancel
dot-cross-self(2) inner-commute
      inner-diff-left)
have ?a · z = ?b
  by (metis Cross3.left-diff-distrib cross-refl cross-triple diff-left-imp-eq diff-self)

have F exposed-face-of (convex hull S)
  using ∗(1) polyhedron-convex-hull[OF assms] exposed-face-of-polyhedron
  by blast
then obtain a′ b′ where a′-b′-facts:
  convex hull S ⊆ {w. a′ · w ≤ b′}
  convex hull T = convex hull S ∩ {w. a′ · w = b′}
  unfolding exposed-face-of-def T-facts(2) by blast
let ?H′ = {w. a′ · w = b′}

have {x, y, z} ⊆ ?H
  using ‹?a · y = ?b› ‹?a · z = ?b› by simp
then have affine hull {x, y, z} ⊆ ?H
  using affine-hyperplane hull-minimal by metis
then have affine hull T ⊆ ?H
  using U-eq U-facts(3) by blast
moreover have affine hull T ⊆ ?H′
  using a′-b′-facts(2)
  by (metis affine-hyperplane affine-imp-convex inf.cobounded2 subset-hull)
ultimately have affine hull T ⊆ ?H ∩ ?H′
  by blast
moreover have aff-dim (affine hull T) = 2
  by (metis U-facts(3) ‹aff-dim U = 2› aff-dim-affine-hull)
ultimately have aff-dim (?H ∩ ?H′) ≥ 2
  using aff-dim-subset
  by metis

show ?rhs
proof (cases ?H′ = UNIV)
  case True
  have S ⊆ affine hull T
    using True a′-b′-facts(2) convex-hull-subset-affine-hull hull-subset
    by fast
  then have s-in-plane: S ⊆ ?H
    using ‹affine hull T ⊆ ?H› by blast
  then have one-side: (∀ w∈S. ?a · w ≤ ?b) ∨ (∀ w∈S. ?a · w ≥ ?b)
    by auto
```

```
    have S ∩ ?H = S
      using s-in-plane by auto
    then have F = convex hull (S ∩ {w. ?a · w = ?b})
      using True a'-b'-facts(2) T-facts(2) by simp

    then show ?thesis
      using xyz-in-S ‹?a ≠ 0› one-side
      unfolding Let-def by blast
  next
    case False
    have planes-eq: ?H = ?H'
    proof (rule ccontr)
      assume neq: ?H ≠ ?H'
      have aff-dim ?H = 2
        using ‹?a ≠ 0› aff-dim-hyperplane[of ?a ?b] by simp
      then have aff-dim (?H ∩ ?H') < 2
        using aff-dim-affine-Int-hyperplane[of ?H a' b']
          neq False subset-hyperplanes[of ?a ?b a' b']
        by (simp add: affine-hyperplane)
      then show False
        using ‹aff-dim (?H ∩ ?H') ≥ 2›
        by simp
    qed

    then have F-eq: F = convex hull (S ∩ ?H)
       by (smt (verit, ccfv-threshold) Int-commute T-facts(1,2) a'-b'-facts(2) con-
vex-convex-hull
          hull-minimal hull-subset inf.orderE le-inf-iff subsetI)

    have ?H' ⊆ ?H
      using planes-eq by auto
    moreover have a' ≠ 0
      by (metis ‹?a ≠ 0› hyperplane-eq-UNIV hyperplane-eq-empty planes-eq)
    ultimately obtain k where ?a = k *R a' ?b = k * b' k ≠ 0
      using ‹?a ≠ 0› hyperplane-subset-imp-scaled
      by blast
    then have (∀ w∈S. ?a · w ≤ ?b) ∨ (∀ w∈S. ?a · w ≥ ?b)
      using a'-b'-facts(1) hull-subset[of S]
      by (auto simp: mult-le-cancel-left)

    then show ?thesis
      using xyz-in-S ‹?a ≠ 0› F-eq unfolding Let-def
      by blast
  qed
next
  assume ?rhs
  then obtain x y z where
    xyz-in-S: x ∈ S y ∈ S z ∈ S and
```

a-neq-0: $(z - x) \times (y - x) \neq 0$ **and**
one-side: $(\forall w \in S.\ (z - x) \times (y - x) \cdot w \leq (z - x) \times (y - x) \cdot x) \vee$
$(\forall w \in S.\ (z - x) \times (y - x) \cdot w \geq (z - x) \times (y - x) \cdot x)$ **and**
F-eq: $F = convex\ hull\ (S \cap \{w.\ (z - x) \times (y - x) \cdot w = (z - x) \times (y - x) \cdot x\})$
unfolding *Let-def* **by** *blast*
let $?a = (z - x) \times (y - x)$
let $?b = ?a \cdot x$
let $?H = \{w.\ ?a \cdot w = ?b\}$
have $convex\ hull\ (S \cap ?H) = convex\ hull\ S \cap ?H$
proof
show $convex\ hull\ (S \cap ?H) \subseteq convex\ hull\ S \cap ?H$
using *convex-hyperplane*[*of ?a ?b*] *hull-mono*[*of S* $\cap$ *?H S convex*]
subset-hull[*of convex ?H S* $\cap$ *?H*]
by *blast*
next
show $convex\ hull\ S \cap ?H \subseteq convex\ hull\ (S \cap ?H)$
proof (*cases* $S \subseteq ?H$)
case *True*
then show *?thesis* **by** (*simp add*: *Int-absorb2*)
next
case *False*
let $?S\text{-}on = S \cap ?H$
let $?S\text{-}off = S - ?H$
have $convex\ hull\ S \cap ?H = convex\ hull\ (convex\ hull\ (?S\text{-}on) \cup convex\ hull\ (?S\text{-}off)) \cap ?H$
by (*metis Int-Diff-Un hull-Un-left hull-Un-right*)
moreover have $convex\ hull\ (convex\ hull\ (?S\text{-}on) \cup convex\ hull\ (?S\text{-}off)) \cap ?H \subseteq convex\ hull\ (S \cap ?H)$
proof
fix *w*
assume $*$: $w \in convex\ hull\ (convex\ hull\ (?S\text{-}on) \cup convex\ hull\ (?S\text{-}off)) \cap ?H$
have *convex*: $convex\ (convex\ hull\ (?S\text{-}on))\ convex\ (convex\ hull\ (?S\text{-}off))$
using *convex-convex-hull* **by** *blast+*
have *nonempty1*: $convex\ hull\ ?S\text{-}on \neq \{\}$
using *xyz-in-S*(*1*) *convex-hull-eq-empty* **by** *blast*
have *nonempty2*: $convex\ hull\ ?S\text{-}off \neq \{\}$
using *False convex-hull-eq-empty* **by** *simp*
obtain *u p q* **where**
w-facts: $w = (1 - u) *_R p + u *_R q\ ?a \cdot w = ?b$ **and**
p-in: $p \in convex\ hull\ ?S\text{-}on$ **and**
q-in: $q \in convex\ hull\ ?S\text{-}off$ **and**
u-range: $0 \leq u\ u \leq 1$
using $*$ *convex-hull-union-nonempty-explicit*[*OF convex*(*1*) *nonempty1 convex*(*2*) *nonempty2*]
by *blast*
show $w \in convex\ hull\ (S \cap ?H)$
proof (*cases* $u = 0$)

case *True*
then show *?thesis*
using *p-in w-facts(1)* **by** *simp*
next
case *u-nonzero*: *False*
have $p \in$ *convex hull ?H*
using *p-in hull-mono*[*of ?S-on ?H*] **by** *blast*
then have $?a \cdot p = ?b$
using *convex-hyperplane hull-same*
by *blast*
have $q \notin ?H$
proof
assume $q \in ?H$
then have $?a \cdot q = ?b$ **by** *simp*
consider
(*leq*) $\forall w \in S.\ ?a \cdot w \leq ?b$ |
(*geq*) $\forall w \in S.\ ?a \cdot w \geq ?b$
using *one-side* **by** *blast*
then show *False*
proof *cases*
case *leq*
then have *?S-off* $\subseteq \{w.\ ?a \cdot w < ?b\}$
by *fastforce*
then have *convex hull ?S-off* $\subseteq \{w.\ ?a \cdot w < ?b\}$
using *convex-halfspace-lt hull-minimal* **by** *metis*
then show *False*
using ‹$?a \cdot q = ?b$› ‹$q \in$ *convex hull ?S-off*› **by** *auto*
next
case *geq*
then have *?S-off* $\subseteq \{w.\ ?a \cdot w > ?b\}$
by *fastforce*
then have *convex hull ?S-off* $\subseteq \{w.\ ?a \cdot w > ?b\}$
using *convex-halfspace-gt hull-minimal* **by** *metis*
then show *False*
using ‹$?a \cdot q = ?b$› ‹$q \in$ *convex hull ?S-off*› **by** *auto*
qed
qed
then have $?a \cdot q \neq ?b$ **by** *simp*
moreover have $?a \cdot ((1 - u) *_R p + u *_R q) = (1 - u) * (?a \cdot p) + u * (?a \cdot q)$
by (*simp add*: *inner-add-right*)
ultimately show *?thesis*
using ‹$q \in$ *convex hull ?S-off*›
by (*metis* ‹$?a \cdot p = ?b$› *real-scaleR-def segment-degen-0 u-nonzero w-facts*)
qed
qed
ultimately show *?thesis* **by** *argo*
qed
qed

then have *F-as-inter*: $F = convex\ hull\ S \cap ?H$
  using *F-eq* **by** *simp*
consider (*case-le*) $\forall\, w{\in}S.\ ?a \cdot w \le ?b$ | (*case-ge*) $\forall\, w{\in}S.\ ?a \cdot w \ge ?b$
    using *one-side* **by** *blast*
then have *face-of*: *F face-of* (*convex hull S*)
proof *cases*
  case *case-le*
  then have *subset*: $S \subseteq \{x.\ ?a \cdot x \le ?b\}$
    by *blast*
  have *convex hull* $S \subseteq \{x.\ ?a \cdot x \le ?b\}$
    using *convex-halfspace-le*[*of ?a ?b*] *hull-minimal*[*OF subset*, *of convex*]
    by *blast*
  then have $\bigwedge w.\ w{\in}convex\ hull\ S \Longrightarrow ?a \cdot w \le ?b$
    by *blast*
  then have *convex hull S* $\cap$ *?H face-of convex hull S*
    using *face-of-Int-supporting-hyperplane-le*[*OF convex-convex-hull*[*of S*]]
    by *blast*
  then show *?thesis*
    using *F-as-inter* **by** *simp*
next
  case *case-ge*
  then have *subset*: $S \subseteq \{x.\ ?a \cdot x \ge ?b\}$
    by *blast*
  have *convex hull* $S \subseteq \{x.\ ?a \cdot x \ge ?b\}$
    using *convex-halfspace-ge*[*of ?b ?a*] *hull-minimal*[*OF subset*, *of convex*]
    by *blast*
  then have $\bigwedge w.\ w{\in}convex\ hull\ S \Longrightarrow ?a \cdot w \ge ?b$
    by *blast*
  then have *convex hull S* $\cap$ *?H face-of convex hull S*
    using *face-of-Int-supporting-hyperplane-ge*[*OF convex-convex-hull*[*of S*]]
    by *blast*
  then show *?thesis*
    using *F-as-inter* **by** *simp*
qed

have *aff-dim ?H* = *2*
  using *aff-dim-hyperplane*[*OF a-neq-0*, *of ?b*] **by** *simp*
have $F \subseteq ?H$
  using *F-as-inter* **by** *blast*
then have *upper*: *aff-dim F* $\le$ *2*
  using *aff-dim-subset*[*of F ?H*] **unfolding** ‹*aff-dim ?H* = *2*›
  by *blast*

have $x \ne y\ x \ne z\ y \ne z$
  using *a-neq-0* **by** *force+*
then have *card* $\{x, y, z\} = 3$
  by *auto*
have $\neg$ *collinear* $\{x, y, z\}$
  using *collinear-3*[*of z x y*] *cross-eq-0 a-neq-0*

by (*simp add*: *insert-commute*)
then have *dim-xyz*: *aff-dim* $\{x, y, z\} = 2$
using *collinear-3-eq-affine-dependent aff-dim-affine-independent*
unfolding ‹*card* $\{x, y, z\} = 3$›
by *fastforce*
have $?a \cdot y = ?b$
by (*metis* (*no-types*, *opaque-lifting*) *dot-cross-self*(*2*) *inner-commute inner-diff-left right-minus-eq*)
then have *y-in*: $y \in S \cap ?H$
using *xyz-in-S*(*2*) **by** *blast*
have $?a \cdot z = ?b$
by (*metis* (*no-types*, *lifting*) *Cross3.right-diff-distrib add-0 add-uminus-conv-diff cross-refl cross-skew cross-triple diff-0-right*)
then have *z-in*: $z \in S \cap ?H$
using *xyz-in-S*(*3*) **by** *blast*
then have $\{x, y, z\} \subseteq$ *convex hull* $(S \cap ?H)$
using *xyz-in-S*(*1*) *y-in hull-subset* **by** *fast*
then have *xyz-in-F*: $\{x, y, z\} \subseteq F$
using *F-eq* **by** *simp*
have *lower*: $2 \leq$ *aff-dim F*
using *aff-dim-subset*[*OF xyz-in-F*] *dim-xyz* **by** *simp*

show *?lhs*
using *face-of upper lower* **by** *simp*
qed

lemma *set-choice-decomposition-1*:
assumes *sym-yz*: $\bigwedge x\ y\ z.\ Q\ x\ y\ z \Longrightarrow Q\ x\ z\ y$
and *sym-xy*: $\bigwedge x\ y\ z.\ Q\ x\ y\ z \Longrightarrow Q\ y\ x\ z$
and *degen*: $\bigwedge x\ z.\ \neg\ Q\ x\ x\ z$
shows $(\exists x\in insert\ v\ S.\ \exists y\in insert\ v\ S.\ \exists z\in insert\ v\ S.\ Q\ x\ y\ z) \longleftrightarrow$
$(\exists y\in S.\ \exists z\in S.\ Q\ v\ y\ z) \vee (\exists x\in S.\ \exists y\in S.\ \exists z\in S.\ Q\ x\ y\ z)$
by (*metis degen insertCI insertE sym-xy sym-yz*)

lemma *compute-faces-2-step-1*:
fixes *F S v* **and** *T* :: (*real*^*3*) *set*
defines $Q \equiv (\lambda x\ y\ z.\ let\ a = (z - x) \times (y - x)\ in$
$a \neq 0 \wedge$
$(let\ b = a \cdot x\ in$
$((\forall w\in T.\ a \cdot w \leq b) \vee (\forall w\in T.\ a \cdot w \geq b)) \wedge$
$F = convex\ hull\ (T \cap \{k.\ a \cdot k = b\})))$
shows $(\exists x\in (insert\ v\ S).\ \exists y \in (insert\ v\ S).\ \exists z \in (insert\ v\ S).\ Q\ x\ y\ z) =$
$((\exists y\in S.\ \exists z\in S.\ Q\ v\ y\ z) \vee (\exists x\in S.\ \exists y\in S.\ \exists z\in S.\ Q\ x\ y\ z))$
(**is** $?lhs = ?rhs$)
proof −
have *Q-degen*: $\bigwedge x\ z.\ \neg\ Q\ x\ x\ z$
unfolding *Q-def Let-def*
by (*simp add*: *cross3-simps*)
have *Q-invariant*: $Q\ x\ y\ z$ **if** $Q\ x\ z\ y$ **for** $x\ y\ z$

using *that cross-skew*[*of* $y - x$ $z - x$] **unfolding** *Q-def Let-def*
by *auto*

have *sym-xy*: $\bigwedge x\ y\ z.\ Q\ x\ y\ z \Longrightarrow Q\ y\ x\ z$
proof −
fix $x\ y\ z$
assume $Q\ x\ y\ z$
then obtain $a\ b$ **where**
a-def: $a = (z - x) \times (y - x)$ **and**
a-nonzero: $a \neq 0$ **and**
b-def: $b = a \cdot x$ **and**
ineq: $(\forall w \in T.\ a \cdot w \leq b) \vee (\forall w \in T.\ a \cdot w \geq b)$ **and**
plane: $F = \mathit{convex\ hull}\ (T \cap \{w.\ a \cdot w = b\})$
unfolding *Q-def Let-def*
by *auto*
let $?a' = (z - y) \times (x - y)$
let $?b' = ?a' \cdot y$
have $?a' = -a$
by (*smt* (*verit*, *del-insts*) *Cross3.right-diff-distrib a-def cross-refl*
cross-skew eq-iff-diff-eq-0)
have *a-orth*: $a \cdot (y - x) = 0$
unfolding *a-def dot-cross-self*(*3*)
by *blast*
then have $a \cdot y = b$
unfolding *b-def*
by (*simp add*: *inner-diff-right*)
show $Q\ y\ x\ z$
unfolding *Q-def Let-def*
using ‹$a \cdot y = b$› ‹$?a' = -a$› *a-nonzero ineq plane*
by *auto*
qed

show *?thesis*
using *set-choice-decomposition-1*[*of* $Q\ v\ S$] *Q-invariant sym-xy Q-degen Q-def*
by *blast*
qed

lemma *set-choice-decomposition-2*:
assumes *sym*: $\bigwedge x\ y.\ Q\ x\ y \Longrightarrow Q\ y\ x$
and *degen*: $\bigwedge x.\ \neg\ Q\ x\ x$
shows $(\exists y \in \mathit{insert}\ u\ S.\ \exists z \in \mathit{insert}\ u\ S.\ Q\ y\ z) \longleftrightarrow$
$(\exists z \in S.\ Q\ u\ z) \vee (\exists y \in S.\ \exists z \in S.\ Q\ y\ z)$
by (*metis degen insertCI insertE sym*)

lemma *compute-faces-2-step-2*:
fixes $F\ u\ v\ S$ **and** T :: (*real*^*3*) *set*
defines $Q \equiv (\lambda y\ z.\ \mathit{let}\ a = (z - v) \times (y - v)\ \mathit{in}$
$a \neq 0\ \wedge$
$(\mathit{let}\ b = a \cdot v\ \mathit{in}$

((∀ w∈T. a · w ≤ b) ∨ (∀ w∈T. a · w ≥ b)) ∧
F = *convex hull* (T ∩ {k. a · k = b})))
shows (∃ y ∈ (*insert u S*). ∃ z ∈ (*insert u S*). Q y z) =
((∃ z∈S. Q u z) ∨ (∃ y∈S. ∃ z∈S. Q y z))
(**is** *?lhs* = *?rhs*)
proof −
have *Q-degen*: ⋀x. ¬ Q x x
unfolding *Q-def Let-def*
by (*simp add*: *cross3-simps*)

have *Q-sym*: ⋀x y. Q x y ⟹ Q y x
proof −
fix x y
assume Q x y
then obtain a b **where**
a-def: a = (y − v) × (x − v) **and**
a-nonzero: a ≠ *0* **and**
b-def: b = a · v **and**
ineq: (∀ w∈T. a · w ≤ b) ∨ (∀ w∈T. a · w ≥ b) **and**
plane: F = *convex hull* (T ∩ {w. a · w = b})
unfolding *Q-def Let-def*
by *auto*
let *?a′* = (x − v) × (y − v)
let *?b′* = *?a′* · v
have *?a′* = −a
unfolding *a-def* **using** *cross-skew*
by *blast*
have {w. *?a′* · w = *?b′*} = {w. a · w = b}
unfolding ‹*?a′* = −a› **using** *b-def*
by *auto*
then show Q y x
unfolding *Q-def Let-def*
using ‹*?a′* = −a› *a-nonzero plane ineq*
by *auto*
qed

show *?thesis*
using *set-choice-decomposition-2*[*of Q u S*] *Q-sym Q-degen Q-def*
by *blast*
qed

4.3.2 1-faces

lemma *compute-faces-1*:
fixes F n d **and** S :: (*real^3*) *set*
assumes ⋀x. x∈S ⟹ n · x = d
and *finite S*
and n ≠ *0*
shows (F *face-of* (*convex hull S*) ∧ *aff-dim F* = *1*) =

```
      (∃ x∈S. ∃ y∈S.
       let a = n × (y − x) in
        a ≠ 0 ∧
        (let b = a · x in
         ((∀ w∈S. a · w ≤ b) ∨ (∀ w∈S. a · w ≥ b)) ∧
         F = convex hull (S ∩ {w. a · w = b})))
  (is ?lhs = ?rhs)
proof
  assume ?lhs
  then have *: F face-of (convex hull S) aff-dim F = 1 by blast+
  then obtain T where T-facts: T ⊆ S F = convex hull T
    by (meson assms face-of-convex-hull-subset finite-imp-compact)
  then obtain U where U-facts: U ⊆ T ¬ affine-dependent U affine hull T =
affine hull U
    using affine-basis-exists[of T] by blast
  then have aff-dim U = 1
    by (metis *(2) T-facts(2) aff-dim-affine-hull aff-dim-convex-hull)
  then have card U = 2
    using U-facts(2) aff-dim-affine-independent by fastforce
  then obtain x y where U-eq: U = {x, y} and xy-neq: x ≠ y
    using card-2-iff[of U] by blast
  have xy-in-S: x ∈ S y ∈ S
    using U-eq U-facts(1) T-facts(1) by blast+

  let ?a = n × (y − x)
  let ?b = ?a · x
  let ?H = {w. ?a · w = ?b}

  have n · (y − x) = 0
    using assms(1)[OF xy-in-S(1)] assms(1)[OF xy-in-S(2)]
    by (simp add: inner-diff-right)
  then have ?a ≠ 0
    using norm-and-cross-eq-0[of n y − x] assms(3) xy-neq
    by simp
  have ?a · y = ?b
     by (metis (no-types, opaque-lifting) add.commute diff-0-right diff-add-cancel
dot-cross-self(2) inner-commute
        inner-diff-left)

  have F exposed-face-of (convex hull S)
    using *(1) polyhedron-convex-hull[OF assms(2)] exposed-face-of-polyhedron
    by blast
  then obtain a0 b0 where a0-b0-facts:
    convex hull S ⊆ {w. a0 · w ≤ b0}
    convex hull T = convex hull S ∩ {w. a0 · w = b0}
    unfolding exposed-face-of-def T-facts(2) by blast
  define a′ where a′ = a0 − ((a0 · n) / (n · n)) *R n
  define b′ where b′ = b0 − ((a0 · n) / (n · n)) * d
  have a′ · n = 0
```

```
  unfolding a'-def
  by (simp add: inner-diff-left dot-square-norm power2-eq-square)
have S ⊆ {w. n · w = d}
  using assms(1) by blast
have ⋀w. w ∈ convex hull S ⟹ (a' · w = b') = (a0 · w = b0)
proof −
  fix w assume w ∈ convex hull S
  then have n · w = d
    using ‹S ⊆ {w. n · w = d}›
    by (metis (mono-tags, lifting) assms(1) convex-hyperplane hull-induct)
  then show (a' · w = b') = (a0 · w = b0)
    unfolding a'-def b'-def by (simp add: inner-diff-left)
qed
moreover have ⋀w. w ∈ convex hull S ⟹ (a' · w ≤ b') = (a0 · w ≤ b0)
proof −
  fix w assume w ∈ convex hull S
  then have n · w = d
    using ‹S ⊆ {w. n · w = d}›
    by (metis (mono-tags, lifting) assms(1) convex-hyperplane hull-induct)
  then show (a' · w ≤ b') = (a0 · w ≤ b0)
    unfolding a'-def b'-def by (simp add: inner-diff-left)
qed
ultimately have a'-b'-facts:
  convex hull S ⊆ {w. a' · w ≤ b'}
  convex hull T = convex hull S ∩ {w. a' · w = b'}
  using a0-b0-facts by auto
let ?H' = {w. a' · w = b'}

have aff-dim (affine hull T) = 1
  by (metis U-facts(3) ‹aff-dim U = 1› aff-dim-affine-hull)
have {x, y} ⊆ ?H
  using ‹?a · y = ?b› by simp
then have affine hull {x, y} ⊆ ?H
  using affine-hyperplane hull-minimal by metis
then have affine hull T ⊆ ?H
  using U-eq U-facts(3) by blast
moreover have affine hull T ⊆ ?H'
  using a'-b'-facts(2)
  by (metis affine-hyperplane affine-imp-convex inf.cobounded2 subset-hull)
ultimately have affine hull T ⊆ ?H ∩ ?H'
  by blast
then have aff-dim (?H ∩ ?H') ≥ 1
  using aff-dim-subset ‹aff-dim (affine hull T) = 1›
  by metis

show ?rhs
proof (cases convex hull T = convex hull S)
  case True
  then have affine hull T = affine hull S
```

using *affine-hull-convex-hull* **by** *metis*
then have $S \subseteq$ *affine hull* T
using *T-facts*(*1*)
by (*simp add*: *hull-subset*)
then have $S \subseteq$ *?H*
using ‹*affine hull T* $\subseteq$ *?H*› **by** *blast*
then show *?thesis*
using *xy-in-S* ‹*?a* $\neq$ *0*› *True T-facts*(*2*) **unfolding** *Let-def*
by (*metis* (*mono-tags*, *lifting*) *Int-Collect dual-order.refl inf.orderE*)

next
case *False*
have *affine hull T* $\subseteq$ *?H'*
using *a'-b'-facts*(*2*) *T-facts*(*2*)
by (*metis affine-hyperplane affine-imp-convex inf.cobounded2 subset-hull*)

have *affine hull T* $\subseteq$ *?H* $\cap$ *?H'*
using ‹*affine hull T* $\subseteq$ *?H*› ‹*affine hull T* $\subseteq$ *?H'*› **by** *blast*

have *aff-dim ?H* = *2*
using ‹*?a* $\neq$ *0*› *aff-dim-hyperplane* **by** *simp*
have *a'* $\neq$ *0*
using $*$(*2*) *False Int-commute T-facts*(*2*) *a'-b'-facts*(*2*)
by *auto*
then have *aff-dim ?H'* = *2*
using *aff-dim-hyperplane* **by** *simp*

have *?H* $\subseteq$ *?H'*
proof (*rule ccontr*)
assume $\neg$(*?H* $\subseteq$ *?H'*)
then have *aff-dim* (*?H* $\cap$ *?H'*) = *1*
using *aff-dim-affine-Int-hyperplane*[*of ?H a' b'*]
‹*aff-dim ?H* = *2*› ‹*aff-dim ?H'* = *2*› ‹*aff-dim* (*?H* $\cap$ *?H'*) $\geq$ *1*›
by (*metis* (*no-types*, *lifting*) *ext affine-hyperplane diff-add-cancel diff-minus-eq-add diff-numeral-special*(*11*) *not-numeral-le-neg-one numeral-eq-one-iff*)
have $x + n \in$ *?H*
by (*simp add*: *dot-cross-self*(*4*) *inner-right-distrib*)
have $x \in$ *?H'*
using *U-eq U-facts*(*1*) *a'-b'-facts*(*2*) *hull-inc* **by** *fastforce*
have $n \cdot (x + n) \neq d$
by (*simp add*: *assms*(*1*,*3*) *inner-right-distrib xy-in-S*(*1*))
have *affine hull S* $\subseteq \{w.\ n \cdot w = d\}$
by (*simp add*: ‹$S \subseteq \{w.\ n \cdot w = d\}$› *affine-hyperplane subset-hull*)
then have *convex hull T* $\subseteq \{w.\ n \cdot w = d\}$
using *a'-b'-facts*(*2*) *convex-hull-subset-affine-hull* **by** *auto*
have $x + n \in$ *?H'*
using ‹$a' \cdot n = 0$›
by (*metis* ‹$x \in$ *?H'*› *add.commute add-0 inner-right-distrib mem-Collect-eq*)
have *?H* $\cap$ *?H'* $\subseteq \{w.\ n \cdot w = d\}$

by (*metis* (*mono-tags*, *lifting*) ‹*aff-dim* (*affine hull T*) = *1*› ‹*aff-dim* (*?H* ∩ *?H′*) = *1*›
‹*affine hull T* ⊆ *?H* ∩ *?H′*› ‹*convex hull T* ⊆ {*w*. *n* · *w* = *d*}› *aff-dim-eq-full-gen*
affine-hull-convex-hull affine-hyperplane subset-hull)
then show *False*
using ‹*x* + *n* ∈ *?H*› ‹*x* + *n* ∈ *?H′*› ‹*n* · (*x* + *n*) ≠ *d*›
by *blast*
qed
then have *?H* = *?H′*
using *subset-hyperplanes*
by (*metis* ‹*a′* ≠ *0*› ‹{*x*, *y*} ⊆ *?H*› *empty-iff hyperplane-eq-UNIV insert-subset*)

then have *F* = *convex hull* (*S* ∩ *?H*)
by (*smt* (*verit*, *ccfv-threshold*) *Int-commute T-facts*(*1*,*2*) *a′-b′-facts*(*2*) *convex-convex-hull*
hull-minimal hull-subset inf.orderE le-inf-iff subsetI)

moreover have (∀ *w*∈*S*. *?a* · *w* ≤ *?b*) ∨ (∀ *w*∈*S*. *?a* · *w* ≥ *?b*)
proof −
have *?H′* ⊆ *?H*
using ‹*?H* = *?H′*› **by** *auto*
then obtain *k* **where** *k-facts*: *?a* = *k* $*_R$ *a′* *?b* = *k* * *b′* *k* ≠ *0*
using ‹*?a* ≠ *0*› ‹*a′* ≠ *0*› *hyperplane-subset-imp-scaled* **by** *blast*
show *?thesis*
proof (*cases k* > *0*)
case *True*
then have ⋀*w*. *?a* · *w* ≤ *?b* ⟷ *a′* · *w* ≤ *b′*
using *k-facts* **by** (*auto simp*: *mult-le-cancel-left*)
then show *?thesis*
using *a′-b′-facts*(*1*) *hull-subset*[*of S*] **by** *auto*
next
case *False*
then have *k* < *0*
using *k-facts* **by** *simp*
then have ⋀*w*. *?a* · *w* ≥ *?b* ⟷ *a′* · *w* ≤ *b′*
using *k-facts* **by** (*auto simp*: *mult-le-cancel-left*)
then show *?thesis*
using *a′-b′-facts*(*1*) *hull-subset*[*of S*] **by** *auto*
qed
qed

ultimately show *?thesis*
using *xy-in-S* ‹*?a* ≠ *0*› **unfolding** *Let-def*
by *blast*
qed
next
assume *?rhs*
then obtain *x y* **where**

xy-in-S: $x \in S$ $y \in S$ **and**
a-neq-0: $n \times (y - x) \neq 0$ **and**
one-side: $(\forall w{\in}S.\ n \times (y - x) \cdot w \leq n \times (y - x) \cdot x) \vee$
$(\forall w{\in}S.\ n \times (y - x) \cdot w \geq n \times (y - x) \cdot x)$ **and**
F-eq: $F = convex\ hull\ (S \cap \{w.\ n \times (y - x) \cdot w = n \times (y - x) \cdot x\})$
unfolding *Let-def* **by** *blast*
let $?a = n \times (y - x)$
let $?b = ?a \cdot x$
let $?H = \{w.\ ?a \cdot w = ?b\}$
have $convex\ hull\ (S \cap ?H) = convex\ hull\ S \cap ?H$
proof
show $convex\ hull\ (S \cap ?H) \subseteq convex\ hull\ S \cap ?H$
using *convex-hyperplane*[*of ?a ?b*] *hull-mono*[*of S* $\cap$ *?H S convex*]
subset-hull[*of convex ?H S* $\cap$ *?H*]
by *blast*
next
show $convex\ hull\ S \cap ?H \subseteq convex\ hull\ (S \cap ?H)$
proof (*cases* $S \subseteq ?H$)
case *True*
then show *?thesis* **by** (*simp add*: *Int-absorb2*)
next
case *False*
let $?S\text{-}on = S \cap ?H$
let $?S\text{-}off = S - ?H$
have $convex\ hull\ S \cap ?H = convex\ hull\ (convex\ hull\ (?S\text{-}on) \cup convex\ hull\ (?S\text{-}off)) \cap ?H$
by (*metis Int-Diff-Un hull-Un-left hull-Un-right*)
moreover have $convex\ hull\ (convex\ hull\ (?S\text{-}on) \cup convex\ hull\ (?S\text{-}off)) \cap ?H \subseteq convex\ hull\ (S \cap ?H)$
proof
fix w
assume $*$: $w \in convex\ hull\ (convex\ hull\ (?S\text{-}on) \cup convex\ hull\ (?S\text{-}off)) \cap ?H$
have *convex*: $convex\ (convex\ hull\ (?S\text{-}on))\ convex\ (convex\ hull\ (?S\text{-}off))$
using *convex-convex-hull* **by** *blast+*
have *nonempty1*: $convex\ hull\ ?S\text{-}on \neq \{\}$
using *xy-in-S*(*1*) *convex-hull-eq-empty* **by** *blast*
have *nonempty2*: $convex\ hull\ ?S\text{-}off \neq \{\}$
using *False convex-hull-eq-empty* **by** *simp*
obtain $u\ p\ q$ **where**
w-facts: $w = (1 - u) *_R p + u *_R q$ $?a \cdot w = ?b$ **and**
p-in: $p \in convex\ hull\ ?S\text{-}on$ **and**
q-in: $q \in convex\ hull\ ?S\text{-}off$ **and**
u-range: $0 \leq u$ $u \leq 1$
using $*$ *convex-hull-union-nonempty-explicit*[*OF convex*(*1*) *nonempty1* *convex*(*2*) *nonempty2*]
by *blast*
show $w \in convex\ hull\ (S \cap ?H)$
proof (*cases* $u = 0$)

```
    case True
    then show ?thesis
      using p-in w-facts(1) by simp
  next
    case u-nonzero: False
    have p ∈ convex hull ?H
      using p-in hull-mono[of ?S-on ?H] by blast
    then have ?a · p = ?b
      using convex-hyperplane hull-same
      by blast
    have q ∈ convex hull ?S-off
      using q-in by simp

    have q ∉ ?H
    proof
      assume q ∈ ?H
      then have ?a · q = ?b by simp
      consider
        (leq) ∀ w∈S. ?a · w ≤ ?b |
        (geq) ∀ w∈S. ?a · w ≥ ?b
        using one-side by blast
      then show False
      proof cases
        case leq
        then have ?S-off ⊆ {w. ?a · w < ?b}
          by fastforce
        then have convex hull ?S-off ⊆ {w. ?a · w < ?b}
          using convex-halfspace-lt hull-minimal by metis
        then show False
          using ‹?a · q = ?b› ‹q ∈ convex hull ?S-off› by auto
      next
        case geq
        then have ?S-off ⊆ {w. ?a · w > ?b}
          by fastforce
        then have convex hull ?S-off ⊆ {w. ?a · w > ?b}
          using convex-halfspace-gt hull-minimal by metis
        then show False
          using ‹?a · q = ?b› ‹q ∈ convex hull ?S-off› by auto
      qed
    qed
    then have q-not-in-H: ?a · q ≠ ?b by simp

    have ?a · ((1 − u) *R p + u *R q) = (1 − u) * (?a · p) + u * (?a · q)
      by (simp add: inner-add-right)
    then show ?thesis
      using q-not-in-H ‹q ∈ convex hull ?S-off›
    by (metis ‹?a · p = ?b› real-scaleR-def segment-degen-0 u-nonzero w-facts)
  qed
qed
```

ultimately show *?thesis* **by** *argo*
qed
qed
then have *F-as-inter*: $F = convex\ hull\ S \cap ?H$
using *F-eq* **by** *simp*
consider (*case-le*) $\forall w \in S.\ ?a \cdot w \leq ?b$ | (*case-ge*) $\forall w \in S.\ ?a \cdot w \geq ?b$
using *one-side* **by** *blast*
then have *face-of*: *F face-of* (*convex hull S*)
proof *cases*
case *case-le*
then have *subset*: $S \subseteq \{x.\ ?a \cdot x \leq ?b\}$
by *blast*
have *convex hull* $S \subseteq \{x.\ ?a \cdot x \leq ?b\}$
using *convex-halfspace-le*[*of ?a ?b*] *hull-minimal*[*OF subset*, *of convex*]
by *blast*
then have $\bigwedge w.\ w \in convex\ hull\ S \Longrightarrow ?a \cdot w \leq ?b$
by *blast*
then have *convex hull S* $\cap$ *?H face-of convex hull S*
using *face-of-Int-supporting-hyperplane-le*[*OF convex-convex-hull*[*of S*]]
by *blast*
then show *?thesis*
using *F-as-inter* **by** *simp*
next
case *case-ge*
then have *subset*: $S \subseteq \{x.\ ?a \cdot x \geq ?b\}$
by *blast*
have *convex hull* $S \subseteq \{x.\ ?a \cdot x \geq ?b\}$
using *convex-halfspace-ge*[*of ?b ?a*] *hull-minimal*[*OF subset*, *of convex*]
by *blast*
then have $\bigwedge w.\ w \in convex\ hull\ S \Longrightarrow ?a \cdot w \geq ?b$
by *blast*
then have *convex hull S* $\cap$ *?H face-of convex hull S*
using *face-of-Int-supporting-hyperplane-ge*[*OF convex-convex-hull*[*of S*]]
by *blast*
then show *?thesis*
using *F-as-inter* **by** *simp*
qed

have $S \subseteq \{x.\ n \cdot x = d\}$
using *assms*(*1*) **by** *blast*
have *aff-dim ?H* = *2*
using *aff-dim-hyperplane*[*OF a-neq-0*, *of ?b*] **by** *simp*
have $\neg(?H \subseteq \{v.\ n \cdot v = d\})$
proof –
have $\exists v.\ n \cdot v \neq d \wedge n \times (y - x) \cdot v = x \cdot n \times (y - x)$
by (*metis* (*no-types*) *assms*(*3*) *diff-0-right diff-add-cancel dot-cross-self*(*1*)
inner-commute
inner-diff-left inner-eq-zero-iff minus-diff-eq)
then show *?thesis*

by (*smt* (*z3*) *Collect-mono-iff inner-commute*)
qed
then have *aff-dim* (*?H* ∩ {*x*. *n* · *x* = *d*}) ≤ *1*
using *aff-dim-affine-Int-hyperplane*[*of ?H n d*] **unfolding** ‹*aff-dim ?H* = *2*›
by (*simp add*: *affine-hyperplane*)
moreover have *F* ⊆ (*?H* ∩ {*x*. *n* · *x* = *d*})
using *F-as-inter assms*(*1*) ‹*S* ⊆ {*x*. *n* · *x* = *d*}›
convex-hyperplane[*of n d*] *subset-hull*[*of convex* {*x*. *n* · *x* = *d*} *S*]
by *blast*
ultimately have *upper*: *aff-dim F* ≤ *1*
using *aff-dim-subset*[*of F ?H* ∩ {*x*. *n* · *x* = *d*}] **unfolding** ‹*aff-dim ?H* = *2*›
by *linarith*
have *x* ≠ *y*
using *a-neq-0* **by** *auto*
moreover have *?a* · *y* = *?b*
by (*metis* (*no-types*, *opaque-lifting*) *dot-cross-self*(*2*) *inner-commute inner-diff-left right-minus-eq*)
moreover have {*x*, *y*} ⊆ *F*
using *xy-in-S*
by (*simp add*: *F-as-inter calculation*(*2*) *hull-inc*)
moreover have *aff-dim* {*x*, *y*} = *1*
by (*simp add*: *calculation*(*1*))
ultimately have *lower*: *aff-dim F* ≥ *1*
using *aff-dim-subset* **by** *metis*

show *?lhs*
using *face-of upper lower* **by** *simp*
qed

lemma *compute-faces-1-step*:
fixes *F n u S* **and** *T* :: (*real*^*3*) *set*
defines *Q* ≡ (λ*x y*. *let a* = *n* × (*y* − *x*) *in*
a ≠ *0* ∧
(*let b* = *a* · *x in*
((∀ *w*∈*T*. *a* · *w* ≤ *b*) ∨ (∀ *w*∈*T*. *a* · *w* ≥ *b*)) ∧
F = *convex hull* (*T* ∩ {*k*. *a* · *k* = *b*})))
shows (∃ *y* ∈ (*insert u S*). ∃ *z* ∈ (*insert u S*). *Q y z*) =
((∃ *z*∈*S*. *Q u z*) ∨ (∃ *y*∈*S*. ∃ *z*∈*S*. *Q y z*))
(**is** *?lhs* = *?rhs*)
proof −
have *Q-degen*: ⋀*x*. ¬ *Q x x*
unfolding *Q-def Let-def*
by (*simp add*: *cross3-simps*)

have *Q-sym*: ⋀*x y*. *Q x y* ⟹ *Q y x*
proof −
fix *x y*
assume *Q x y*
then obtain *a b* **where**

```
      a-def: a = n × (y − x) and
      a-nonzero: a ≠ 0 and
      b-def: b = a · x and
      ineq: (∀ w∈T. a · w ≤ b) ∨ (∀ w∈T. a · w ≥ b) and
      plane: F = convex hull (T ∩ {w. a · w = b})
      unfolding Q-def Let-def
      by auto
    let ?a′ = n × (x − y)
    let ?b′ = ?a′ · y
    have ?a′ = −a
      unfolding a-def Cross3.right-diff-distrib
      by auto
    have a · y − a · x = 0
      by (metis a-def dot-cross-self(3) inner-diff-right)
    then have {w. ?a′ · w = ?b′} = {w. a · w = b}
      unfolding ‹?a′ = −a› b-def
      by auto
    then show Q y x
      unfolding Q-def Let-def
      using ‹?a′ = −a› a-nonzero plane ineq
      by auto
  qed

  show ?thesis
    using set-choice-decomposition-2[of Q u S] Q-sym Q-degen Q-def
    by blast
qed

method facet-edges uses plane n-nz defs =
  (simp only: compute-faces-1[OF plane - n-nz] finite-insert finite.emptyI,
   simp only: compute-faces-1-step bex-empty simp-thms(31),
  simp only: empty-iff ball-insert bex-empty bex-simps(5) ball-simps(5) simp-thms(21,31),
   simp add: defs vector3-sub vector3-cross vector3-dot vector3-eq-0 Let-def
           rat5-mul rat5-add rat5-sub rat5-eq rat5-eq-0 rat5-le)
```

4.4 Full-dimensionality

```
lemma polytope-3D:
  fixes S :: (real^3) set
  assumes (z − x) × (y − x) ≠ 0
      and ∃ w∈S. ((z − x) × (y − x)) · w ≠ ((z − x) × (y − x)) · x
  shows aff-dim (convex hull (insert x (insert y (insert z S)))) = 3
  (is aff-dim ?hull = 3)
proof −
  have aff-dim ?hull ≤ int (DIM(real^3))
    by (rule aff-dim-le-DIM)
  then have upper: aff-dim ?hull ≤ 3
    by auto
  let ?a = (z − x) × (y − x)
```

obtain *w* **where** *w-facts*: $w \in S$ *?a* $\cdot$ $w \neq$ *?a* $\cdot$ *x*
  using *assms*(*2*) **by** *blast*
have $\{w, x, y, z\} \subseteq$ *?hull*
  using *hull-subset w-facts*(*1*) **by** *fastforce*
have *aff-dim* $\{w, x, y, z\} \geq 3$
proof (*cases* $w \in$ *affine hull* $\{x, y, z\}$)
  case *True*
  have *?a* $\cdot$ $y =$ *?a* $\cdot$ *x* *?a* $\cdot$ $z =$ *?a* $\cdot$ *x*
  by (*metis* (*no-types*, *lifting*) *Cross3.left-diff-distrib cross-refl cross-skew cross-triple diff-0 diff-0-right*)+
  then have $\{x,y,z\} \subseteq \{v.$ *?a* $\cdot$ $v =$ *?a* $\cdot$ $x\}$
    by *blast*
  then have $w \in \{v.$ *?a* $\cdot$ $v =$ *?a* $\cdot$ $x\}$
    using *True affine-hyperplane*[*of ?a ?a* $\cdot$ *x*]
      subset-hull[*of affine* $\{v.$ *?a* $\cdot$ $v =$ *?a* $\cdot$ $x\}$ $\{x, y, z\}$]
    by *blast*
  then show *?thesis*
    using *w-facts*(*2*) **by** *blast*
next
  case *False*
  have $\neg$ *collinear* $\{x, y, z\}$
    by (*metis* (*no-types*, *opaque-lifting*) *assms*(*1*) *NO-MATCH-def collinear-3 cross-eq-0*
      insert-commute)
  then show *?thesis*
    using *False aff-dim-insert*
    by (*smt* (*verit*, *ccfv-threshold*) *collinear-aff-dim*)
qed
then have *aff-dim ?hull* ≥ 3
  by (*meson* ‹$\{w, x, y, z\} \subseteq$ *?hull*› *aff-dim-subset order-trans*)
then show *?thesis*
  using *upper* **by** *simp*
qed

4.5 Relations between faces of different dimensions

These lemmas are used when formalizing the number of edges per face and number of edges per vertex.

corollary *edge-belongs-to-face*:
  assumes *e face-of p*
    and *aff-dim* $e = 1$
    and *aff-dim* $p = 3$
    and *polyhedron p*
  obtains *f* **where** *f face-of p aff-dim* $f = 2$ *e face-of f*
  using *assms*(*2*,*3*) *ridge-belongs-to-facet*[*OF assms*(*1*) - - *assms*(*4*)]
  by *auto*

corollary *vertex-belongs-to-edge*:
  assumes *v face-of p*

and *aff-dim v = 0*
and *aff-dim p = 3*
and *polyhedron p*
obtains *e* **where** *e face-of p aff-dim e = 1 v face-of e*
using *assms(2,3) peak-belongs-to-ridge[OF assms(1) - - assms(4)]*
by *auto*

lemma *vertices-of-edge*:
assumes *e = convex hull {a, b}*
and *v face-of e*
and *aff-dim v = 0*
shows *v = {a} ∨ v = {b}*
by *(metis assms aff-dim-eq-0 extreme-point-of-convex-hull-2 face-of-singleton)*

lemma *vertex-edge-set-eq*:
assumes *v face-of S*
shows *card {e. e face-of S ∧ aff-dim e = 1 ∧ v ⊆ e} =*
card {e. e face-of S ∧ aff-dim e = 1 ∧ v face-of e}
by *(meson assms face-of-face)*

lemma *face-edge-set-eq*:
assumes *f face-of S*
shows *card {e. e face-of S ∧ aff-dim e = 1 ∧ e ⊆ f} =*
card {e. e face-of f ∧ aff-dim e = 1}
using *face-of-face[OF assms(1)]*
by *meson*

lemma *set-disj-union-distr*:
shows *{e. (P e ∨ Q e) ∧ R e} = {e. P e ∧ R e} ∪ {e. Q e ∧ R e}*
by *blast*

lemma *v-sing-face-of-e*:
shows *(e = a ∧ {x} face-of e) = (e = a ∧ x extreme-point-of a)*
using *face-of-singleton* **by** *blast*

lemma *card-3-iff2*:
shows *(card {x, y, z} = 3) = (x ≠ y ∧ x ≠ z ∧ y ≠ z)*
by *(simp add: card-insert-if)*

lemma *card-4-iff2*:
shows *(card {w, x, y, z} = 4) = (w ≠ x ∧ w ≠ y ∧ w ≠ z ∧ x ≠ y ∧ x ≠ z ∧ y ≠ z)*
by *(simp add: card-insert-if)*

lemma *card-5-iff*:
shows *card {u,w,x,y,z} = 5 ⟷ (u ≠ w ∧ u ≠ x ∧ u ≠ y ∧ u ≠ z ∧ w ≠ x ∧ w ≠ y ∧ w ≠ z ∧ x ≠ y ∧ y ≠ z ∧ x ≠ z)*
by *(simp add: card-insert-if)*

method *edges-per-vertex* =
(*unfold set-disj-union-distr segment-convex-hull*[*symmetric*] *v-sing-face-of-e*;
simp-all add: *extreme-point-of-segment vector3-eq rat5-eq*;
unfold card-3-iff2 card-4-iff2 card-5-iff closed-segment-eq;
simp-all add: *vector3-eq rat5-eq doubleton-eq-iff*)

lemma *set-cases*:
shows $\{x.\ x = a\} = \{a\}$ $\{x.\ x = a \vee P\ x\} = insert\ a\ \{x.\ P\ x\}$
by *auto*

4.6 Congruence

definition *congruent* :: $['a{::}euclidean\text{-}space\ set,\ 'a\ set] \Rightarrow bool$ (**infixr** ‹(*congruent*)› *50*)
where S *congruent* $T \longleftrightarrow (\exists\, c\ f.\ orthogonal\text{-}transformation\ f \wedge T = (\lambda x.\ c + f\ x)\ `\ S)$

lemma *congruent-refl*:
shows S *congruent* S
unfolding *congruent-def*
using *orthogonal-transformation-id image-add-0*
by *blast*

lemma *congruent-sym*:
shows S *congruent* $T \longleftrightarrow T$ *congruent* S
proof −
have *forward*: $\bigwedge S\ T.\ S$ *congruent* $T \Longrightarrow T$ *congruent* S
proof −
fix $S\ T :: 'b\ set$
assume S *congruent* T
then obtain $c\ f$ **where** *orthog*: *orthogonal-transformation f* **and** T: $T = (\lambda x.\ c + f\ x)\ `\ S$
unfolding *congruent-def* **by** *blast*
then have *orthog-inv*: *orthogonal-transformation* (*inv f*)
using *orthogonal-transformation-inv* **by** *blast*
have *bounded-linear* (*inv f*)
using *orthogonal-transformation-linear*[*OF orthog-inv*]
by (*simp add*: *linear-conv-bounded-linear*)

have $(\lambda y.\ -\ inv\ f\ c + inv\ f\ y)\ `\ T = (\lambda x.\ -inv\ f\ c + inv\ f\ (c + f\ x))\ `\ S$
using T **by** *blast*
also have $\ldots = (\lambda x.\ -inv\ f\ c + inv\ f\ c + inv\ f\ (f\ x))\ `\ S$
using ‹*bounded-linear* (*inv f*)›
by (*simp add*: *linear-simps*)
also have $\ldots = (\lambda x.\ inv\ f\ (f\ x))\ `\ S$
by *simp*
finally have $S = (\lambda y.\ -\ inv\ f\ c + inv\ f\ y)\ `\ T$
by (*simp add*: *orthog bijection.intro bijection.inv-left orthogonal-transformation-bij*)
then show T *congruent* S

unfolding *congruent-def* **using** *orthog-inv*
by *blast*
qed
then show *?thesis*
by *blast*
qed

lemma *congruent-trans*:
assumes *R congruent S*
assumes *S congruent T*
shows *R congruent T*
proof –
obtain *f c* **where** *orthog-f*: *orthogonal-transformation f* **and** *S*: $S = (\lambda x.\ c + f\ x)$ ‘ R
using *assms*(*1*) **unfolding** *congruent-def* **by** *blast*
obtain *g d* **where** *orthog-g*: *orthogonal-transformation g* **and** *T*: $T = (\lambda x.\ d + g\ x)$ ‘ S
using *assms*(*2*) **unfolding** *congruent-def* **by** *blast*
then have $T = (\lambda x.\ d + g\ (c + f\ x))$ ‘ R
unfolding *S* **by** *blast*
then have $T = (\lambda x.\ d + g\ c + g\ (f\ x))$ ‘ R
using *orthogonal-transformation-linear*[*OF orthog-g*] *linear-conv-bounded-linear*[*of g*]
linear-simps(*1*)
by *fastforce*
moreover have *orthogonal-transformation* (*g o f*)
using *orthog-f orthog-g orthogonal-transformation-compose*
by *blast*
ultimately show *?thesis*
unfolding *congruent-def*
by *fastforce*
qed

lemma *congruent-set*:
assumes $\forall t \in (insert\ s\ S).\ s\ congruent\ t$
assumes $x \in (insert\ s\ S)$
assumes $y \in (insert\ s\ S)$
shows *x congruent y*
using *assms congruent-sym congruent-trans*
by *metis*

4.6.1 Congruence of segments

lemma *congruent-segments*:
fixes *a b c d* :: *real^'n*
assumes *dist a b = dist c d*
shows (*closed-segment a b*) *congruent* (*closed-segment c d*)
proof –
have $norm\ (b - a) = norm\ (d - c)$

using *assms*
by (*simp add*: *dist-norm norm-minus-commute*)
then obtain *f* **where** *f-facts*: *orthogonal-transformation f f* $(b - a) = d - c$
using *orthogonal-transformation-exists*
by *blast*
have *linear f*
using *orthogonal-transformation-linear*[*OF f-facts*(*1*)] **.**
have $d = (c - f\ a) + f\ b$
using *linear-diff* [*OF* ‹*linear f*›, *of b a*] **unfolding** *f-facts*(*2*)
by (*metis add.left-commute diff-add-cancel*)
moreover have $(\lambda x.\ (c - f\ a) + f\ x) = (\lambda x.\ (c - f\ a) + x)$ *o f*
by *auto*
ultimately have $(\lambda x.\ (c - f\ a) + f\ x)$ ‘ *closed-segment a b* =
$((\lambda x.\ (c - f\ a) + x) \circ f)$ ‘ (*convex hull* $\{a, b\}$)
unfolding *segment-convex-hull*[*of a b*]
by *argo*
also have ... = $(\lambda x.\ (c - f\ a) + x)$ ‘ (*convex hull* (*f* ‘ $\{a, b\}$))
using *image-comp convex-hull-linear-image*[*OF* ‹*linear f*›]
by *metis*
also have ... = *convex hull* $\{(c - f\ a) + f\ a, (c - f\ a) + f\ b\}$
by (*metis* (*no-types*, *opaque-lifting*) *convex-hull-translation*
image-empty image-insert)
also have ... = *convex hull* $\{c, d\}$
unfolding ‹$d = (c - f\ a) + f\ b$› **by** *auto*
finally show *?thesis*
unfolding *congruent-def segment-convex-hull*[*of c d*]
using *f-facts*(*1*)
by *blast*
qed

4.6.2 Congruence of 2-faces

lemma *congruent-simple*:
assumes $\exists$ *A*::*real^3^3*. *orthogonal-matrix A* $\wedge$ $(\lambda x.\ A *v\ x)$ ‘ $S = T$
shows (*convex hull S*) *congruent* (*convex hull T*)
proof −
obtain *A* **where** *A-facts*: *orthogonal-matrix A* $(\lambda x.\ A *v\ x)$ ‘ $S = T$
using *assms* **by** *blast*
then have *convex hull T* = $(\lambda x.\ A *v\ x)$ ‘ (*convex hull S*)
using *convex-hull-linear-image* **by** *blast*
moreover have *orthogonal-transformation* $(\lambda x.\ A *v\ x)$
using *A-facts*(*1*) *orthogonal-transformation-matrix*
by *fastforce*
ultimately show *?thesis*
unfolding *congruent-def*
by (*metis image-add-0*[*of convex hull T*]
image-image[*of* (+) *0* (*v) *A convex hull S*])
qed

lemma *matrix-vector-mul-3*:
  shows (*vector* [*vector*[*a11*, *a12*, *a13*],
                  vector[*a21*, *a22*, *a23*],
                  vector[*a31*, *a32*, *a33*]]::*real^3^3*) *∗v* *vector* [*x1*, *x2*, *x3*] =
          vector [*a11* ∗ *x1* + *a12* ∗ *x2* + *a13* ∗ *x3*,
                 a21 ∗ *x1* + *a22* ∗ *x2* + *a23* ∗ *x3*,
                 a31 ∗ *x1* + *a32* ∗ *x2* + *a33* ∗ *x3*]
  by (*simp add*: *vec-eq-iff matrix-vector-mult-def forall-3 sum-3*)

lemma *matrix-lemma*:
  fixes *A* :: *real^3^3*
  shows (*A* ∗*v* *x1* = *x2* ∧ *A* ∗*v* *y1* = *y2* ∧ *A* ∗*v* *z1* = *z2*) =
        ((*vector* [*x1*, *y1*, *z1*]::*real^3^3*) ∗*v* (*row 1 A*) = *vector*[*x2*\$*1*, *y2*\$*1*, *z2*\$*1*] ∧
         (*vector* [*x1*, *y1*, *z1*]::*real^3^3*) ∗*v* (*row 2 A*) = *vector*[*x2*\$*2*, *y2*\$*2*, *z2*\$*2*] ∧
          (*vector* [*x1*, *y1*, *z1*]::*real^3^3*) ∗*v* (*row 3 A*) = *vector*[*x2*\$*3*, *y2*\$*3*, *z2*\$*3*])
 by (*simp add*: *vec-eq-iff matrix-vector-mult-def row-def forall-3 sum-3 mult.commute conj-comms*)

lemma *matrix-by-cramer-lemma*:
  fixes *A* :: *real^3^3*
  assumes *det* (*vector*[*x1*, *y1*, *z1*]::*real^3^3*) ≠ *0*
  shows (*A* ∗*v* *x1* = *x2* ∧ *A* ∗*v* *y1* = *y2* ∧ *A* ∗*v* *z1* = *z2*) =
          (*A* = (χ *m k*. *det* ((χ *i j*.
                          if j = *k*
                          then (*vector* [*x2*\$*m*, *y2*\$*m*, *z2*\$*m*]::*real^3*)\$*i*
                          else (*vector* [*x1*, *y1*, *z1*]::*real^3^3*)\$*i*\$*j*)
                         ::*real^3^3*) /
                   det (*vector* [*x1*, *y1*, *z1*]::*real^3^3*)))
  unfolding *matrix-lemma cramer*[*OF assms*(*1*)]
  by (*simp add*: *vec-eq-iff row-def forall-3*)

definition *mk-matrix33* :: (*real^3*) ⇒ (*real^3*) ⇒ (*real^3*) ⇒ (*real^3*) ⇒ (*real^3*) ⇒ (*real^3*) ⇒ *real^3^3*
 where *mk-matrix33 x1 y1 z1 x2 y2 z2* = (*let d* = *det* (*vector*[*x1*, *y1*, *z1*]::*real^3^3*) *in vector* [*vector* [
          (*x2*\$*1* ∗ *y1*\$*2* ∗ *z1*\$*3* +
           x1\$*2* ∗ *y1*\$*3* ∗ *z2*\$*1* +
           x1\$*3* ∗ *y2*\$*1* ∗ *z1*\$*2* −
           x2\$*1* ∗ *y1*\$*3* ∗ *z1*\$*2* −
           x1\$*2* ∗ *y2*\$*1* ∗ *z1*\$*3* −
           x1\$*3* ∗ *y1*\$*2* ∗ *z2*\$*1*) / *d*,
          (*x1*\$*1* ∗ *y2*\$*1* ∗ *z1*\$*3* +
           x2\$*1* ∗ *y1*\$*3* ∗ *z1*\$*1* +
           x1\$*3* ∗ *y1*\$*1* ∗ *z2*\$*1* −
           x1\$*1* ∗ *y1*\$*3* ∗ *z2*\$*1* −
           x2\$*1* ∗ *y1*\$*1* ∗ *z1*\$*3* −
           x1\$*3* ∗ *y2*\$*1* ∗ *z1*\$*1*) / *d*,
          (*x1*\$*1* ∗ *y1*\$*2* ∗ *z2*\$*1* +
           x1\$*2* ∗ *y2*\$*1* ∗ *z1*\$*1* +

```
      x2$1 ∗ y1$1 ∗ z1$2 −
      x1$1 ∗ y2$1 ∗ z1$2 −
      x1$2 ∗ y1$1 ∗ z2$1 −
      x2$1 ∗ y1$2 ∗ z1$1) / d], vector [
     (x2$2 ∗ y1$2 ∗ z1$3 +
      x1$2 ∗ y1$3 ∗ z2$2 +
      x1$3 ∗ y2$2 ∗ z1$2 −
      x2$2 ∗ y1$3 ∗ z1$2 −
      x1$2 ∗ y2$2 ∗ z1$3 −
      x1$3 ∗ y1$2 ∗ z2$2) / d,
     (x1$1 ∗ y2$2 ∗ z1$3 +
      x2$2 ∗ y1$3 ∗ z1$1 +
      x1$3 ∗ y1$1 ∗ z2$2 −
      x1$1 ∗ y1$3 ∗ z2$2 −
      x2$2 ∗ y1$1 ∗ z1$3 −
      x1$3 ∗ y2$2 ∗ z1$1) / d,
     (x1$1 ∗ y1$2 ∗ z2$2 +
      x1$2 ∗ y2$2 ∗ z1$1 +
      x2$2 ∗ y1$1 ∗ z1$2 −
      x1$1 ∗ y2$2 ∗ z1$2 −
      x1$2 ∗ y1$1 ∗ z2$2 −
      x2$2 ∗ y1$2 ∗ z1$1) / d], vector [
     (x2$3 ∗ y1$2 ∗ z1$3 +
      x1$2 ∗ y1$3 ∗ z2$3 +
      x1$3 ∗ y2$3 ∗ z1$2 −
      x2$3 ∗ y1$3 ∗ z1$2 −
      x1$2 ∗ y2$3 ∗ z1$3 −
      x1$3 ∗ y1$2 ∗ z2$3) / d,
     (x1$1 ∗ y2$3 ∗ z1$3 +
      x2$3 ∗ y1$3 ∗ z1$1 +
      x1$3 ∗ y1$1 ∗ z2$3 −
      x1$1 ∗ y1$3 ∗ z2$3 −
      x2$3 ∗ y1$1 ∗ z1$3 −
      x1$3 ∗ y2$3 ∗ z1$1) / d,
     (x1$1 ∗ y1$2 ∗ z2$3 +
      x1$2 ∗ y2$3 ∗ z1$1 +
      x2$3 ∗ y1$1 ∗ z1$2 −
      x1$1 ∗ y2$3 ∗ z1$2 −
      x1$2 ∗ y1$1 ∗ z2$3 −
      x2$3 ∗ y1$2 ∗ z1$1) / d]])

lemma set3-neqI:
  assumes a ∉ {x, y, z} ∨ b ∉ {x, y, z} ∨ c ∉ {x, y, z}
  shows {a, b, c} ≠ {x, y, z}
  using assms by auto

lemma set4-neqI:
  assumes a ∉ {w, x, y, z} ∨ b ∉ {w, x, y, z} ∨ c ∉ {w, x, y, z} ∨ d ∉ {w, x, y,
z}
```

shows $\{a, b, c, d\} \neq \{w, x, y, z\}$
using *assms* **by** *auto*

lemma *orthogonal-matrix33*:
fixes Q :: *real^3^3*
shows (*orthogonal-matrix* Q) =
$(Q\$1\$1 * Q\$1\$1 + Q\$2\$1 * Q\$2\$1 + Q\$3\$1 * Q\$3\$1 = 1 \wedge$
$Q\$1\$1 * Q\$1\$2 + Q\$2\$1 * Q\$2\$2 + Q\$3\$1 * Q\$3\$2 = 0 \wedge$
$Q\$1\$1 * Q\$1\$3 + Q\$2\$1 * Q\$2\$3 + Q\$3\$1 * Q\$3\$3 = 0 \wedge$
$Q\$1\$2 * Q\$1\$1 + Q\$2\$2 * Q\$2\$1 + Q\$3\$2 * Q\$3\$1 = 0 \wedge$
$Q\$1\$2 * Q\$1\$2 + Q\$2\$2 * Q\$2\$2 + Q\$3\$2 * Q\$3\$2 = 1 \wedge$
$Q\$1\$2 * Q\$1\$3 + Q\$2\$2 * Q\$2\$3 + Q\$3\$2 * Q\$3\$3 = 0 \wedge$
$Q\$1\$3 * Q\$1\$1 + Q\$2\$3 * Q\$2\$1 + Q\$3\$3 * Q\$3\$1 = 0 \wedge$
$Q\$1\$3 * Q\$1\$2 + Q\$2\$3 * Q\$2\$2 + Q\$3\$3 * Q\$3\$2 = 0 \wedge$
$Q\$1\$3 * Q\$1\$3 + Q\$2\$3 * Q\$2\$3 + Q\$3\$3 * Q\$3\$3 = 1)$
by (*simp add*: *orthogonal-matrix transpose-def matrix-matrix-mult-def sum-3 mat-def vec-eq-iff forall-3*)

lemma *matrix-by-cramer*:
fixes A :: *real^3^3*
assumes A = *mk-matrix33 x1 y1 z1 x2 y2 z2*
assumes *det* (*vector* [*x1*, *y1*, *z1*]::*real^3^3*) $\neq 0$
shows A **v x1* = *x2* A **v y1* = *y2* A **v z1* = *z2*
using *assms*(*1*) *matrix-by-cramer-lemma*[*OF assms*(*2*)]
unfolding *mk-matrix33-def Let-def*
by (*auto simp add*: *det-3 vec-eq-iff forall-3*)

lemma *matrix-by-cramer-orthog*:
fixes *x1 y1 z1 x2 y2 z2* :: *real^3*
assumes *orthogonal-matrix* (*mk-matrix33 x1 y1 z1 x2 y2 z2*)
assumes *det* (*vector* [*x1*, *y1*, *z1*]::*real^3^3*) $\neq 0$
shows *orthogonal-matrix* (*mk-matrix33 x1 y1 z1 x2 y2 z2*) $\wedge$
(**v*) (*mk-matrix33 x1 y1 z1 x2 y2 z2*) ‘ {*x1*, *y1*, *z1*} = {*x2*, *y2*, *z2*}
using *assms*(*1*) *matrix-by-cramer*[*OF - assms*(*2*)] **by** *auto*

Lookup table mapping a face to three vertices on that face that are adjacent (vertices that define a pair of intersecting edges)

definition *adjacency-table* :: ((*real*, *3*) *vec set* × ((*real*, *3*) *vec* × (*real*, *3*) *vec* × (*real*, *3*) *vec*)) *list* **where**
adjacency-table = [
({*vector*[−*1*, −*1*, *1*], *vector*[−*1*, *1*, −*1*], *vector*[*1*, −*1*, −*1*]}, (*vector* [−*1*, −*1*, *1*], *vector* [−*1*, *1*, −*1*], *vector* [*1*, −*1*, −*1*])),
({*vector*[−*1*, −*1*, *1*], *vector*[−*1*, *1*, −*1*], *vector*[*1*, *1*, *1*]}, (*vector*[−*1*, −*1*, *1*], *vector*[−*1*, *1*, − *1*], *vector*[*1*, *1*, *1*])),
({*vector*[−*1*, −*1*, *1*], *vector*[*1*, − *1*, − *1*], *vector*[*1*, *1*, *1*]}, (*vector*[−*1*, −*1*, *1*], *vector*[*1*, − *1*, − *1*], *vector*[*1*, *1*, *1*])),
({*vector*[−*1*, *1*, −*1*], *vector*[*1*, − *1*, − *1*], *vector*[*1*, *1*, *1*]}, (*vector*[−*1*, *1*, −*1*], *vector*[*1*, − *1*, − *1*], *vector*[*1*, *1*, *1*])),

({vector[−1, −1, −1], vector[−1, −1, 1], vector[−1, 1, −1], vector[−1, 1, 1]}, (vector[−1, −1, −1], vector[−1, −1, 1], vector[−1, 1, 1])),

({vector[−1, −1, −1], vector[−1, −1, 1], vector[1, −1, −1], vector[1, −1, 1]}, (vector[−1, −1, −1], vector[−1, −1, 1], vector[1, −1, 1])),

({vector[−1, −1, −1], vector[−1, 1, −1], vector[1, −1, −1], vector[1, 1, −1]}, (vector[−1, −1, −1], vector[−1, 1, −1], vector[1, 1, −1])),

({vector[−1, −1, 1], vector[−1, 1, 1], vector[1, −1, 1], vector[1, 1, 1]}, (vector[−1, −1, 1], vector[−1, 1, 1], vector[1, 1, 1])),

({vector[−1, 1, −1], vector[−1, 1, 1], vector[1, 1, −1], vector[1, 1, 1]}, (vector[−1, 1, −1], vector[−1, 1, 1], vector[1, 1, 1])),

({vector[1, −1, −1], vector[1, −1, 1], vector[1, 1, −1], vector[1, 1, 1]}, (vector[1, −1, −1], vector[1, −1, 1], vector[1, 1, 1])),

({vector[−1, 0, 0], vector[0, −1, 0], vector[0, 0, −1]}, (vector[−1, 0, 0], vector[0, −1, 0], vector[0, 0, −1])),

({vector[−1, 0, 0], vector[0, −1, 0], vector[0, 0, 1]}, (vector[−1, 0, 0], vector[0, −1, 0], vector[0, 0, 1])),

({vector[−1, 0, 0], vector[0, 1, 0], vector[0, 0, −1]}, (vector[−1, 0, 0], vector[0, 1, 0], vector[0, 0, −1])),

({vector[−1, 0, 0], vector[0, 1, 0], vector[0, 0, 1]}, (vector[−1, 0, 0], vector[0, 1, 0], vector[0, 0, 1])),

({vector[1, 0, 0], vector[0, −1, 0], vector[0, 0, −1]}, (vector[1, 0, 0], vector[0, −1, 0], vector[0, 0, −1])),

({vector[1, 0, 0], vector[0, −1, 0], vector[0, 0, 1]}, (vector[1, 0, 0], vector[0, −1, 0], vector[0, 0, 1])),

({vector[1, 0, 0], vector[0, 1, 0], vector[0, 0, −1]}, (vector[1, 0, 0], vector[0, 1, 0], vector[0, 0, −1])),

({vector[1, 0, 0], vector[0, 1, 0], vector[0, 0, 1]}, (vector[1, 0, 0], vector[0, 1, 0], vector[0, 0, 1])),

({vector[rat5 (−1/2) (−1/2), rat5 0 0, rat5 (−1/2) (1/2)], vector[rat5 (−1/2) (−1/2), rat5 0 0, rat5 (1/2) (−1/2)], vector[rat5 (1/2) (−1/2), rat5 (−1/2) (−1/2), rat5 0 0], vector[rat5 (−1) 0, rat5 (−1) 0, rat5 (−1) 0], vector[rat5 (−1) 0, rat5 (−1) 0, rat5 1 0]}, (vector[rat5 (−1/2) (−1/2), rat5 0 0, rat5 (−1/2) (1/2)], vector[rat5 (−1/2) (−1/2), rat5 0 0, rat5 (1/2) (−1/2)], vector[rat5 (−1) 0, rat5 (−1) 0, rat5 1 0])),

({vector[rat5 (−1/2) (−1/2), rat5 0 0, rat5 (−1/2) (1/2)], vector[rat5 (−1/2) (−1/2), rat5 0 0, rat5 (1/2) (−1/2)], vector[rat5 (1/2) (−1/2), rat5 (1/2) (1/2), rat5 0 0], vector[rat5 (−1) 0, rat5 1 0, rat5 (−1) 0], vector[rat5 (−1) 0, rat5 1 0, rat5 1 0]}, (vector[rat5 (−1/2) (−1/2), rat5 0 0, rat5 (−1/2) (1/2)], vector[rat5 (−1/2) (−1/2), rat5 0 0, rat5 (1/2) (−1/2)], vector[rat5 (−1) 0, rat5 1 0, rat5 1 0])),

({vector[rat5 (−1/2) (−1/2), rat5 0 0, rat5 (−1/2) (1/2)], vector[rat5 (−1) 0, rat5 (−1) 0, rat5 1 0], vector[rat5 (−1) 0, rat5 1 0, rat5 1 0], vector[rat5 0 0, rat5 (−1/2) (1/2), rat5 (1/2) (1/2)], vector[rat5 0 0, rat5 (1/2) (−1/2), rat5 (1/2) (1/2)]}, (vector[rat5 (−1/2) (−1/2), rat5 0 0, rat5 (−1/2) (1/2)], vector[rat5 (−1) 0, rat5 (−1) 0, rat5 1 0], vector[rat5 (−1) 0, rat5 1 0, rat5 1 0])), ({vector[rat5 (−1/2) (−1/2), rat5 0 0, rat5 (1/2) (−1/2)], vector[rat5 (−1) 0, rat5 (−1) 0, rat5 (−1) 0], vector[rat5 (−1) 0, rat5 1 0, rat5 (−1) 0], vector[rat5 0 0, rat5 (−1/2) (1/2), rat5 (−1/2) (−1/2)], vector[rat5 0 0, rat5 (1/2) (−1/2), rat5 (−1/2) (−1/2)]}, (vector[rat5 (−1/2) (−1/2), rat5 0 0, rat5 (1/2) (−1/2)], vector[rat5 (−1) 0, rat5 (−1) 0, rat5 (−1) 0], vector[rat5 (−1) 0, rat5 1 0, rat5 (−1) 0])),

({vector[rat5 (−1/2) (1/2), rat5 (−1/2) (−1/2), rat5 0 0], vector[rat5 (1/2) (−1/2), rat5 (−1/2) (−1/2), rat5 0 0], vector[rat5 (−1) 0, rat5 (−1) 0, rat5 (−1) 0], vector[rat5 1 0, rat5 (−1) 0, rat5 (−1) 0], vector[rat5 0 0, rat5 (1/2) (−1/2), rat5 (−1/2) (−1/2)]}, (vector[rat5 (−1/2) (1/2), rat5 (−1/2) (−1/2), rat5 0 0], vector[rat5 (1/2) (−1/2), rat5 (−1/2) (−1/2), rat5 0 0], vector[rat5 1 0, rat5 (−1) 0, rat5 (−1) 0])),

({vector[rat5 (−1/2) (1/2), rat5 (−1/2) (−1/2), rat5 0 0], vector[rat5 (1/2) (−1/2), rat5 (−1/2) (−1/2), rat5 0 0], vector[rat5 (−1) 0, rat5 (−1) 0, rat5 1 0], vector[rat5 1 0, rat5 (−1) 0, rat5 1 0], vector[rat5 0 0, rat5 (1/2) (−1/2), rat5 (1/2) (1/2)]}, (vector[rat5 (−1/2) (1/2), rat5 (−1/2) (−1/2), rat5 0 0], vector[rat5 (1/2) (−1/2), rat5 (−1/2) (−1/2), rat5 0 0], vector[rat5 1 0, rat5 (−1) 0, rat5 1 0])),

({vector[rat5 (−1/2) (1/2), rat5 (−1/2) (−1/2), rat5 0 0], vector[rat5 (1/2) (1/2), rat5 0 0, rat5 (−1/2) (1/2)], vector[rat5 (1/2) (1/2), rat5 0 0, rat5 (1/2) (−1/2)], vector[rat5 1 0, rat5 (−1) 0, rat5 (−1) 0], vector[rat5 1 0, rat5 (−1) 0, rat5 1 0]}, (vector[rat5 (−1/2) (1/2), rat5 (−1/2) (−1/2), rat5 0 0], vector[rat5 1 0, rat5 (−1) 0, rat5 (−1) 0], vector[rat5 1 0, rat5 (−1) 0, rat5 1 0])),

({vector[rat5 (−1/2) (1/2), rat5 (1/2) (1/2), rat5 0 0], vector[rat5 (1/2) (−1/2), rat5 (1/2) (1/2), rat5 0 0], vector[rat5 (−1) 0, rat5 1 0, rat5 (−1) 0], vector[rat5 1 0, rat5 1 0, rat5 (−1) 0], vector[rat5 0 0, rat5 (−1/2) (1/2), rat5 (−1/2) (−1/2)]}, (vector[rat5 (−1/2) (1/2), rat5 (1/2) (1/2), rat5 0 0], vector[rat5 (1/2) (−1/2), rat5 (1/2) (1/2), rat5 0 0], vector[rat5 1 0, rat5 1 0, rat5 (−1) 0])),

({vector[rat5 (−1/2) (1/2), rat5 (1/2) (1/2), rat5 0 0], vector[rat5 (1/2) (−1/2), rat5 (1/2) (1/2), rat5 0 0], vector[rat5 (−1) 0, rat5 1 0, rat5 1 0], vector[rat5 1 0, rat5 1 0, rat5 1 0], vector[rat5 0 0, rat5 (−1/2) (1/2), rat5 (1/2) (1/2)]}, (vector[rat5 (−1/2) (1/2), rat5 (1/2) (1/2), rat5 0 0], vector[rat5 (1/2) (−1/2), rat5 (1/2) (1/2), rat5 0 0], vector[rat5 1 0, rat5 1 0, rat5 1 0])),

({vector[rat5 (−1/2) (1/2), rat5 (1/2) (1/2), rat5 0 0], vector[rat5 (1/2) (1/2), rat5 0 0, rat5 (−1/2) (1/2)], vector[rat5 (1/2) (1/2), rat5 0 0, rat5 (1/2) (−1/2)], vector[rat5 1 0, rat5 1 0, rat5 (−1) 0], vector[rat5 1 0, rat5 1 0, rat5 1 0]}, (vector[rat5 (−1/2) (1/2), rat5 (1/2) (1/2), rat5 0 0], vector[rat5 1 0, rat5 1 0, rat5 (−1) 0], vector[rat5 1 0, rat5 1 0, rat5 1 0])),

({vector[rat5 (1/2) (1/2), rat5 0 0, rat5 (−1/2) (1/2)], vector[rat5 1 0, rat5 (−1) 0, rat5 1 0], vector[rat5 1 0, rat5 1 0, rat5 1 0], vector[rat5 0 0, rat5 (−1/2) (1/2), rat5 (1/2) (1/2)], vector[rat5 0 0, rat5 (1/2) (−1/2), rat5 (1/2) (1/2)]}, (vector[rat5 (1/2) (1/2), rat5 0 0, rat5 (−1/2) (1/2)], vector[rat5 1 0, rat5 (−1) 0, rat5 1 0], vector[rat5 1 0, rat5 1 0, rat5 1 0])),

({vector[rat5 (1/2) (1/2), rat5 0 0, rat5 (1/2) (−1/2)], vector[rat5 1 0, rat5 (−1) 0, rat5 (−1) 0], vector[rat5 1 0, rat5 1 0, rat5 (−1) 0], vector[rat5 0 0, rat5 (−1/2) (1/2), rat5 (−1/2) (−1/2)], vector[rat5 0 0, rat5 (1/2) (−1/2), rat5 (−1/2) (−1/2)]}, (vector[rat5 (1/2) (1/2), rat5 0 0, rat5 (1/2) (−1/2)], vector[rat5 1 0, rat5 (−1) 0, rat5 (−1) 0], vector[rat5 1 0, rat5 1 0, rat5 (−1)

0])),

({*vector*[*rat5* (−*1*/*2*) (−*1*/*2*), *rat5 0 0*, *rat5* (−*1*) *0*], *vector*[*rat5* (−*1*/*2*) (−*1*/*2*), *rat5 0 0*, *rat5 1 0*], *vector*[*rat5* (−*1*) *0*, *rat5* (−*1*/*2*) (−*1*/*2*), *rat5 0 0*]}, (*vector*[*rat5* (−*1*/*2*) (−*1*/*2*), *rat5 0 0*, *rat5* (−*1*) *0*], *vector*[*rat5* (−*1*/*2*) (−*1*/*2*), *rat5 0 0*, *rat5 1 0*], *vector*[*rat5* (−*1*) *0*, *rat5* (−*1*/*2*) (−*1*/*2*), *rat5 0 0*])),

({*vector*[*rat5* (−*1*/*2*) (−*1*/*2*), *rat5 0 0*, *rat5* (−*1*) *0*], *vector*[*rat5* (−*1*/*2*) (−*1*/*2*), *rat5 0 0*, *rat5 1 0*], *vector*[*rat5* (−*1*) *0*, *rat5* (*1*/*2*) (*1*/*2*), *rat5 0 0*]}, (*vector*[*rat5* (−*1*/*2*) (−*1*/*2*), *rat5 0 0*, *rat5* (−*1*) *0*], *vector*[*rat5* (−*1*/*2*) (−*1*/*2*), *rat5 0 0*, *rat5 1 0*], *vector*[*rat5* (−*1*) *0*, *rat5* (*1*/*2*) (*1*/*2*), *rat5 0 0*])),

({*vector*[*rat5* (−*1*/*2*) (−*1*/*2*), *rat5 0 0*, *rat5* (−*1*) *0*], *vector*[*rat5* (−*1*) *0*, *rat5* (−*1*/*2*) (−*1*/*2*), *rat5 0 0*], *vector*[*rat5 0 0*, *rat5* (−*1*) *0*, *rat5* (−*1*/*2*) (−*1*/*2*)]}, (*vector*[*rat5* (−*1*/*2*) (−*1*/*2*), *rat5 0 0*, *rat5* (−*1*) *0*], *vector*[*rat5* (−*1*) *0*, *rat5* (−*1*/*2*) (−*1*/*2*), *rat5 0 0*], *vector*[*rat5 0 0*, *rat5* (−*1*) *0*, *rat5* (−*1*/*2*) (−*1*/*2*)])),

({*vector*[*rat5* (−*1*/*2*) (−*1*/*2*), *rat5 0 0*, *rat5* (−*1*) *0*], *vector*[*rat5* (−*1*) *0*, *rat5* (*1*/*2*) (*1*/*2*), *rat5 0 0*], *vector*[*rat5 0 0*, *rat5 1 0*, *rat5* (−*1*/*2*) (−*1*/*2*)]}, (*vector*[*rat5* (−*1*/*2*) (−*1*/*2*), *rat5 0 0*, *rat5* (−*1*) *0*], *vector*[*rat5* (−*1*) *0*, *rat5* (*1*/*2*) (*1*/*2*), *rat5 0 0*], *vector*[*rat5 0 0*, *rat5 1 0*, *rat5* (−*1*/*2*) (−*1*/*2*)])),

({*vector*[*rat5* (−*1*/*2*) (−*1*/*2*), *rat5 0 0*, *rat5* (−*1*) *0*], *vector*[*rat5 0 0*, *rat5* (−*1*) *0*, *rat5* (−*1*/*2*) (−*1*/*2*)], *vector*[*rat5 0 0*, *rat5 1 0*, *rat5* (−*1*/*2*) (−*1*/*2*)]}, (*vector*[*rat5* (−*1*/*2*) (−*1*/*2*), *rat5 0 0*, *rat5* (−*1*) *0*], *vector*[*rat5 0 0*, *rat5* (−*1*) *0*, *rat5* (−*1*/*2*) (−*1*/*2*)], *vector*[*rat5 0 0*, *rat5 1 0*, *rat5* (−*1*/*2*) (−*1*/*2*)])),

({*vector*[*rat5* (−*1*/*2*) (−*1*/*2*), *rat5 0 0*, *rat5 1 0*], *vector*[*rat5* (−*1*) *0*, *rat5* (−*1*/*2*) (−*1*/*2*), *rat5 0 0*], *vector*[*rat5 0 0*, *rat5* (−*1*) *0*, *rat5* (*1*/*2*) (*1*/*2*)]}, (*vector*[*rat5* (−*1*/*2*) (−*1*/*2*), *rat5 0 0*, *rat5 1 0*], *vector*[*rat5* (−*1*) *0*, *rat5* (−*1*/*2*) (−*1*/*2*), *rat5 0 0*], *vector*[*rat5 0 0*, *rat5* (−*1*) *0*, *rat5* (*1*/*2*) (*1*/*2*)])),

({*vector*[*rat5* (−*1*/*2*) (−*1*/*2*), *rat5 0 0*, *rat5 1 0*], *vector*[*rat5* (−*1*) *0*, *rat5* (*1*/*2*) (*1*/*2*), *rat5 0 0*], *vector*[*rat5 0 0*, *rat5 1 0*, *rat5* (*1*/*2*) (*1*/*2*)]}, (*vector*[*rat5* (−*1*/*2*) (−*1*/*2*), *rat5 0 0*, *rat5 1 0*], *vector*[*rat5* (−*1*) *0*, *rat5* (*1*/*2*) (*1*/*2*), *rat5 0 0*], *vector*[*rat5 0 0*, *rat5 1 0*, *rat5* (*1*/*2*) (*1*/*2*)])),

({*vector*[*rat5* (−*1*/*2*) (−*1*/*2*), *rat5 0 0*, *rat5 1 0*], *vector*[*rat5 0 0*, *rat5* (−*1*) *0*, *rat5* (*1*/*2*) (*1*/*2*)], *vector*[*rat5 0 0*, *rat5 1 0*, *rat5* (*1*/*2*) (*1*/*2*)]}, (*vector*[*rat5* (−*1*/*2*) (−*1*/*2*), *rat5 0 0*, *rat5 1 0*], *vector*[*rat5 0 0*, *rat5* (−*1*) *0*, *rat5* (*1*/*2*) (*1*/*2*)], *vector*[*rat5 0 0*, *rat5 1 0*, *rat5* (*1*/*2*) (*1*/*2*)])),

({*vector*[*rat5* (*1*/*2*) (*1*/*2*), *rat5 0 0*, *rat5* (−*1*) *0*], *vector*[*rat5* (*1*/*2*) (*1*/*2*), *rat5 0 0*, *rat5 1 0*], *vector*[*rat5 1 0*, *rat5* (−*1*/*2*) (−*1*/*2*), *rat5 0 0*]}, (*vector*[*rat5* (*1*/*2*) (*1*/*2*), *rat5 0 0*, *rat5* (−*1*) *0*], *vector*[*rat5* (*1*/*2*) (*1*/*2*), *rat5 0 0*, *rat5 1 0*], *vector*[*rat5 1 0*, *rat5* (−*1*/*2*) (−*1*/*2*), *rat5 0 0*])),

({*vector*[*rat5* (*1*/*2*) (*1*/*2*), *rat5 0 0*, *rat5* (−*1*) *0*], *vector*[*rat5* (*1*/*2*) (*1*/*2*), *rat5 0 0*, *rat5 1 0*], *vector*[*rat5 1 0*, *rat5* (*1*/*2*) (*1*/*2*), *rat5 0 0*]}, (*vector*[*rat5* (*1*/*2*) (*1*/*2*), *rat5 0 0*, *rat5* (−*1*) *0*], *vector*[*rat5* (*1*/*2*) (*1*/*2*), *rat5 0 0*, *rat5 1 0*], *vector*[*rat5 1 0*, *rat5* (*1*/*2*) (*1*/*2*), *rat5 0 0*])),

({*vector*[*rat5* (*1*/*2*) (*1*/*2*), *rat5 0 0*, *rat5* (−*1*) *0*], *vector*[*rat5 1 0*, *rat5* (−*1*/*2*) (−*1*/*2*), *rat5 0 0*], *vector*[*rat5 0 0*, *rat5* (−*1*) *0*, *rat5* (−*1*/*2*) (−*1*/*2*)]}, (*vector*[*rat5* (*1*/*2*) (*1*/*2*), *rat5 0 0*, *rat5* (−*1*) *0*], *vector*[*rat5 1 0*, *rat5* (−*1*/*2*) (−*1*/*2*), *rat5 0 0*], *vector*[*rat5 0 0*, *rat5* (−*1*) *0*, *rat5* (−*1*/*2*) (−*1*/*2*)])),

({*vector*[*rat5* (*1*/*2*) (*1*/*2*), *rat5 0 0*, *rat5* (−*1*) *0*], *vector*[*rat5 1 0*, *rat5* (*1*/*2*) (*1*/*2*), *rat5 0 0*], *vector*[*rat5 0 0*, *rat5 1 0*, *rat5* (−*1*/*2*) (−*1*/*2*)]}, (*vector*[*rat5* (*1*/*2*) (*1*/*2*), *rat5 0 0*, *rat5* (−*1*) *0*], *vector*[*rat5 1 0*, *rat5* (*1*/*2*) (*1*/*2*), *rat5 0 0*],

vector[*rat5 0 0*, *rat5 1 0*, *rat5* (−*1*/*2*) (−*1*/*2*)])),
({*vector*[*rat5* (*1*/*2*) (*1*/*2*), *rat5 0 0*, *rat5* (−*1*) *0*], *vector*[*rat5 0 0*, *rat5* (−*1*) *0*, *rat5* (−*1*/*2*) (−*1*/*2*)], *vector*[*rat5 0 0*, *rat5 1 0*, *rat5* (−*1*/*2*) (−*1*/*2*)]}, (*vector*[*rat5* (*1*/*2*) (*1*/*2*), *rat5 0 0*, *rat5* (−*1*) *0*], *vector*[*rat5 0 0*, *rat5* (−*1*) *0*, *rat5* (−*1*/*2*) (−*1*/*2*)], *vector*[*rat5 0 0*, *rat5 1 0*, *rat5* (−*1*/*2*) (−*1*/*2*)])),
({*vector*[*rat5* (*1*/*2*) (*1*/*2*), *rat5 0 0*, *rat5 1 0*], *vector*[*rat5 1 0*, *rat5* (−*1*/*2*) (−*1*/*2*), *rat5 0 0*], *vector*[*rat5 0 0*, *rat5* (−*1*) *0*, *rat5* (*1*/*2*) (*1*/*2*)]}, (*vector*[*rat5* (*1*/*2*) (*1*/*2*), *rat5 0 0*, *rat5 1 0*], *vector*[*rat5 1 0*, *rat5* (−*1*/*2*) (−*1*/*2*), *rat5 0 0*], *vector*[*rat5 0 0*, *rat5* (−*1*) *0*, *rat5* (*1*/*2*) (*1*/*2*)])),
({*vector*[*rat5* (*1*/*2*) (*1*/*2*), *rat5 0 0*, *rat5 1 0*], *vector*[*rat5 1 0*, *rat5* (*1*/*2*) (*1*/*2*), *rat5 0 0*], *vector*[*rat5 0 0*, *rat5 1 0*, *rat5* (*1*/*2*) (*1*/*2*)]}, (*vector*[*rat5* (*1*/*2*) (*1*/*2*), *rat5 0 0*, *rat5 1 0*], *vector*[*rat5 1 0*, *rat5* (*1*/*2*) (*1*/*2*), *rat5 0 0*], *vector*[*rat5 0 0*, *rat5 1 0*, *rat5* (*1*/*2*) (*1*/*2*)])),
({*vector*[*rat5* (*1*/*2*) (*1*/*2*), *rat5 0 0*, *rat5 1 0*], *vector*[*rat5 0 0*, *rat5* (−*1*) *0*, *rat5* (*1*/*2*) (*1*/*2*)], *vector*[*rat5 0 0*, *rat5 1 0*, *rat5* (*1*/*2*) (*1*/*2*)]}, (*vector*[*rat5* (*1*/*2*) (*1*/*2*), *rat5 0 0*, *rat5 1 0*], *vector*[*rat5 0 0*, *rat5* (−*1*) *0*, *rat5* (*1*/*2*) (*1*/*2*)], *vector*[*rat5 0 0*, *rat5 1 0*, *rat5* (*1*/*2*) (*1*/*2*)])),
({*vector*[*rat5* (−*1*) *0*, *rat5* (−*1*/*2*) (−*1*/*2*), *rat5 0 0*], *vector*[*rat5 1 0*, *rat5* (−*1*/*2*) (−*1*/*2*), *rat5 0 0*], *vector*[*rat5 0 0*, *rat5* (−*1*) *0*, *rat5* (−*1*/*2*) (−*1*/*2*)]}, (*vector*[*rat5* (−*1*) *0*, *rat5* (−*1*/*2*) (−*1*/*2*), *rat5 0 0*], *vector*[*rat5 1 0*, *rat5* (−*1*/*2*) (−*1*/*2*), *rat5 0 0*], *vector*[*rat5 0 0*, *rat5* (−*1*) *0*, *rat5* (−*1*/*2*) (−*1*/*2*)])),
({*vector*[*rat5* (−*1*) *0*, *rat5* (−*1*/*2*) (−*1*/*2*), *rat5 0 0*], *vector*[*rat5 1 0*, *rat5* (−*1*/*2*) (−*1*/*2*), *rat5 0 0*], *vector*[*rat5 0 0*, *rat5* (−*1*) *0*, *rat5* (*1*/*2*) (*1*/*2*)]}, (*vector*[*rat5* (−*1*) *0*, *rat5* (−*1*/*2*) (−*1*/*2*), *rat5 0 0*], *vector*[*rat5 1 0*, *rat5* (−*1*/*2*) (−*1*/*2*), *rat5 0 0*], *vector*[*rat5 0 0*, *rat5* (−*1*) *0*, *rat5* (*1*/*2*) (*1*/*2*)])),
({*vector*[*rat5* (−*1*) *0*, *rat5* (*1*/*2*) (*1*/*2*), *rat5 0 0*], *vector*[*rat5 1 0*, *rat5* (*1*/*2*) (*1*/*2*), *rat5 0 0*], *vector*[*rat5 0 0*, *rat5 1 0*, *rat5* (−*1*/*2*) (−*1*/*2*)]}, (*vector*[*rat5* (−*1*) *0*, *rat5* (*1*/*2*) (*1*/*2*), *rat5 0 0*], *vector*[*rat5 1 0*, *rat5* (*1*/*2*) (*1*/*2*), *rat5 0 0*], *vector*[*rat5 0 0*, *rat5 1 0*, *rat5* (−*1*/*2*) (−*1*/*2*)])),
({*vector*[*rat5* (−*1*) *0*, *rat5* (*1*/*2*) (*1*/*2*), *rat5 0 0*], *vector*[*rat5 1 0*, *rat5* (*1*/*2*) (*1*/*2*), *rat5 0 0*], *vector*[*rat5 0 0*, *rat5 1 0*, *rat5* (*1*/*2*) (*1*/*2*)]}, (*vector*[*rat5* (−*1*) *0*, *rat5* (*1*/*2*) (*1*/*2*), *rat5 0 0*], *vector*[*rat5 1 0*, *rat5* (*1*/*2*) (*1*/*2*), *rat5 0 0*], *vector*[*rat5 0 0*, *rat5 1 0*, *rat5* (*1*/*2*) (*1*/*2*)]))
]

definition *adjacent-points* :: (*real*, *3*) *vec set* ⇒ (*real*, *3*) *vec* × (*real*, *3*) *vec* × (*real*, *3*) *vec* **where**
adjacent-points S = (
case map-of adjacency-table S of
Some res ⇒ *res*
| *None* ⇒ *undefined*
)

lemma *set43-neqI*:
assumes *a* ∉ {*x*, *y*, *z*} ∨ *b* ∉ {*x*, *y*, *z*} ∨ *c* ∉ {*x*, *y*, *z*} ∨ *d* ∉ {*x*, *y*, *z*}
shows {*a*, *b*, *c*, *d*} ≠ {*x*, *y*, *z*}
{*x*, *y*, *z*} ≠ {*a*, *b*, *c*, *d*}
using *assms* **by** *auto*

lemma *solids-vectors-to-rat5*:
shows $vector[1, 1, 1] = vector[rat5\ (1)\ 0, rat5\ (1)\ 0, rat5\ (1)\ 0]$
$vector[1, 1, -1] = vector[rat5\ (1)\ 0, rat5\ (1)\ 0, rat5\ (-1)\ 0]$
$vector[1, -1, 1] = vector[rat5\ (1)\ 0, rat5\ (-1)\ 0, rat5\ (1)\ 0]$
$vector[1, -1, -1] = vector[rat5\ (1)\ 0, rat5\ (-1)\ 0, rat5\ (-1)\ 0]$
$vector[-1, 1, 1] = vector[rat5\ (-1)\ 0, rat5\ (1)\ 0, rat5\ (1)\ 0]$
$vector[-1, 1, -1] = vector[rat5\ (-1)\ 0, rat5\ (1)\ 0, rat5\ (-1)\ 0]$
$vector[-1, -1, 1] = vector[rat5\ (-1)\ 0, rat5\ (-1)\ 0, rat5\ (1)\ 0]$
$vector[-1, -1, -1] = vector[rat5\ (-1)\ 0, rat5\ (-1)\ 0, rat5\ (-1)\ 0]$
$vector[1, 0, 0] = vector[rat5\ (1)\ 0, rat5\ (0)\ 0, rat5\ (0)\ 0]$
$vector[-1, 0, 0] = vector[rat5\ (-1)\ 0, rat5\ (0)\ 0, rat5\ (0)\ 0]$
$vector[0, 0, 1] = vector[rat5\ (0)\ 0, rat5\ (0)\ 0, rat5\ (1)\ 0]$
$vector[0, 0, -1] = vector[rat5\ (0)\ 0, rat5\ (0)\ 0, rat5\ (-1)\ 0]$
$vector[0, 1, 0] = vector[rat5\ (0)\ 0, rat5\ (1)\ 0, rat5\ (0)\ 0]$
$vector[0, -1, 0] = vector[rat5\ (0)\ 0, rat5\ (-1)\ 0, rat5\ (0)\ 0]$
unfolding *rat5-def*
by *auto*

lemma *set53-neqI*:
assumes $a \notin \{x, y, z\} \vee b \notin \{x, y, z\} \vee c \notin \{x, y, z\} \vee d \notin \{x, y, z\} \vee e \notin \{x, y, z\}$
shows $\{a, b, c, d,e\} \neq \{x, y, z\}$
$\{x, y, z\} \neq \{a, b, c, d, e\}$
using *assms* **by** *auto*

lemma *set54-neqI*:
assumes $a \notin \{w, x, y, z\} \vee b \notin \{w, x, y, z\} \vee c \notin \{w, x, y, z\} \vee d \notin \{w, x, y, z\} \vee e \notin \{w, x, y, z\}$
shows $\{a, b, c, d,e\} \neq \{w, x, y, z\}$
$\{w, x, y, z\} \neq \{a, b, c, d, e\}$
using *assms* **by** *auto*

lemma *set5-neqI*:
assumes $a \notin \{v, w, x, y, z\} \vee b \notin \{v, w, x, y, z\} \vee c \notin \{v, w, x, y, z\} \vee d \notin \{v, w, x, y, z\} \vee e \notin \{v, w, x, y, z\}$
shows $\{a, b, c, d,e\} \neq \{v, w, x, y, z\}$
using *assms* **by** *auto*

lemma *set3-eq-iff*:
shows $(\{a, b, c\} = \{d, e, f\}) =$
$(a \in \{d, e, f\} \wedge b \in \{d, e, f\} \wedge c \in \{d, e, f\} \wedge$
$d \in \{a, b, c\} \wedge e \in \{a, b, c\} \wedge f \in \{a, b, c\})$
by *auto*

lemma *set3-vector-neq*:
assumes $(a1 \neq d1 \wedge a1 \neq d2 \wedge a1 \neq d3)$
shows $\{(vector\ [a1, b1, c1]::real\hat{\ }3), vector\ [a2, b2, c2], vector\ [a3, b3, c3]\} \neq$
$\{vector\ [d1, e1, f1], vector\ [d2, e2, f2], vector\ [d3, e3, f3]\}$
by (*auto simp add*: *set3-neqI assms vector3-neq*)

lemma *adjacent-points*:
shows *adjacent-points* {*vector*[−1, −1, 1], *vector*[− 1, 1, − 1], *vector*[1, − 1, − 1]} = (*vector* [−1, −1, 1], *vector* [−1, 1, −1], *vector* [1, −1, −1])
adjacent-points {*vector*[−1, −1, 1], *vector*[−1, 1, −1], *vector*[1, 1, 1]} = (*vector*[−1, −1, 1], *vector*[−1, 1, − 1], *vector*[1, 1, 1])
adjacent-points {*vector*[−1, −1, 1], *vector*[1, − 1, − 1], *vector*[1, 1, 1]} = (*vector*[−1, −1, 1], *vector*[1, − 1, − 1], *vector*[1, 1, 1])
adjacent-points {*vector*[−1, 1, −1], *vector*[1, − 1, − 1], *vector*[1, 1, 1]} = (*vector*[−1, 1, −1], *vector*[1, − 1, − 1], *vector*[1, 1, 1])

adjacent-points {*vector*[−1, −1, −1], *vector*[−1, −1, 1], *vector*[−1, 1, −1], *vector*[−1, 1, 1]} = (*vector*[−1, −1, −1], *vector*[−1, −1, 1], *vector*[−1, 1, 1])
adjacent-points {*vector*[−1, −1, −1], *vector*[−1, −1, 1], *vector*[1, −1, −1], *vector*[1, −1, 1]} = (*vector*[−1, −1, −1], *vector*[−1, −1, 1], *vector*[1, −1, 1])
adjacent-points {*vector*[−1, −1, −1], *vector*[−1, 1, −1], *vector*[1, −1, −1], *vector*[1, 1, −1]} = (*vector*[−1, −1, −1], *vector*[−1, 1, −1], *vector*[1, 1, −1])
adjacent-points {*vector*[−1, −1, 1], *vector*[−1, 1, 1], *vector*[1, −1, 1], *vector*[1, 1, 1]} = (*vector*[−1, −1, 1], *vector*[−1, 1, 1], *vector*[1, 1, 1])
adjacent-points {*vector*[−1, 1, −1], *vector*[−1, 1, 1], *vector*[1, 1, −1], *vector*[1, 1, 1]} = (*vector*[−1, 1, −1], *vector*[−1, 1, 1], *vector*[1, 1, 1])
adjacent-points {*vector*[1, −1, −1], *vector*[1, −1, 1], *vector*[1, 1, −1], *vector*[1, 1, 1]} = (*vector*[1, −1, −1], *vector*[1, −1, 1], *vector*[1, 1, 1])

adjacent-points {*vector*[−1, 0, 0], *vector*[0, −1, 0], *vector*[0, 0, −1]} = (*vector*[−1, 0, 0], *vector*[0, −1, 0], *vector*[0, 0, −1])
adjacent-points {*vector*[−1, 0, 0], *vector*[0, −1, 0], *vector*[0, 0, 1]} = (*vector*[−1, 0, 0], *vector*[0, −1, 0], *vector*[0, 0, 1])
adjacent-points {*vector*[−1, 0, 0], *vector*[0, 1, 0], *vector*[0, 0, −1]} = (*vector*[−1, 0, 0], *vector*[0, 1, 0], *vector*[0, 0, −1])
adjacent-points {*vector*[−1, 0, 0], *vector*[0, 1, 0], *vector*[0, 0, 1]} = (*vector*[−1, 0, 0], *vector*[0, 1, 0], *vector*[0, 0, 1])
adjacent-points {*vector*[1, 0, 0], *vector*[0, −1, 0], *vector*[0, 0, −1]} = (*vector*[1, 0, 0], *vector*[0, −1, 0], *vector*[0, 0, −1])
adjacent-points {*vector*[1, 0, 0], *vector*[0, −1, 0], *vector*[0, 0, 1]} = (*vector*[1, 0, 0], *vector*[0, −1, 0], *vector*[0, 0, 1])
adjacent-points {*vector*[1, 0, 0], *vector*[0, 1, 0], *vector*[0, 0, −1]} = (*vector*[1, 0, 0], *vector*[0, 1, 0], *vector*[0, 0, −1])
adjacent-points {*vector*[1, 0, 0], *vector*[0, 1, 0], *vector*[0, 0, 1]} = (*vector*[1, 0, 0], *vector*[0, 1, 0], *vector*[0, 0, 1])

adjacent-points {*vector*[*rat5* (−1/2) (−1/2), *rat5* 0 0, *rat5* (−1/2) (1/2)], *vector*[*rat5* (−1/2) (−1/2), *rat5* 0 0, *rat5* (1/2) (−1/2)], *vector*[*rat5* (1/2) (−1/2), *rat5* (−1/2) (−1/2), *rat5* 0 0], *vector*[*rat5* (−1) 0, *rat5* (−1) 0, *rat5* (−1) 0], *vector*[*rat5* (−1) 0, *rat5* (−1) 0, *rat5* 1 0]} = (*vector*[*rat5* (−1/2) (−1/2), *rat5* 0 0, *rat5* (−1/2) (1/2)], *vector*[*rat5* (−1/2) (−1/2), *rat5* 0 0, *rat5* (1/2) (−1/2)], *vector*[*rat5* (−1) 0, *rat5* (−1) 0, *rat5* 1 0])
adjacent-points {*vector*[*rat5* (−1/2) (−1/2), *rat5* 0 0, *rat5* (−1/2) (1/2)], *vector*[*rat5* (−1/2) (−1/2), *rat5* 0 0, *rat5* (1/2) (−1/2)], *vector*[*rat5* (1/2)

(−1/2), rat5 (1/2) (1/2), rat5 0 0], vector[rat5 (−1) 0, rat5 1 0, rat5 (−1) 0], vector[rat5 (−1) 0, rat5 1 0, rat5 1 0]} = (vector[rat5 (−1/2) (−1/2), rat5 0 0, rat5 (−1/2) (1/2)], vector[rat5 (−1/2) (−1/2), rat5 0 0, rat5 (1/2) (−1/2)], vector[rat5 (−1) 0, rat5 1 0, rat5 1 0])

adjacent-points {vector[rat5 (−1/2) (−1/2), rat5 0 0, rat5 (−1/2) (1/2)], vector[rat5 (−1) 0, rat5 (−1) 0, rat5 1 0], vector[rat5 (−1) 0, rat5 1 0, rat5 1 0], vector[rat5 0 0, rat5 (−1/2) (1/2), rat5 (1/2) (1/2)], vector[rat5 0 0, rat5 (1/2) (−1/2), rat5 (1/2) (1/2)]} = (vector[rat5 (−1/2) (−1/2), rat5 0 0, rat5 (−1/2) (1/2)], vector[rat5 (−1) 0, rat5 (−1) 0, rat5 1 0], vector[rat5 (−1) 0, rat5 1 0, rat5 1 0])

adjacent-points {vector[rat5 (−1/2) (−1/2), rat5 0 0, rat5 (1/2) (−1/2)], vector[rat5 (−1) 0, rat5 (−1) 0, rat5 (−1) 0], vector[rat5 (−1) 0, rat5 1 0, rat5 (−1) 0], vector[rat5 0 0, rat5 (−1/2) (1/2), rat5 (−1/2) (−1/2)], vector[rat5 0 0, rat5 (1/2) (−1/2), rat5 (−1/2) (−1/2)]} = (vector[rat5 (−1/2) (−1/2), rat5 0 0, rat5 (1/2) (−1/2)], vector[rat5 (−1) 0, rat5 (−1) 0, rat5 (−1) 0], vector[rat5 (−1) 0, rat5 1 0, rat5 (−1) 0])

adjacent-points {vector[rat5 (−1/2) (1/2), rat5 (−1/2) (−1/2), rat5 0 0], vector[rat5 (1/2) (−1/2), rat5 (−1/2) (−1/2), rat5 0 0], vector[rat5 (−1) 0, rat5 (−1) 0, rat5 (−1) 0], vector[rat5 1 0, rat5 (−1) 0, rat5 (−1) 0], vector[rat5 0 0, rat5 (1/2) (−1/2), rat5 (−1/2) (−1/2)]} = (vector[rat5 (−1/2) (1/2), rat5 (−1/2) (−1/2), rat5 0 0], vector[rat5 (1/2) (−1/2), rat5 (−1/2) (−1/2), rat5 0 0], vector[rat5 1 0, rat5 (−1) 0, rat5 (−1) 0])

adjacent-points {vector[rat5 (−1/2) (1/2), rat5 (−1/2) (−1/2), rat5 0 0], vector[rat5 (1/2) (−1/2), rat5 (−1/2) (−1/2), rat5 0 0], vector[rat5 (−1) 0, rat5 (−1) 0, rat5 1 0], vector[rat5 1 0, rat5 (−1) 0, rat5 1 0], vector[rat5 0 0, rat5 (1/2) (−1/2), rat5 (1/2) (1/2)]} = (vector[rat5 (−1/2) (1/2), rat5 (−1/2) (−1/2), rat5 0 0], vector[rat5 (1/2) (−1/2), rat5 (−1/2) (−1/2), rat5 0 0], vector[rat5 1 0, rat5 (−1) 0, rat5 1 0])

adjacent-points {vector[rat5 (−1/2) (1/2), rat5 (−1/2) (−1/2), rat5 0 0], vector[rat5 (1/2) (1/2), rat5 0 0, rat5 (−1/2) (1/2)], vector[rat5 (1/2) (1/2), rat5 0 0, rat5 (1/2) (−1/2)], vector[rat5 1 0, rat5 (−1) 0, rat5 (−1) 0], vector[rat5 1 0, rat5 (−1) 0, rat5 1 0]} = (vector[rat5 (−1/2) (1/2), rat5 (−1/2) (−1/2), rat5 0 0], vector[rat5 1 0, rat5 (−1) 0, rat5 (−1) 0], vector[rat5 1 0, rat5 (−1) 0, rat5 1 0])

adjacent-points {vector[rat5 (−1/2) (1/2), rat5 (1/2) (1/2), rat5 0 0], vector[rat5 (1/2) (−1/2), rat5 (1/2) (1/2), rat5 0 0], vector[rat5 (−1) 0, rat5 1 0, rat5 (−1) 0], vector[rat5 1 0, rat5 1 0, rat5 (−1) 0], vector[rat5 0 0, rat5 (−1/2) (1/2), rat5 (−1/2) (−1/2)]} = (vector[rat5 (−1/2) (1/2), rat5 (1/2) (1/2), rat5 0 0], vector[rat5 (1/2) (−1/2), rat5 (1/2) (1/2), rat5 0 0], vector[rat5 1 0, rat5 1 0, rat5 (−1) 0])

adjacent-points {vector[rat5 (−1/2) (1/2), rat5 (1/2) (1/2), rat5 0 0], vector[rat5 (1/2) (−1/2), rat5 (1/2) (1/2), rat5 0 0], vector[rat5 (−1) 0, rat5 1 0, rat5 1 0], vector[rat5 1 0, rat5 1 0, rat5 1 0], vector[rat5 0 0, rat5 (−1/2) (1/2), rat5 (1/2) (1/2)]} = (vector[rat5 (−1/2) (1/2), rat5 (1/2) (1/2), rat5 0 0], vector[rat5 (1/2) (−1/2), rat5 (1/2) (1/2), rat5 0 0], vector[rat5 1 0, rat5 1 0, rat5 1 0])

adjacent-points {vector[rat5 (−1/2) (1/2), rat5 (1/2) (1/2), rat5 0 0], vector[rat5 (1/2) (1/2), rat5 0 0, rat5 (−1/2) (1/2)], vector[rat5 (1/2) (1/2), rat5 0 0, rat5 (1/2) (−1/2)], vector[rat5 1 0, rat5 1 0, rat5 (−1) 0], vector[rat5

1 0, rat5 1 0, rat5 1 0]} = (vector[rat5 (−1/2) (1/2), rat5 (1/2) (1/2), rat5 0 0], vector[rat5 1 0, rat5 1 0, rat5 (−1) 0], vector[rat5 1 0, rat5 1 0, rat5 1 0])

adjacent-points {vector[rat5 (1/2) (1/2), rat5 0 0, rat5 (−1/2) (1/2)], vector[rat5 1 0, rat5 (−1) 0, rat5 1 0], vector[rat5 1 0, rat5 1 0, rat5 1 0], vector[rat5 0 0, rat5 (−1/2) (1/2), rat5 (1/2) (1/2)], vector[rat5 0 0, rat5 (1/2) (−1/2), rat5 (1/2) (1/2)]} = (vector[rat5 (1/2) (1/2), rat5 0 0, rat5 (−1/2) (1/2)], vector[rat5 1 0, rat5 (−1) 0, rat5 1 0], vector[rat5 1 0, rat5 1 0, rat5 1 0])

adjacent-points {vector[rat5 (1/2) (1/2), rat5 0 0, rat5 (1/2) (−1/2)], vector[rat5 1 0, rat5 (−1) 0, rat5 (−1) 0], vector[rat5 1 0, rat5 1 0, rat5 (−1) 0], vector[rat5 0 0, rat5 (−1/2) (1/2), rat5 (−1/2) (−1/2)], vector[rat5 0 0, rat5 (1/2) (−1/2), rat5 (−1/2) (−1/2)]} = (vector[rat5 (1/2) (1/2), rat5 0 0, rat5 (1/2) (−1/2)], vector[rat5 1 0, rat5 (−1) 0, rat5 (−1) 0], vector[rat5 1 0, rat5 1 0, rat5 (−1) 0])

adjacent-points {vector[rat5 (−1/2) (−1/2), rat5 0 0, rat5 (−1) 0], vector[rat5 (−1/2) (−1/2), rat5 0 0, rat5 1 0], vector[rat5 (−1) 0, rat5 (−1/2) (−1/2), rat5 0 0]} = (vector[rat5 (−1/2) (−1/2), rat5 0 0, rat5 (−1) 0], vector[rat5 (−1/2) (−1/2), rat5 0 0, rat5 1 0], vector[rat5 (−1) 0, rat5 (−1/2) (−1/2), rat5 0 0])

adjacent-points {vector[rat5 (−1/2) (−1/2), rat5 0 0, rat5 (−1) 0], vector[rat5 (−1/2) (−1/2), rat5 0 0, rat5 1 0], vector[rat5 (−1) 0, rat5 (1/2) (1/2), rat5 0 0]} = (vector[rat5 (−1/2) (−1/2), rat5 0 0, rat5 (−1) 0], vector[rat5 (−1/2) (−1/2), rat5 0 0, rat5 1 0], vector[rat5 (−1) 0, rat5 (1/2) (1/2), rat5 0 0])

adjacent-points {vector[rat5 (−1/2) (−1/2), rat5 0 0, rat5 (−1) 0], vector[rat5 (−1) 0, rat5 (−1/2) (−1/2), rat5 0 0], vector[rat5 0 0, rat5 (−1) 0, rat5 (−1/2) (−1/2)]} = (vector[rat5 (−1/2) (−1/2), rat5 0 0, rat5 (−1) 0], vector[rat5 (−1) 0, rat5 (−1/2) (−1/2), rat5 0 0], vector[rat5 0 0, rat5 (−1) 0, rat5 (−1/2) (−1/2)])

adjacent-points {vector[rat5 (−1/2) (−1/2), rat5 0 0, rat5 (−1) 0], vector[rat5 (−1) 0, rat5 (1/2) (1/2), rat5 0 0], vector[rat5 0 0, rat5 1 0, rat5 (−1/2) (−1/2)]} = (vector[rat5 (−1/2) (−1/2), rat5 0 0, rat5 (−1) 0], vector[rat5 (−1) 0, rat5 (1/2) (1/2), rat5 0 0], vector[rat5 0 0, rat5 1 0, rat5 (−1/2) (−1/2)])

adjacent-points {vector[rat5 (−1/2) (−1/2), rat5 0 0, rat5 (−1) 0], vector[rat5 0 0, rat5 (−1) 0, rat5 (−1/2) (−1/2)], vector[rat5 0 0, rat5 1 0, rat5 (−1/2) (−1/2)]} = (vector[rat5 (−1/2) (−1/2), rat5 0 0, rat5 (−1) 0], vector[rat5 0 0, rat5 (−1) 0, rat5 (−1/2) (−1/2)], vector[rat5 0 0, rat5 1 0, rat5 (−1/2) (−1/2)])

adjacent-points {vector[rat5 (−1/2) (−1/2), rat5 0 0, rat5 1 0], vector[rat5 (−1) 0, rat5 (−1/2) (−1/2), rat5 0 0], vector[rat5 0 0, rat5 (−1) 0, rat5 (1/2) (1/2)]} = (vector[rat5 (−1/2) (−1/2), rat5 0 0, rat5 1 0], vector[rat5 (−1) 0, rat5 (−1/2) (−1/2), rat5 0 0], vector[rat5 0 0, rat5 (−1) 0, rat5 (1/2) (1/2)])

adjacent-points {vector[rat5 (−1/2) (−1/2), rat5 0 0, rat5 1 0], vector[rat5 (−1) 0, rat5 (1/2) (1/2), rat5 0 0], vector[rat5 0 0, rat5 1 0, rat5 (1/2) (1/2)]} = (vector[rat5 (−1/2) (−1/2), rat5 0 0, rat5 1 0], vector[rat5 (−1) 0, rat5 (1/2) (1/2), rat5 0 0], vector[rat5 0 0, rat5 1 0, rat5 (1/2) (1/2)])

adjacent-points {vector[rat5 (−1/2) (−1/2), rat5 0 0, rat5 1 0], vector[rat5 0 0, rat5 (−1) 0, rat5 (1/2) (1/2)], vector[rat5 0 0, rat5 1 0, rat5 (1/2) (1/2)]}

= (vector[rat5 (−1/2) (−1/2), rat5 0 0, rat5 1 0], vector[rat5 0 0, rat5 (−1) 0, rat5 (1/2) (1/2)], vector[rat5 0 0, rat5 1 0, rat5 (1/2) (1/2)])

adjacent-points {vector[rat5 (1/2) (1/2), rat5 0 0, rat5 (−1) 0], vector[rat5 (1/2) (1/2), rat5 0 0, rat5 1 0], vector[rat5 1 0, rat5 (−1/2) (−1/2), rat5 0 0]} = (vector[rat5 (1/2) (1/2), rat5 0 0, rat5 (−1) 0], vector[rat5 (1/2) (1/2), rat5 0 0, rat5 1 0], vector[rat5 1 0, rat5 (−1/2) (−1/2), rat5 0 0])

adjacent-points {vector[rat5 (1/2) (1/2), rat5 0 0, rat5 (−1) 0], vector[rat5 (1/2) (1/2), rat5 0 0, rat5 1 0], vector[rat5 1 0, rat5 (1/2) (1/2), rat5 0 0]} = (vector[rat5 (1/2) (1/2), rat5 0 0, rat5 (−1) 0], vector[rat5 (1/2) (1/2), rat5 0 0, rat5 1 0], vector[rat5 1 0, rat5 (1/2) (1/2), rat5 0 0])

adjacent-points {vector[rat5 (1/2) (1/2), rat5 0 0, rat5 (−1) 0], vector[rat5 1 0, rat5 (−1/2) (−1/2), rat5 0 0], vector[rat5 0 0, rat5 (−1) 0, rat5 (−1/2) (−1/2)]} = (vector[rat5 (1/2) (1/2), rat5 0 0, rat5 (−1) 0], vector[rat5 1 0, rat5 (−1/2) (−1/2), rat5 0 0], vector[rat5 0 0, rat5 (−1) 0, rat5 (−1/2) (−1/2)])

adjacent-points {vector[rat5 (1/2) (1/2), rat5 0 0, rat5 (−1) 0], vector[rat5 1 0, rat5 (1/2) (1/2), rat5 0 0], vector[rat5 0 0, rat5 1 0, rat5 (−1/2) (−1/2)]} = (vector[rat5 (1/2) (1/2), rat5 0 0, rat5 (−1) 0], vector[rat5 1 0, rat5 (1/2) (1/2), rat5 0 0], vector[rat5 0 0, rat5 1 0, rat5 (−1/2) (−1/2)])

adjacent-points {vector[rat5 (1/2) (1/2), rat5 0 0, rat5 (−1) 0], vector[rat5 0 0, rat5 (−1) 0, rat5 (−1/2) (−1/2)], vector[rat5 0 0, rat5 1 0, rat5 (−1/2) (−1/2)]} = (vector[rat5 (1/2) (1/2), rat5 0 0, rat5 (−1) 0], vector[rat5 0 0, rat5 (−1) 0, rat5 (−1/2) (−1/2)], vector[rat5 0 0, rat5 1 0, rat5 (−1/2) (−1/2)])

adjacent-points {vector[rat5 (1/2) (1/2), rat5 0 0, rat5 1 0], vector[rat5 1 0, rat5 (−1/2) (−1/2), rat5 0 0], vector[rat5 0 0, rat5 (−1) 0, rat5 (1/2) (1/2)]} = (vector[rat5 (1/2) (1/2), rat5 0 0, rat5 1 0], vector[rat5 1 0, rat5 (−1/2) (−1/2), rat5 0 0], vector[rat5 0 0, rat5 (−1) 0, rat5 (1/2) (1/2)])

adjacent-points {vector[rat5 (1/2) (1/2), rat5 0 0, rat5 1 0], vector[rat5 1 0, rat5 (1/2) (1/2), rat5 0 0], vector[rat5 0 0, rat5 1 0, rat5 (1/2) (1/2)]} = (vector[rat5 (1/2) (1/2), rat5 0 0, rat5 1 0], vector[rat5 1 0, rat5 (1/2) (1/2), rat5 0 0], vector[rat5 0 0, rat5 1 0, rat5 (1/2) (1/2)])

adjacent-points {vector[rat5 (1/2) (1/2), rat5 0 0, rat5 1 0], vector[rat5 0 0, rat5 (−1) 0, rat5 (1/2) (1/2)], vector[rat5 0 0, rat5 1 0, rat5 (1/2) (1/2)]} = (vector[rat5 (1/2) (1/2), rat5 0 0, rat5 1 0], vector[rat5 0 0, rat5 (−1) 0, rat5 (1/2) (1/2)], vector[rat5 0 0, rat5 1 0, rat5 (1/2) (1/2)])

adjacent-points {vector[rat5 (−1) 0, rat5 (−1/2) (−1/2), rat5 0 0], vector[rat5 1 0, rat5 (−1/2) (−1/2), rat5 0 0], vector[rat5 0 0, rat5 (−1) 0, rat5 (−1/2) (−1/2)]} = (vector[rat5 (−1) 0, rat5 (−1/2) (−1/2), rat5 0 0], vector[rat5 1 0, rat5 (−1/2) (−1/2), rat5 0 0], vector[rat5 0 0, rat5 (−1) 0, rat5 (−1/2) (−1/2)])

adjacent-points {vector[rat5 (−1) 0, rat5 (−1/2) (−1/2), rat5 0 0], vector[rat5 1 0, rat5 (−1/2) (−1/2), rat5 0 0], vector[rat5 0 0, rat5 (−1) 0, rat5 (1/2) (1/2)]} = (vector[rat5 (−1) 0, rat5 (−1/2) (−1/2), rat5 0 0], vector[rat5 1 0, rat5 (−1/2) (−1/2), rat5 0 0], vector[rat5 0 0, rat5 (−1) 0, rat5 (1/2) (1/2)])

adjacent-points {vector[rat5 (−1) 0, rat5 (1/2) (1/2), rat5 0 0], vector[rat5 1 0, rat5 (1/2) (1/2), rat5 0 0], vector[rat5 0 0, rat5 1 0, rat5 (−1/2) (−1/2)]} = (vector[rat5 (−1) 0, rat5 (1/2) (1/2), rat5 0 0], vector[rat5 1 0, rat5 (1/2) (1/2), rat5 0 0], vector[rat5 0 0, rat5 1 0, rat5 (−1/2) (−1/2)])

adjacent-points {vector[rat5 (−1) 0, rat5 (1/2) (1/2), rat5 0 0], vector[rat5

1 0, *rat5* (*1/2*) (*1/2*), *rat5 0 0*], *vector*[*rat5 0 0*, *rat5 1 0*, *rat5* (*1/2*) (*1/2*)]}
= (*vector*[*rat5* (*−1*) *0*, *rat5* (*1/2*) (*1/2*), *rat5 0 0*], *vector*[*rat5 1 0*, *rat5* (*1/2*) (*1/2*), *rat5 0 0*], *vector*[*rat5 0 0*, *rat5 1 0*, *rat5* (*1/2*) (*1/2*)])
apply (*simp-all add*: *adjacent-points-def adjacency-table-def set3-neqI set4-neqI set43-neqI*
set53-neqI set54-neqI set5-neqI vector3-neq)[*18*]
by (*simp-all add*: *rat5-eq adjacent-points-def adjacency-table-def set3-neqI set4-neqI set43-neqI*
set53-neqI set54-neqI set5-neqI vector3-neq set3-vector-neq solids-vectors-to-rat5)

definition *mk-matrix33-ap* :: (*real^3*) *set* ⇒ (*real^3*) *set* ⇒ *real^3^3* **where**
mk-matrix33-ap f1 f2 = (*let vs* = *adjacent-points f1*; *us* = *adjacent-points f2 in*
mk-matrix33 (*fst vs*) (*fst* (*snd vs*)) (*snd* (*snd vs*))
(*fst us*) (*fst* (*snd us*)) (*snd* (*snd us*)))

method *show-congruence* =
(((*erule forw-subst*)+)?;
(*match* **conclusion in** *convex hull* (*s*::(*real^3*) *set*) *congruent convex hull* (*t*::(*real^3*) *set*) **for** *s t* ⇒
‹(*rule congruent-simple*), (*rule exI*[*where x*=*mk-matrix33-ap s t*])›);
(*simp only*: *mk-matrix33-ap-def Let-def fst-conv snd-conv adjacent-points*);
(*rule matrix-by-cramer-orthog* | *simp add*: *matrix-by-cramer det-3 rat5-eq-0 rat5-add rat5-mul rat5-sub*);
(*simp only*: *mk-matrix33-def vector-3 det-3 rat5-add rat5-mul rat5-sub rat5-eq*);
(*simp add*: *matrix-vector-mul-3 orthogonal-matrix33 rat5-add rat5-mul rat5-sub rat5-div rat5-eq*
rat5-eq-0 rat5-eq-1 Let-def; *fast*))

4.7 Equiangularity

vangle is from Manuel Eberl's AFP entry on triangles (see https://www.isa-afp.org/thys/Triangle/Angles.html)

definition *vangle* :: *'a*::*real-inner* ⇒ *'a* ⇒ *real* **where**
vangle x y = (*if x* = *0* ∨ *y* = *0 then pi* / *2 else arccos* (*x* · *y* / (*norm x* ∗ *norm y*)))

lemma *vangle-commute*:
shows *vangle x y* = *vangle y x*
unfolding *vangle-def inner-commute*[*of y x*]
by *argo*

lemma *isometry-maintains-angles-forw*:
fixes *c*
assumes *orthogonal-transformation h*
defines *g* ≡ (λ*x*. *c* + *h x*)
shows ((*e1 face-of f1* ∧ *aff-dim e1* = *1*) ∧
(*e2 face-of f1* ∧ *aff-dim e2* = *1*) ∧
e1 = *convex hull* {*v*, *a*} ∧ *e2* = *convex hull* {*v*, *b*} ∧
vangle (*a* − *v*) (*b* − *v*) = *t*) ⟹

$((g \text{ ` } e1 \text{ face-of } g \text{ ` } f1 \wedge \text{aff-dim } (g \text{ ` } e1) = 1) \wedge$
$(g \text{ ` } e2 \text{ face-of } g \text{ ` } f1 \wedge \text{aff-dim } (g \text{ ` } e2) = 1) \wedge$
$g \text{ ` } e1 = \text{convex hull } \{g\ v,\ g\ a\} \wedge g \text{ ` } e2 = \text{convex hull } \{g\ v,\ g\ b\} \wedge$
$\text{vangle } (g\ a - g\ v)\ (g\ b - g\ v) = t)$
(**is** *?lhs* $\Longrightarrow$ *?rhs*)
proof –
assume ∗: *?lhs*
then have $g \text{ ` } e1 \text{ face-of } g \text{ ` } f1 \wedge \text{aff-dim } (g \text{ ` } e1) = 1$
$g \text{ ` } e2 \text{ face-of } g \text{ ` } f1 \wedge \text{aff-dim } (g \text{ ` } e2) = 1$
using *face-of-isometry-eq*[*OF assms*(*1*)] *aff-dim-isometry-eq*[*OF assms*(*1*)] *g-def*
by *auto*
moreover have $g \text{ ` } e1 = \text{convex hull } \{g\ v,\ g\ a\}$
$g \text{ ` } e2 = \text{convex hull } \{g\ v,\ g\ b\}$
using ∗ *convex-hull-isometry*[*OF assms*(*1*), *of c*] *g-def*
by *simp-all*
moreover have $\text{vangle } (g\ a - g\ v)\ (g\ b - g\ v) = t$
proof –
have $a - v \neq 0$ $b - v \neq 0$
using ∗ **by** *auto*
have $\text{vangle } (g\ a - g\ v)\ (g\ b - g\ v) =$
$\text{arccos } ((g\ a - g\ v) \cdot (g\ b - g\ v) / (\text{norm } (g\ a - g\ v) * \text{norm } (g\ b - g\ v)))$
unfolding *vangle-def* **by** *simp*
also have ... $= \text{arccos } ((h\ a - h\ v) \cdot (h\ b - h\ v) / (\text{norm } (h\ a - h\ v) * \text{norm } (h\ b - h\ v)))$
unfolding *g-def* **by** *simp*
also have ... $= \text{arccos } (h\ (a - v) \cdot h\ (b - v) / (\text{norm } (h\ (a - v)) * \text{norm } (h\ (b - v))))$
using *orthogonal-transformation-linear*[*OF assms*(*1*)]
by (*simp add*: *linear-diff*)
also have ... $= \text{arccos } ((a - v) \cdot (b - v) / (\text{norm } (a - v) * \text{norm } (b - v)))$
unfolding *orthogonal-transformation-norm*[*OF assms*(*1*)]
by (*metis* (*mono-tags*, *opaque-lifting*) *assms*(*1*) *orthogonal-transformation-def*)
finally show *?thesis*
using ∗ *vangle-def* ‹$a - v \neq 0$› ‹$b - v \neq 0$›
by *metis*
qed
ultimately show *?rhs*
by *blast*
qed

definition *equiangular* :: $'a::\text{euclidean-space set} \Rightarrow \text{real} \Rightarrow \text{bool}$
where *equiangular f t* $\longleftrightarrow$
$(\forall\, e1\ e2\ v.\ (e1 \text{ face-of } f \wedge \text{aff-dim } e1 = 1) \wedge (e2 \text{ face-of } f \wedge \text{aff-dim } e2 = 1) \wedge$
$e1 \neq e2 \wedge v \text{ extreme-point-of } e1 \wedge v \text{ extreme-point-of } e2 \longrightarrow$
$(\exists\, a\ b.\ e1 = \text{convex hull } \{v,\ a\} \wedge e2 = \text{convex hull } \{v,\ b\} \wedge$
$\text{vangle } (a - v)\ (b - v) = t))$

lemma *equiangular-congruent-forw*:
  assumes *f1 congruent f2*
  assumes *equiangular f1 t*
  shows *equiangular f2 t*
proof −
  {
    fix *e1' e2' v'*
    assume ∗: (*e1' face-of f2* ∧ *aff-dim e1'* = *1*) ∧
              (*e2' face-of f2* ∧ *aff-dim e2'* = *1*) ∧
              e1' ≠ *e2'* ∧ *v' extreme-point-of e1'* ∧ *v' extreme-point-of e2'*
    obtain *c h* **where** *orthog*:*orthogonal-transformation h* **and** *f2*: *f2* = (λ*x*. *c* + *h x*) ‘ *f1*
      using *assms* **unfolding** *congruent-def* **by** *blast*
    then have *orthog-inv*: *orthogonal-transformation* (*inv h*)
      using *orthogonal-transformation-inv* **by** *blast*
    have *bounded-linear* (*inv h*)
      using *orthogonal-transformation-linear*[*OF orthog-inv*]
      by (*simp add*: *linear-conv-bounded-linear*)
    have (λ*y*. − *inv h c* + *inv h y*) ‘ *f2* = (λ*x*. −*inv h c* + *inv h* (*c* + *h x*)) ‘ *f1*
      using *f2* **by** *blast*
    also have ... = (λ*x*. −*inv h c* + *inv h c* + *inv h* (*h x*)) ‘ *f1*
      using ‹*bounded-linear* (*inv h*)›
      by (*simp add*: *linear-simps*)
    also have ... = (λ*x*. *inv h* (*h x*)) ‘ *f1*
      by *simp*
    finally have *f1*: *f1* = (λ*y*. − *inv h c* + *inv h y*) ‘ *f2*
    by (*simp add*: *orthog bijection.intro bijection.inv-left orthogonal-transformation-bij*)

    define *g* **where** *g* = (λ*x*. *c* + *h x*)
    define *g-inv* **where** *g-inv* = (λ*x*. − *inv h c* + *inv h x*)
    have *g-inv-g*: *g-inv* (*g y*) = *y* **for** *y*
      using *g-def g-inv-def*
        by (*simp add*: ‹*bounded-linear* (*inv h*)› *linear-simps*(*1*) *orthog orthogonal-transformation-inj*)
    then have *g-g-inv*: *g* (*g-inv x*) = *x* **for** *x*
      using *g-def g-inv-def*
      by (*metis add-minus-cancel bijection.intro*
          bijection.inv-right orthog orthogonal-transformation-bij)

    define *e1* **where** *e1* = *g-inv* ‘ *e1'*
    define *e2* **where** *e2* = *g-inv* ‘ *e2'*
    define *v* **where** *v* = *g-inv v'*

    have *e1-e2*: *e1 face-of f1* ∧ *aff-dim e1* = *1 e2 face-of f1* ∧ *aff-dim e2* = *1*
      using ∗ *e1-def e2-def f1 face-of-isometry-eq*[*OF orthog-inv*]
        aff-dim-isometry-eq[*OF orthog-inv*]
      unfolding *g-def g-inv-def*
      by *presburger*+
    moreover have *e1* ≠ *e2*

using *e1-def e2-def* $*$ *g-inv-g g-g-inv*
by (*metis* (*no-types*) *image-f-inv-f surj-def surj-imp-inv-eq*)
moreover have *v extreme-point-of e1 v extreme-point-of e2*
unfolding *e1-def e2-def v-def* **using** $*$ *extreme-point-of-isometry-eq*[*OF orthog-inv*] *g-inv-def*
by *blast*+
ultimately obtain *a b* **where** *ab*: $e1 = convex\ hull\ \{v,\ a\}$ $e2 = convex\ hull\ \{v,\ b\}$
$vangle\ (a - v)\ (b - v) = t$
using *assms*(*2*) **unfolding** *equiangular-def*
by *meson*

then have $g\ `\ e1 = convex\ hull\ \{g\ v,\ g\ a\} \wedge g\ `\ e2 = convex\ hull\ \{g\ v,\ g\ b\} \wedge$
$vangle\ (g\ a - g\ v)\ (g\ b - g\ v) = t$
using *isometry-maintains-angles-forw*[*OF orthog*, *of e1 f1 e2 v*] *g-def e1-e2*
by *presburger*
then have $\exists\, a\ b.\ e1' = convex\ hull\ \{v',\ a\} \wedge e2' = convex\ hull\ \{v',\ b\} \wedge$
$vangle\ (a - v')\ (b - v') = t$
using $*$ **unfolding** *e1-def e2-def v-def*
by (*metis* (*mono-tags*, *lifting*) *image-f-inv-f g-inv-g g-g-inv surj-def surj-imp-inv-eq*)
} **then show** *?thesis*
unfolding *equiangular-def*
by *blast*
qed

lemma *equiangular-congruent*:
assumes *f1 congruent f2*
shows *equiangular f1 t* $\longleftrightarrow$ *equiangular f2 t*
using *equiangular-congruent-forw assms congruent-sym*
by *blast*

end

5 Tetrahedron

theory *Tetrahedron*
imports *Computation*
begin

5.1 Definition

definition *std-tetrahedron* :: (*real*^*3*) *set*
where *std-tetrahedron* $\equiv$ *convex hull*
$\{vector\ [1,\ 1,\ 1],\ vector\ [-1,\ -1,\ 1],$
$vector\ [-1,\ 1,\ -1],\ vector\ [1,\ -1,\ -1]\}$

lemma *tetrahedron-fulldim*:
shows *aff-dim std-tetrahedron* = *3*
unfolding *std-tetrahedron-def*

apply (*rule polytope-3D*)
by (*simp add*: *vector3-sub vector3-cross vector3-dot vector3-eq-0*)+

lemma *tetrahedron-polyhedron*:
shows *polyhedron std-tetrahedron*
by (*simp add*: *std-tetrahedron-def polytope-convex-hull polytope-imp-polyhedron*)

5.2 Facets, edges, and vertices

lemma *tetrahedron-facets*:
f *face-of std-tetrahedron* $\wedge$ *aff-dim* $f = 2 \longleftrightarrow$
f = *convex hull* {*vector*[−*1*, −*1*, *1*], *vector*[− *1*, *1*, − *1*], *vector*[*1*, − *1*, − *1*]} ∨
f = *convex hull* {*vector*[−*1*, −*1*, *1*], *vector*[− *1*, *1*, − *1*], *vector*[*1*, *1*, *1*]} ∨
f = *convex hull* {*vector*[−*1*, −*1*, *1*], *vector*[*1*, − *1*, − *1*], *vector*[*1*, *1*, *1*]} ∨
f = *convex hull* {*vector*[−*1*, *1*, −*1*], *vector*[*1*, − *1*, − *1*], *vector*[*1*, *1*, *1*]}
unfolding *std-tetrahedron-def*
apply (*simp only*: *compute-faces-2 finite-insert finite.emptyI*)
apply (*simp only*: *compute-faces-2-step-1 bex-empty simp-thms*(*31*))
apply (*simp only*: *compute-faces-2-step-2 bex-empty simp-thms*(*31*))
apply (*simp only*: *empty-iff ball-insert bex-empty bex-simps*(*5*) *ball-simps*(*5*) *simp-thms*(*21*,*31*))
apply (*simp add*: *vector3-sub vector3-cross vector3-eq-0 vector3-dot Let-def*)
apply (*simp only*: *insert-commute*)
by *linarith*

lemma *tetrahedron-facet1-edges*:
defines *v1* ≡ *vector*[−*1*, −*1*, *1*] :: *real^3*
and *v2* ≡ *vector*[−*1*, *1*, −*1*] :: *real^3*
and *v3* ≡ *vector*[*1*, −*1*, −*1*] :: *real^3*
and *n* ≡ *vector*[−*4*, −*4*, −*4*] :: *real^3*
shows (*e face-of* (*convex hull* {*v1*, *v2*, *v3*}) ∧ *aff-dim e* = *1*) =
(*e* = *convex hull* {*v1*, *v2*} ∨ *e* = *convex hull* {*v1*, *v3*} ∨ *e* = *convex hull* {*v2*, *v3*})
proof −
have *n-nz*: $n \neq 0$
unfolding *n-def* **by** (*simp add*: *vector3-eq-0*)
have *plane*: $\bigwedge x.\ x \in \{v1, v2, v3\} \Longrightarrow n \cdot x = 4$
by (*auto simp add*: *assms vector3-dot*)
have *finite*: *finite* {*v1*, *v2*, *v3*}
using *assms* **by** *blast*
show *?thesis*
by (*facet-edges plane*: *plane n-nz*: *n-nz defs*: *assms*)
qed

lemma *tetrahedron-facet2-edges*:
defines *v1* ≡ *vector*[−*1*,−*1*,*1*] :: *real^3*
and *v2* ≡ *vector*[−*1*,*1*,−*1*] :: *real^3*
and *v3* ≡ *vector*[*1*,*1*,*1*] :: *real^3*

and $n \equiv vector[-4, 4, 4] :: real\hat{}3$
shows $(e$ *face-of* $(convex\ hull\ \{v1, v2, v3\}) \wedge aff\text{-}dim\ e = 1) =$
$(e = convex\ hull\ \{v1, v2\} \vee e = convex\ hull\ \{v1, v3\} \vee e = convex\ hull\ \{v2, v3\})$
proof –
have *n-nz*: $n \neq 0$
unfolding *n-def* **by** (*simp add*: *vector3-eq-0*)
have *plane*: $\bigwedge x.\ x \in \{v1, v2, v3\} \Longrightarrow n \cdot x = 4$
by (*auto simp add*: *assms vector3-dot*)
have *finite*: *finite* $\{v1, v2, v3\}$
using *assms* **by** *blast*
show *?thesis*
by (*facet-edges plane*: *plane n-nz*: *n-nz defs*: *assms*)
qed

lemma *tetrahedron-facet3-edges*:
defines $v1 \equiv vector[-1,-1,1] :: real\hat{}3$
and $v2 \equiv vector[1,-1,-1] :: real\hat{}3$
and $v3 \equiv vector[1,1,1] :: real\hat{}3$
and $n \equiv vector[4, -4, 4] :: real\hat{}3$
shows $(e$ *face-of* $(convex\ hull\ \{v1, v2, v3\}) \wedge aff\text{-}dim\ e = 1) =$
$(e = convex\ hull\ \{v1, v2\} \vee e = convex\ hull\ \{v1, v3\} \vee e = convex\ hull\ \{v2, v3\})$
proof –
have *n-nz*: $n \neq 0$
unfolding *n-def* **by** (*simp add*: *vector3-eq-0*)
have *plane*: $\bigwedge x.\ x \in \{v1, v2, v3\} \Longrightarrow n \cdot x = 4$
by (*auto simp add*: *assms vector3-dot*)
have *finite*: *finite* $\{v1, v2, v3\}$
using *assms* **by** *blast*
show *?thesis*
by (*facet-edges plane*: *plane n-nz*: *n-nz defs*: *assms*)
qed

lemma *tetrahedron-facet4-edges*:
defines $v1 \equiv vector[-1,1,-1] :: real\hat{}3$
and $v2 \equiv vector[1,-1,-1] :: real\hat{}3$
and $v3 \equiv vector[1,1,1] :: real\hat{}3$
and $n \equiv vector[4, 4, -4] :: real\hat{}3$
shows $(e$ *face-of* $(convex\ hull\ \{v1, v2, v3\}) \wedge aff\text{-}dim\ e = 1) =$
$(e = convex\ hull\ \{v1, v2\} \vee e = convex\ hull\ \{v1, v3\} \vee e = convex\ hull\ \{v2, v3\})$
proof –
have *n-nz*: $n \neq 0$
unfolding *n-def* **by** (*simp add*: *vector3-eq-0*)
have *plane*: $\bigwedge x.\ x \in \{v1, v2, v3\} \Longrightarrow n \cdot x = 4$
by (*auto simp add*: *assms vector3-dot*)
have *finite*: *finite* $\{v1, v2, v3\}$
using *assms* **by** *blast*

show *?thesis*
  by (*facet-edges plane*: *plane n-nz*: *n-nz defs*: *assms*)
qed

lemmas *tetrahedron-facet-edges* =
tetrahedron-facet1-edges tetrahedron-facet2-edges
tetrahedron-facet3-edges tetrahedron-facet4-edges

lemma *tetrahedron-edges-per-face*:
assumes *f face-of std-tetrahedron*
  and *aff-dim f* = *2*
shows *card* {*e*. *e face-of std-tetrahedron* ∧ *aff-dim e* = *1* ∧ *e* ⊆ *f*} = *3*
using *conjI*[*OF assms*] **unfolding** *tetrahedron-facets face-edge-set-eq*[*OF assms*(*1*)]
apply (*elim disjE forw-subst*)
unfolding *tetrahedron-facet-edges set-cases card-3-iff2*
segment-convex-hull[*symmetric*] *closed-segment-eq*
by (*simp-all add*: *vector3-eq doubleton-eq-iff*)

lemma *tetrahedron-edges*:
e face-of std-tetrahedron ∧ *aff-dim e* = *1* ⟷
e = *convex hull* {*vector* [−*1*, −*1*, *1*], *vector* [−*1*, *1*, −*1*]} ∨
e = *convex hull* {*vector* [−*1*, −*1*, *1*], *vector* [*1*, −*1*, −*1*]} ∨
e = *convex hull* {*vector* [−*1*, −*1*, *1*], *vector* [*1*, *1*, *1*]} ∨
e = *convex hull* {*vector* [−*1*, *1*, −*1*], *vector* [*1*, −*1*, −*1*]} ∨
e = *convex hull* {*vector* [−*1*, *1*, −*1*], *vector* [*1*, *1*, *1*]} ∨
e = *convex hull* {*vector* [*1*, −*1*, −*1*], *vector* [*1*, *1*, *1*]}
(**is** *?lhs* ⟷ *?rhs*)
proof
assume ∗: *?lhs*
then obtain *f* **where** *f-facts*: *f face-of std-tetrahedron* ∧ *aff-dim f* = *2 e face-of f* ∧ *aff-dim e* = *1*
  using *tetrahedron-polyhedron tetrahedron-fulldim edge-belongs-to-face*
  by *blast*
show *?rhs*
  using *f-facts* **unfolding** *tetrahedron-facets*
  apply (*elim disjE*) **apply** (*all* ‹*hypsubst-thin*›)
  unfolding *tetrahedron-facet-edges*
  by *linarith*+
next
assume *?rhs*
then show *?lhs*
  using *tetrahedron-facets tetrahedron-facet-edges*
  by (*smt* (*verit*, *ccfv-threshold*) *face-of-trans insertI1 insert-commute mem-Collect-eq*)
qed

lemma *tetrahedron-vertices*:
shows (*v face-of std-tetrahedron* ∧ *aff-dim v* = *0*) =
(*v* = {*vector* [− *1*, − *1*, *1*]} ∨
v = {*vector* [− *1*, *1*, − *1*]} ∨

$v = \{vector\ [1,\ -\ 1,\ -\ 1]\} \lor$
$v = \{vector\ [1,\ 1,\ 1]\})$
(**is** *?lhs* = *?rhs*)
proof
assume $*$: *?lhs*
then have $aff\text{-}dim\ v = 0$
by *blast*
obtain *e* **where** *e-facts*: $e\ face\text{-}of\ std\text{-}tetrahedron \land aff\text{-}dim\ e = 1\ v\ face\text{-}of\ e$
using $*$ *tetrahedron-polyhedron tetrahedron-fulldim vertex-belongs-to-edge*
by *blast*
show *?rhs*
using *e-facts*(*1*)
unfolding *tetrahedron-edges*
using *vertices-of-edge*[*OF* - ‹*v face-of e*› ‹$aff\text{-}dim\ v = 0$›]
by *metis*
next
assume *?rhs*
then show *?lhs*
apply (*elim disjE*)
by (*metis* (*no-types*, *opaque-lifting*) *aff-dim-sing extreme-point-of-segment face-of-singleton face-of-trans segment-convex-hull tetrahedron-edges*)+
qed

lemma *tetrahedron-edges-per-vertex*:
assumes *v face-of std-tetrahedron*
and $aff\text{-}dim\ v = 0$
shows $card\ \{e.\ e\ face\text{-}of\ std\text{-}tetrahedron \land aff\text{-}dim\ e = 1 \land v \subseteq e\} = 3$
unfolding *vertex-edge-set-eq*[*OF assms*(*1*)]
using *conjI*[*OF assms*]

unfolding *tetrahedron-vertices conj-assoc*[*symmetric*] *tetrahedron-edges*
by (*elim disjE forw-subst*) *edges-per-vertex*+

5.3 Regularity

lemma *tetrahedron-congruent-edges*:
assumes $e1\ face\text{-}of\ std\text{-}tetrahedron \land aff\text{-}dim\ e1 = 1$
and $e2\ face\text{-}of\ std\text{-}tetrahedron \land aff\text{-}dim\ e2 = 1$
shows *e1 congruent e2*
proof −
let *?edges* =
$\{convex\ hull\ \{(vector\ [-1,\ -1,\ 1]::real\hat{\ }3),\ vector\ [-1,\ 1,\ -1]\},$
$convex\ hull\ \{vector\ [-1,\ -1,\ 1],\ vector\ [1,\ -1,\ -1]\},$
$convex\ hull\ \{vector\ [-1,\ -1,\ 1],\ vector\ [1,\ 1,\ 1]\},$
$convex\ hull\ \{vector\ [-1,\ 1,\ -1],\ vector\ [1,\ -1,\ -1]\},$
$convex\ hull\ \{vector\ [-1,\ 1,\ -1],\ vector\ [1,\ 1,\ 1]\},$
$convex\ hull\ \{vector\ [1,\ -1,\ -1],\ vector\ [1,\ 1,\ 1]\}\}$
have $e1 \in\ ?edges\ e2 \in\ ?edges$
using *assms* **unfolding** *tetrahedron-edges*

by *auto*
moreover have ∀ *e*∈*?edges*. *convex hull* {*vector* [−*1*, −*1*, *1*], *vector* [−*1*, *1*, −*1*]} *congruent e*
apply (*simp only*: *ball-simps*(*7*) *Set.ball-empty simp-thms*(*21*))
unfolding *conj-assoc*[*symmetric*] *segment-convex-hull*[*symmetric*]
apply (*rule conjI*)+
apply (*all* ‹*rule congruent-segments*›)
unfolding *dist-norm vector3-sub*
apply (*simp-all add*: *norm-eq*)
unfolding *vector3-dot*
by *linarith*+
ultimately show *?thesis*
using *congruent-set* **by** *meson*
qed

lemma *tetrahedron-congruent-faces*:
assumes *f1 face-of std-tetrahedron* ∧ *aff-dim f1* = *2*
and *f2 face-of std-tetrahedron* ∧ *aff-dim f2* = *2*
shows *f1 congruent f2*
proof −
let *?faces* =
{*convex hull* {(*vector*[−*1*, −*1*, *1*]::*real^3*), *vector*[− *1*, *1*, − *1*], *vector*[*1*, − *1*, − *1*]},
convex hull {*vector*[−*1*, −*1*, *1*], *vector*[− *1*, *1*, − *1*], *vector*[*1*, *1*, *1*]},
convex hull {*vector*[−*1*, −*1*, *1*], *vector*[*1*, − *1*, − *1*], *vector*[*1*, *1*, *1*]},
convex hull {*vector*[−*1*, *1*, −*1*], *vector*[*1*, − *1*, − *1*], *vector*[*1*, *1*, *1*]}}
have *f1* ∈ *?faces f2* ∈ *?faces*
using *assms tetrahedron-facets*
by *auto*
moreover have ∀ *f*∈*?faces*.
convex hull {*vector*[−*1*, −*1*, *1*], *vector*[− *1*, *1*, − *1*], *vector*[*1*, − *1*, − *1*]} *congruent f*
apply (*simp only*: *ball-simps*(*7*) *Set.ball-empty simp-thms*(*21*))
unfolding *conj-assoc*[*symmetric*]
by (*rule conjI*)+ *show-congruence*+
ultimately show *?thesis*
using *congruent-set* **by** *meson*
qed

lemma *tetrahedron-facet1-equiangular*:
defines *v1* ≡ *vector*[−*1*, −*1*, *1*] :: *real^3*
defines *v2* ≡ *vector*[−*1*, *1*, −*1*] :: *real^3*
defines *v3* ≡ *vector*[*1*, −*1*, −*1*] :: *real^3*
shows *equiangular* (*convex hull* {*v1*, *v2*, *v3*}) (*pi* / *3*)
unfolding *equiangular-def*
proof (*clarify*)
fix *e1 e2 v*
assume ∗: *e1 face-of convex hull* {*v1*, *v2*, *v3*} *aff-dim e1* = *1*
e2 face-of convex hull {*v1*, *v2*, *v3*} *aff-dim e2* = *1*

e1 ≠ *e2* *v extreme-point-of e1* *v extreme-point-of e2*
have (*v* = *v1* ∧ ((*e1* = *convex hull* {*v1*, *v2*} ∧ *e2* = *convex hull* {*v1*, *v3*}) ∨
(*e1* = *convex hull* {*v1*, *v3*} ∧ *e2* = *convex hull* {*v1*, *v2*}))) ∨
(*v* = *v2* ∧ ((*e1* = *convex hull* {*v2*, *v1*} ∧ *e2* = *convex hull* {*v2*, *v3*}) ∨
(*e1* = *convex hull* {*v2*, *v3*} ∧ *e2* = *convex hull* {*v2*, *v1*}))) ∨
(*v* = *v3* ∧ ((*e1* = *convex hull* {*v3*, *v1*} ∧ *e2* = *convex hull* {*v3*, *v2*}) ∨
(*e1* = *convex hull* {*v3*, *v2*} ∧ *e2* = *convex hull* {*v3*, *v1*})))
using *conjI*[*OF* ∗(*1*,*2*)] *conjI*[*OF* ∗(*3*,*4*)] ∗(*5*−*7*) **unfolding** *assms tetrahedron-facet1-edges*
by (*smt* (*verit*, *best*) *extreme-point-of-convex-hull-2 insert-commute vector3-eq*)
moreover have *vangle* (*v2* − *v1*) (*v3* − *v1*) = *pi* / *3*
vangle (*v1* − *v2*) (*v3* − *v2*) = *pi* / *3*
vangle (*v1* − *v3*) (*v2* − *v3*) = *pi* / *3*
unfolding *assms vector3-sub norm-eq-sqrt-inner vector3-dot vangle-def*
by (*simp-all add*: *vector3-eq-0*)
ultimately show (∃ *a b*. *e1* = *convex hull* {*v*, *a*} ∧ *e2* = *convex hull* {*v*, *b*} ∧
vangle (*a* − *v*) (*b* − *v*) = *pi* / *3*)
using *vangle-commute*
by *metis*
qed

lemma *tetrahedron-equiangular*:
assumes *f face-of std-tetrahedron* ∧ *aff-dim f* = *2*
shows *equiangular f* (*pi* / *3*)
proof −
define *f1* **where** *f1* = *convex hull* {*vector*[−*1*, −*1*, *1*]::*real^3*, *vector*[−*1*, *1*, −*1*], *vector*[*1*, −*1*, −*1*]}
have *f1 face-of std-tetrahedron* ∧ *aff-dim f1* = *2*
using *f1-def tetrahedron-facets* **by** *blast*
then have *f1 congruent f*
using *assms tetrahedron-congruent-faces* **by** *blast*
then show *?thesis*
using *equiangular-congruent tetrahedron-facet1-equiangular f1-def*
by *blast*
qed

end

6 Cube

theory *Cube*
imports *Computation*
begin

6.1 Definition

definition *std-cube* :: (*real*, *3*) *vec set*
where *std-cube* ≡ *convex hull*
{*vector* [*1*, *1*, *1*], *vector* [*1*, *1*, −*1*],

vector [*1*, −*1*, *1*], *vector* [*1*, −*1*, −*1*],
vector [−*1*, *1*, *1*], *vector* [−*1*, *1*, −*1*],
vector [−*1*, −*1*, *1*], *vector* [−*1*, −*1*, −*1*]}

lemma *cube-fulldim*:
shows *aff-dim std-cube* = *3*
unfolding *std-cube-def*
apply (*rule polytope-3D*)
by (*simp add*: *vector3-sub vector3-cross vector3-dot vector3-eq-0*)+

lemma *cube-polyhedron*:
shows *polyhedron std-cube*
by (*simp add*: *std-cube-def polytope-convex-hull polytope-imp-polyhedron*)

6.2 Facets, edges, and vertices

lemma *cube-facets*:
fixes *f* :: (*real^3*) *set*
shows *f face-of std-cube* ∧ *aff-dim f* = *2* ⟷
f = *convex hull* {*vector*[− *1*, − *1*, − *1*], *vector*[− *1*, − *1*, *1*], *vector*[− *1*, *1*, − *1*], *vector*[− *1*, *1*, *1*]} ∨
f = *convex hull* {*vector*[− *1*, − *1*, − *1*], *vector*[− *1*, − *1*, *1*], *vector*[*1*, − *1*, − *1*], *vector*[*1*, − *1*, *1*]} ∨
f = *convex hull* {*vector*[− *1*, − *1*, − *1*], *vector*[− *1*, *1*, − *1*], *vector*[*1*, − *1*, − *1*], *vector*[*1*, *1*, − *1*]} ∨
f = *convex hull* {*vector*[− *1*, − *1*, *1*], *vector*[− *1*, *1*, *1*], *vector*[*1*, − *1*, *1*], *vector*[*1*, *1*, *1*]} ∨
f = *convex hull* {*vector*[− *1*, *1*, − *1*], *vector*[− *1*, *1*, *1*], *vector*[*1*, *1*, − *1*], *vector*[*1*, *1*, *1*]} ∨
f = *convex hull* {*vector*[*1*, − *1*, − *1*], *vector*[*1*, − *1*, *1*], *vector*[*1*, *1*, − *1*], *vector*[*1*, *1*, *1*]}
unfolding *std-cube-def*
apply (*simp only*: *compute-faces-2 finite-insert finite.emptyI*)
apply (*simp only*: *compute-faces-2-step-1 bex-empty simp-thms*(*31*))
apply (*simp only*: *compute-faces-2-step-2 bex-empty simp-thms*(*31*))
apply (*simp only*: *empty-iff ball-insert bex-empty bex-simps*(*5*) *ball-simps*(*5*) *simp-thms*(*21*,*31*))
apply (*simp add*: *vector3-sub vector3-cross vector3-eq-0 vector3-dot Let-def*)
apply (*simp only*: *insert-commute*)
by *linarith*

lemma *cube-facet1-edges*:
defines *v1* ≡ *vector*[−*1*, −*1*, −*1*] :: *real^3*
and *v2* ≡ *vector*[−*1*, −*1*, *1*] :: *real^3*
and *v3* ≡ *vector*[−*1*, *1*, −*1*] :: *real^3*
and *v4* ≡ *vector*[−*1*, *1*, *1*] :: *real^3*
and *n* ≡ *vector*[*1*, *0*, *0*] :: *real^3*
shows (*e face-of* (*convex hull* {*v1*, *v2*, *v3*, *v4*}) ∧ *aff-dim e* = *1*) =
(*e* = *convex hull* {*v1* ,*v2*} ∨

$e = \textit{convex hull}\ \{v1,\ v3\} \vee$
$e = \textit{convex hull}\ \{v2,\ v4\} \vee$
$e = \textit{convex hull}\ \{v3,\ v4\})$

proof −
have *n-nz*: $n \neq 0$
unfolding *n-def* **by** (*simp add*: *vector3-eq-0*)
have *plane*: $\bigwedge x.\ x \in \{v1,\ v2,\ v3,\ v4\} \Longrightarrow n \cdot x = -1$
unfolding *assms* **using** *vector3-dot* **by** *auto*
show *?thesis*
by (*facet-edges plane*: *plane n-nz*: *n-nz defs*: *assms*)
qed

lemma *cube-facet2-edges*:
defines $v1 \equiv vector[-1,\ -1,\ -1] :: real^\frown 3$
and $v2 \equiv vector[-1,\ -1,\ 1] :: real^\frown 3$
and $v3 \equiv vector[1,\ -1,\ -1] :: real^\frown 3$
and $v4 \equiv vector[1,\ -1,\ 1] :: real^\frown 3$
and $n \equiv vector[0,\ 1,\ 0] :: real^\frown 3$
shows $(e\ \textit{face-of}\ (\textit{convex hull}\ \{v1,\ v2,\ v3,\ v4\}) \wedge \textit{aff-dim}\ e = 1) =$
$(e = \textit{convex hull}\ \{v1\ ,v2\} \vee$
$e = \textit{convex hull}\ \{v1,\ v3\} \vee$
$e = \textit{convex hull}\ \{v2,\ v4\} \vee$
$e = \textit{convex hull}\ \{v3,\ v4\})$
proof −
have *n-nz*: $n \neq 0$
unfolding *n-def* **by** (*simp add*: *vector3-eq-0*)
have *plane*: $\bigwedge x.\ x \in \{v1,\ v2,\ v3,\ v4\} \Longrightarrow n \cdot x = -1$
unfolding *assms* **using** *vector3-dot* **by** *auto*
show *?thesis*
by (*facet-edges plane*: *plane n-nz*: *n-nz defs*: *assms*)
qed

lemma *cube-facet3-edges*:
defines $v1 \equiv vector[-1,\ -1,\ -1] :: real^\frown 3$
and $v2 \equiv vector[-1,\ 1,\ -1] :: real^\frown 3$
and $v3 \equiv vector[1,\ -1,\ -1] :: real^\frown 3$
and $v4 \equiv vector[1,\ 1,\ -1] :: real^\frown 3$
and $n \equiv vector[0,\ 0,\ 1] :: real^\frown 3$
shows $(e\ \textit{face-of}\ (\textit{convex hull}\ \{v1,\ v2,\ v3,\ v4\}) \wedge \textit{aff-dim}\ e = 1) =$
$(e = \textit{convex hull}\ \{v1\ ,v2\} \vee$
$e = \textit{convex hull}\ \{v1,\ v3\} \vee$
$e = \textit{convex hull}\ \{v2,\ v4\} \vee$
$e = \textit{convex hull}\ \{v3,\ v4\})$
proof −
have *n-nz*: $n \neq 0$
unfolding *n-def* **by** (*simp add*: *vector3-eq-0*)
have *plane*: $\bigwedge x.\ x \in \{v1,\ v2,\ v3,\ v4\} \Longrightarrow n \cdot x = -1$
unfolding *assms* **using** *vector3-dot* **by** *auto*
show *?thesis*

    by (*facet-edges plane*: *plane n-nz*: *n-nz defs*: *assms*)
qed

lemma *cube-facet4-edges*:
  defines $v1 \equiv vector[-1,\ -1,\ 1] :: real\^{}3$
    and $v2 \equiv vector[-1,\ 1,\ 1] :: real\^{}3$
    and $v3 \equiv vector[1,\ -1,\ 1] :: real\^{}3$
    and $v4 \equiv vector[1,\ 1,\ 1] :: real\^{}3$
    and $n \equiv vector[0,\ 0,\ 1] :: real\^{}3$
  shows $(e\ face\text{-}of\ (convex\ hull\ \{v1,\ v2,\ v3,\ v4\}) \wedge aff\text{-}dim\ e = 1) =$
    $(e = convex\ hull\ \{v1\ ,v2\} \vee$
    $e = convex\ hull\ \{v1,\ v3\} \vee$
    $e = convex\ hull\ \{v2,\ v4\} \vee$
    $e = convex\ hull\ \{v3,\ v4\})$
proof –
  have *n-nz*: $n \neq 0$
    unfolding *n-def* **by** (*simp add*: *vector3-eq-0*)
  have *plane*: $\bigwedge x.\ x \in \{v1,\ v2,\ v3,\ v4\} \Longrightarrow n \cdot x = 1$
    unfolding *assms* **using** *vector3-dot* **by** *auto*
  show *?thesis*
    by (*facet-edges plane*: *plane n-nz*: *n-nz defs*: *assms*)
qed

lemma *cube-facet5-edges*:
  defines $v1 \equiv vector[-1,\ 1,\ -1] :: real\^{}3$
    and $v2 \equiv vector[-1,\ 1,\ 1] :: real\^{}3$
    and $v3 \equiv vector[1,\ 1,\ -1] :: real\^{}3$
    and $v4 \equiv vector[1,\ 1,\ 1] :: real\^{}3$
    and $n \equiv vector[0,\ 1,\ 0] :: real\^{}3$
  shows $(e\ face\text{-}of\ (convex\ hull\ \{v1,\ v2,\ v3,\ v4\}) \wedge aff\text{-}dim\ e = 1) =$
    $(e = convex\ hull\ \{v1\ ,v2\} \vee$
    $e = convex\ hull\ \{v1,\ v3\} \vee$
    $e = convex\ hull\ \{v2,\ v4\} \vee$
    $e = convex\ hull\ \{v3,\ v4\})$
proof –
  have *n-nz*: $n \neq 0$
    unfolding *n-def* **by** (*simp add*: *vector3-eq-0*)
  have *plane*: $\bigwedge x.\ x \in \{v1,\ v2,\ v3,\ v4\} \Longrightarrow n \cdot x = 1$
    unfolding *assms* **using** *vector3-dot* **by** *auto*
  show *?thesis*
    by (*facet-edges plane*: *plane n-nz*: *n-nz defs*: *assms*)
qed

lemma *cube-facet6-edges*:
  defines $v1 \equiv vector[1,\ -1,\ -1] :: real\^{}3$
    and $v2 \equiv vector[1,\ -1,\ 1] :: real\^{}3$
    and $v3 \equiv vector[1,\ 1,\ -1] :: real\^{}3$
    and $v4 \equiv vector[1,\ 1,\ 1] :: real\^{}3$
    and $n \equiv vector[1,\ 0,\ 0] :: real\^{}3$

shows (*e face-of* (*convex hull* {*v1*, *v2*, *v3*, *v4*}) ∧ *aff-dim e* = *1*) =
        (*e* = *convex hull* {*v1* ,*v2*} ∨
         e = *convex hull* {*v1*, *v3*} ∨
         e = *convex hull* {*v2*, *v4*} ∨
         e = *convex hull* {*v3*, *v4*})
proof −
  have *n-nz*: *n* ≠ *0*
    unfolding *n-def* **by** (*simp add*: *vector3-eq-0*)
  have *plane*: ⋀*x*. *x* ∈ {*v1*, *v2*, *v3*, *v4*} ⟹ *n* · *x* = *1*
    unfolding *assms* **using** *vector3-dot* **by** *auto*
  show *?thesis*
    by (*facet-edges plane*: *plane n-nz*: *n-nz defs*: *assms*)
qed

lemmas *cube-facet-edges* =
  cube-facet1-edges cube-facet2-edges cube-facet3-edges
  cube-facet4-edges cube-facet5-edges cube-facet6-edges

lemma *cube-edges-per-face*:
  assumes *f face-of std-cube*
      and *aff-dim f* = *2*
  shows *card* {*e*. *e face-of std-cube* ∧ *aff-dim e* = *1* ∧ *e* ⊆ *f*} = *4*
  using *conjI*[*OF assms*] **unfolding** *cube-facets face-edge-set-eq*[*OF assms*(*1*)]
  apply (*elim disjE forw-subst*)
  unfolding *cube-facet-edges set-cases card-4-iff2*
    segment-convex-hull[*symmetric*] *closed-segment-eq*
  by (*simp-all add*: *vector3-eq doubleton-eq-iff*)

lemma *cube-edges*:
  e face-of std-cube ∧ *aff-dim e* = *1* ⟷
   e = *convex hull* {*vector*[− *1*, − *1*, − *1*], *vector*[− *1*, − *1*, *1*]} ∨
   e = *convex hull* {*vector*[− *1*, − *1*, − *1*], *vector*[− *1*, *1*, − *1*]} ∨
   e = *convex hull* {*vector*[− *1*, − *1*, − *1*], *vector*[*1*, − *1*, − *1*]} ∨
   e = *convex hull* {*vector*[− *1*, − *1*, *1*], *vector*[− *1*, *1*, *1*]} ∨
   e = *convex hull* {*vector*[− *1*, − *1*, *1*], *vector*[*1*, − *1*, *1*]} ∨
   e = *convex hull* {*vector*[− *1*, *1*, − *1*], *vector*[− *1*, *1*, *1*]} ∨
   e = *convex hull* {*vector*[− *1*, *1*, − *1*], *vector*[*1*, *1*, − *1*]} ∨
   e = *convex hull* {*vector*[− *1*, *1*, *1*], *vector*[*1*, *1*, *1*]} ∨
   e = *convex hull* {*vector*[*1*, − *1*, − *1*], *vector*[*1*, − *1*, *1*]} ∨
   e = *convex hull* {*vector*[*1*, − *1*, − *1*], *vector*[*1*, *1*, − *1*]} ∨
   e = *convex hull* {*vector*[*1*, − *1*, *1*], *vector*[*1*, *1*, *1*]} ∨
   e = *convex hull* {*vector*[*1*, *1*, − *1*], *vector*[*1*, *1*, *1*]}
  (**is** *?lhs* ⟷ *?rhs*)
proof
  assume ∗: *?lhs*
  then obtain *f* **where** *f-facts*: *f face-of std-cube* ∧ *aff-dim f* = *2 e face-of f* ∧ *aff-dim e* = *1*
    using *cube-polyhedron cube-fulldim edge-belongs-to-face*
    by *blast*

```
  show ?rhs
    using f-facts unfolding cube-facets
    apply (elim disjE) apply (all ‹hypsubst-thin›)
    unfolding cube-facet-edges
    by linarith+
next
  assume ?rhs
  then show ?lhs
    using cube-facets cube-facet-edges
   by (smt (verit, ccfv-threshold) face-of-trans insertI1 insert-commute mem-Collect-eq)
qed

lemma cube-vertices:
  shows (v face-of std-cube ∧ aff-dim v = 0) =
        (v = {vector [1, 1, 1]} ∨
         v = {vector [1, 1, − 1]} ∨
         v = {vector [1, −1, 1]} ∨
         v = {vector [1, −1, −1]} ∨
         v = {vector [−1, 1, 1]} ∨
         v = {vector [−1, 1, −1]} ∨
         v = {vector [−1, −1, 1]} ∨
         v = {vector [−1, −1, −1]})
  (is ?lhs = ?rhs)
proof
  assume ∗: ?lhs
  obtain e where e face-of std-cube aff-dim e = 1 v face-of e
    using ∗ cube-polyhedron cube-fulldim vertex-belongs-to-edge
    by blast
  show ?rhs
    using conjI[OF ‹e face-of std-cube› ‹aff-dim e = 1›]
    unfolding cube-edges
    apply (elim disjE)
    using vertices-of-edge[OF - ‹v face-of e›] ∗
    by metis+
next
  assume ?rhs
  then show ?lhs
    apply (elim disjE)
    by (meson aff-dim-sing extreme-point-of-convex-hull-2 face-of-singleton
        face-of-trans cube-edges)+
qed

lemma cube-edges-per-vertex:
  assumes v face-of std-cube
      and aff-dim v = 0
  shows card {e. e face-of std-cube ∧ aff-dim e = 1 ∧ v ⊆ e} = 3
  unfolding vertex-edge-set-eq[OF assms(1)]
  using conjI[OF assms]
  unfolding cube-vertices conj-assoc[symmetric] cube-edges
```

by (*elim disjE forw-subst*) *edges-per-vertex*+

6.3 Regularity

lemma *cube-congruent-edges*:
assumes *e1 face-of std-cube* ∧ *aff-dim e1* = *1*
and *e2 face-of std-cube* ∧ *aff-dim e2* = *1*
shows *e1 congruent e2*
proof −
let *?edges* =
{*convex hull* {(*vector*[− *1*, − *1*, − *1*]::*real*^*3*), *vector*[− *1*, − *1*, *1*]},
convex hull {*vector*[− *1*, − *1*, − *1*], *vector*[− *1*, *1*, − *1*]},
convex hull {*vector*[− *1*, − *1*, − *1*], *vector*[*1*, − *1*, − *1*]},
convex hull {*vector*[− *1*, − *1*, *1*], *vector*[− *1*, *1*, *1*]},
convex hull {*vector*[− *1*, − *1*, *1*], *vector*[*1*, − *1*, *1*]},
convex hull {*vector*[− *1*, *1*, − *1*], *vector*[− *1*, *1*, *1*]},
convex hull {*vector*[− *1*, *1*, − *1*], *vector*[*1*, *1*, − *1*]},
convex hull {*vector*[− *1*, *1*, *1*], *vector*[*1*, *1*, *1*]},
convex hull {*vector*[*1*, − *1*, − *1*], *vector*[*1*, − *1*, *1*]},
convex hull {*vector*[*1*, − *1*, − *1*], *vector*[*1*, *1*, − *1*]},
convex hull {*vector*[*1*, − *1*, *1*], *vector*[*1*, *1*, *1*]},
convex hull {*vector*[*1*, *1*, − *1*], *vector*[*1*, *1*, *1*]}}
have *e1* ∈ *?edges e2* ∈ *?edges*
using *assms* **unfolding** *cube-edges*
by *auto*
moreover have ∀ *e*∈*?edges*. *convex hull* {*vector*[− *1*, − *1*, − *1*], *vector*[− *1*, − *1*, *1*]} *congruent e*
apply (*simp only*: *ball-simps*(*7*) *Set.ball-empty simp-thms*(*21*))
unfolding *conj-assoc*[*symmetric*] *segment-convex-hull*[*symmetric*]
apply (*rule conjI*)+
apply (*all* ‹*rule congruent-segments*›)
unfolding *dist-norm vector3-sub*
apply (*simp-all add*: *norm-eq*)
unfolding *vector3-dot*
by *linarith*+
ultimately show *?thesis*
using *congruent-set* **by** *meson*
qed

lemma *cube-congruent-faces*:
assumes *f1 face-of std-cube* ∧ *aff-dim f1* = *2*
and *f2 face-of std-cube* ∧ *aff-dim f2* = *2*
shows *f1 congruent f2*
proof −
let *?faces* =
{*convex hull* {(*vector*[− *1*, − *1*, − *1*]::*real*^*3*), *vector*[− *1*, − *1*, *1*], *vector*[− *1*, *1*, − *1*], *vector*[− *1*, *1*, *1*]},
convex hull {*vector*[− *1*, − *1*, − *1*], *vector*[− *1*, − *1*, *1*], *vector*[*1*, − *1*, − *1*], *vector*[*1*, − *1*, *1*]},

convex hull {*vector*[− *1*, − *1*, − *1*], *vector*[− *1*, *1*, − *1*], *vector*[*1*, − *1*, − *1*], *vector*[*1*, *1*, − *1*]},
convex hull {*vector*[− *1*, − *1*, *1*], *vector*[− *1*, *1*, *1*], *vector*[*1*, − *1*, *1*], *vector*[*1*, *1*, *1*]},
convex hull {*vector*[− *1*, *1*, − *1*], *vector*[− *1*, *1*, *1*], *vector*[*1*, *1*, − *1*], *vector*[*1*, *1*, *1*]},
convex hull {*vector*[*1*, − *1*, − *1*], *vector*[*1*, − *1*, *1*], *vector*[*1*, *1*, − *1*], *vector*[*1*, *1*, *1*]}}
have *f1* ∈ *?faces f2* ∈ *?faces*
using *assms cube-facets*
by *auto*
moreover have ∀ *f*∈*?faces*.
convex hull {*vector*[− *1*, − *1*, − *1*], *vector*[− *1*, − *1*, *1*], *vector*[− *1*, *1*, − *1*], *vector*[− *1*, *1*, *1*]} *congruent f*
apply (*simp only*: *ball-simps*(*7*) *Set.ball-empty simp-thms*(*21*))
unfolding *conj-assoc*[*symmetric*]
by (*rule conjI*)+ *show-congruence*+
ultimately show *?thesis*
using *congruent-set* **by** *meson*
qed

lemma *cube-facet1-equiangular*:
defines *v1* ≡ *vector*[−*1*, −*1*, −*1*] :: *real^3*
defines *v2* ≡ *vector*[−*1*, −*1*, *1*] :: *real^3*
defines *v3* ≡ *vector*[−*1*, *1*, −*1*] :: *real^3*
defines *v4* ≡ *vector*[−*1*, *1*, *1*] :: *real^3*
shows *equiangular* (*convex hull* {*v1*, *v2*, *v3*, *v4*}) (*pi* / *2*)
unfolding *equiangular-def*
proof (*clarify*)
fix *e1 e2 v*
assume ∗: *e1 face-of convex hull* {*v1*, *v2*, *v3*, *v4*} *aff-dim e1* = *1*
e2 face-of convex hull {*v1*, *v2*, *v3*, *v4*} *aff-dim e2* = *1*
e1 ≠ *e2 v extreme-point-of e1 v extreme-point-of e2*
have (*v* = *v1* ∧ ((*e1* = *convex hull* {*v1*, *v2*} ∧ *e2* = *convex hull* {*v1*, *v3*}) ∨
(*e1* = *convex hull* {*v1*, *v3*} ∧ *e2* = *convex hull* {*v1*, *v2*}))) ∨
(*v* = *v2* ∧ ((*e1* = *convex hull* {*v2*, *v1*} ∧ *e2* = *convex hull* {*v2*, *v4*}) ∨
(*e1* = *convex hull* {*v2*, *v4*} ∧ *e2* = *convex hull* {*v2*, *v1*}))) ∨
(*v* = *v3* ∧ ((*e1* = *convex hull* {*v3*, *v1*} ∧ *e2* = *convex hull* {*v3*, *v4*}) ∨
(*e1* = *convex hull* {*v3*, *v4*} ∧ *e2* = *convex hull* {*v3*, *v1*}))) ∨
(*v* = *v4* ∧ ((*e1* = *convex hull* {*v4*, *v2*} ∧ *e2* = *convex hull* {*v4*, *v3*}) ∨
(*e1* = *convex hull* {*v4*, *v3*} ∧ *e2* = *convex hull* {*v4*, *v2*})))
using *conjI*[*OF* ∗(*1*,*2*)] *conjI*[*OF* ∗(*3*,*4*)] ∗(*5*−*7*) **unfolding** *assms cube-facet1-edges*
by (*smt* (*verit*, *best*) *extreme-point-of-convex-hull-2 insert-commute vector3-eq*)
moreover have *vangle* (*v2* − *v1*) (*v3* − *v1*) = *pi* / *2*
vangle (*v1* − *v2*) (*v4* − *v2*) = *pi* / *2*
vangle (*v1* − *v3*) (*v4* − *v3*) = *pi* / *2*
vangle (*v2* − *v4*) (*v3* − *v4*) = *pi* / *2*
unfolding *assms vector3-sub norm-eq-sqrt-inner vector3-dot vangle-def*
by (*simp-all add*: *vector3-eq-0*)

ultimately show (∃ *a b*. *e1* = *convex hull* {*v*, *a*} ∧ *e2* = *convex hull* {*v*, *b*} ∧
vangle (*a* − *v*) (*b* − *v*) = *pi* / *2*)
using *vangle-commute*
by *metis*
qed

lemma *cube-equiangular*:
assumes *f face-of std-cube* ∧ *aff-dim f* = *2*
shows *equiangular f* (*pi* / *2*)
proof −
define *f1* **where** *f1* = *convex hull* {*vector*[− *1*, − *1*, − *1*]::*real^3*, *vector*[− *1*, − *1*, *1*], *vector*[− *1*, *1*, − *1*], *vector*[− *1*, *1*, *1*]}
have *f1 face-of std-cube* ∧ *aff-dim f1* = *2*
using *f1-def cube-facets* **by** *blast*
then have *f1 congruent f*
using *assms cube-congruent-faces* **by** *blast*
then show *?thesis*
using *equiangular-congruent cube-facet1-equiangular f1-def*
by *blast*
qed

end

7 Octahedron

theory *Octahedron*
imports *Computation*
begin

7.1 Definition

definition *std-octahedron* :: (*real*, *3*) *vec set*
where *std-octahedron* ≡ *convex hull*
{*vector* [*1*, *0*, *0*], *vector* [−*1*, *0*, *0*],
vector [*0*, *0*, *1*], *vector* [*0*, *0*, −*1*],
vector [*0*, *1*, *0*], *vector* [*0*, −*1*, *0*]}

lemma *octahedron-fulldim*:
shows *aff-dim std-octahedron* = *3*
unfolding *std-octahedron-def*
apply (*rule polytope-3D*)
by (*simp add*: *vector3-sub vector3-cross vector3-dot vector3-eq-0*)+

lemma *octahedron-polyhedron*:
shows *polyhedron std-octahedron*
by (*simp add*: *std-octahedron-def polytope-convex-hull polytope-imp-polyhedron*)

7.2 Facets, edges, and vertices

lemma *octahedron-facets*:
 fixes f :: (*real*^*3*) *set*
 shows f *face-of std-octahedron* ∧ *aff-dim* $f = 2 \longleftrightarrow$
  f = *convex hull* {*vector*[− *1*, *0*, *0*], *vector*[*0*, − *1*, *0*], *vector*[*0*, *0*, − *1*]} ∨
  f = *convex hull* {*vector*[− *1*, *0*, *0*], *vector*[*0*, − *1*, *0*], *vector*[*0*, *0*, *1*]} ∨
  f = *convex hull* {*vector*[− *1*, *0*, *0*], *vector*[*0*, *1*, *0*], *vector*[*0*, *0*, − *1*]} ∨
  f = *convex hull* {*vector*[− *1*, *0*, *0*], *vector*[*0*, *1*, *0*], *vector*[*0*, *0*, *1*]} ∨
  f = *convex hull* {*vector*[*1*, *0*, *0*], *vector*[*0*, − *1*, *0*], *vector*[*0*, *0*, − *1*]} ∨
  f = *convex hull* {*vector*[*1*, *0*, *0*], *vector*[*0*, − *1*, *0*], *vector*[*0*, *0*, *1*]} ∨
  f = *convex hull* {*vector*[*1*, *0*, *0*], *vector*[*0*, *1*, *0*], *vector*[*0*, *0*, − *1*]} ∨
  f = *convex hull* {*vector*[*1*, *0*, *0*], *vector*[*0*, *1*, *0*], *vector*[*0*, *0*, *1*]}
 unfolding *std-octahedron-def*
 apply (*simp only*: *compute-faces-2 finite-insert finite.emptyI*)
 apply (*simp only*: *compute-faces-2-step-1 bex-empty simp-thms*(*31*))
 apply (*simp only*: *compute-faces-2-step-2 bex-empty simp-thms*(*31*))
 apply (*simp only*: *empty-iff ball-insert bex-empty bex-simps*(*5*) *ball-simps*(*5*) *simp-thms*(*21*,*31*))
 apply (*simp add*: *vector3-sub vector3-cross vector3-eq-0 vector3-dot Let-def*)
 apply (*simp only*: *insert-commute*)
 by *linarith*

lemma *octahedron-facet1-edges*:
 defines *v1* ≡ *vector*[−*1*, *0*, *0*] :: *real*^*3*
  and *v2* ≡ *vector*[*0*, −*1*, *0*] :: *real*^*3*
  and *v3* ≡ *vector*[*0*, *0*, −*1*] :: *real*^*3*
  and n ≡ *vector*[*1*, *1*, *1*] :: *real*^*3*
 shows (e *face-of* (*convex hull* {*v1*, *v2*, *v3*}) ∧ *aff-dim* $e = 1$) =
  (e = *convex hull* {*v1*, *v2*} ∨ e = *convex hull* {*v1*, *v3*} ∨ e = *convex hull* {*v2*, *v3*})
proof −
 have *n-nz*: $n \neq 0$
  unfolding *n-def* **by** (*simp add*: *vector3-eq-0*)
 have *plane*: $\bigwedge x.\ x \in \{v1, v2, v3\} \Longrightarrow n \cdot x = -1$
  by (*auto simp add*: *assms vector3-dot*)
 have *finite*: *finite* {*v1*, *v2*, *v3*}
  using *assms* **by** *blast*
 show *?thesis*
  by (*facet-edges plane*: *plane n-nz*: *n-nz defs*: *assms*)
qed

lemma *octahedron-facet2-edges*:
 defines *v1* ≡ *vector*[−*1*, *0*, *0*] :: *real*^*3*
  and *v2* ≡ *vector*[*0*, −*1*, *0*] :: *real*^*3*
  and *v3* ≡ *vector*[*0*, *0*, *1*] :: *real*^*3*
  and n ≡ *vector*[−*1*, −*1*, *1*] :: *real*^*3*
 shows (e *face-of* (*convex hull* {*v1*, *v2*, *v3*}) ∧ *aff-dim* $e = 1$) =
  (e = *convex hull* {*v1*, *v2*} ∨ e = *convex hull* {*v1*, *v3*} ∨ e = *convex hull* {*v2*, *v3*})

proof –
  have *n-nz*: $n \neq 0$
    unfolding *n-def* **by** (*simp add*: *vector3-eq-0*)
  have *plane*: $\bigwedge x.\ x \in \{v1,\ v2,\ v3\} \Longrightarrow n \cdot x = 1$
    by (*auto simp add*: *assms vector3-dot*)
  have *finite*: *finite* $\{v1,\ v2,\ v3\}$
    using *assms* **by** *blast*
  show *?thesis*
    by (*facet-edges plane*: *plane n-nz*: *n-nz defs*: *assms*)
qed

lemma *octahedron-facet3-edges*:
  defines $v1 \equiv vector[-1,\ 0,\ 0] :: real\hat{\ }3$
    and $v2 \equiv vector[0,\ 1,\ 0] :: real\hat{\ }3$
    and $v3 \equiv vector[0,\ 0,\ -1] :: real\hat{\ }3$
    and $n \equiv vector[-1,\ 1,\ -1] :: real\hat{\ }3$
  shows (e *face-of* (*convex hull* $\{v1,\ v2,\ v3\}$) $\wedge$ *aff-dim* $e = 1$) =
        (e = *convex hull* $\{v1,\ v2\}$ $\vee$ e = *convex hull* $\{v1,\ v3\}$ $\vee$ e = *convex hull*
$\{v2,\ v3\}$)
proof –
  have *n-nz*: $n \neq 0$
    unfolding *n-def* **by** (*simp add*: *vector3-eq-0*)
  have *plane*: $\bigwedge x.\ x \in \{v1,\ v2,\ v3\} \Longrightarrow n \cdot x = 1$
    by (*auto simp add*: *assms vector3-dot*)
  have *finite*: *finite* $\{v1,\ v2,\ v3\}$
    using *assms* **by** *blast*
  show *?thesis*
    by (*facet-edges plane*: *plane n-nz*: *n-nz defs*: *assms*)
qed

lemma *octahedron-facet4-edges*:
  defines $v1 \equiv vector[-1,\ 0,\ 0] :: real\hat{\ }3$
    and $v2 \equiv vector[0,\ 1,\ 0] :: real\hat{\ }3$
    and $v3 \equiv vector[0,\ 0,\ 1] :: real\hat{\ }3$
    and $n \equiv vector[1,\ -1,\ -1] :: real\hat{\ }3$
  shows (e *face-of* (*convex hull* $\{v1,\ v2,\ v3\}$) $\wedge$ *aff-dim* $e = 1$) =
        (e = *convex hull* $\{v1,\ v2\}$ $\vee$ e = *convex hull* $\{v1,\ v3\}$ $\vee$ e = *convex hull*
$\{v2,\ v3\}$)
proof –
  have *n-nz*: $n \neq 0$
    unfolding *n-def* **by** (*simp add*: *vector3-eq-0*)
  have *plane*: $\bigwedge x.\ x \in \{v1,\ v2,\ v3\} \Longrightarrow n \cdot x = -1$
    by (*auto simp add*: *assms vector3-dot*)
  have *finite*: *finite* $\{v1,\ v2,\ v3\}$
    using *assms* **by** *blast*
  show *?thesis*
    by (*facet-edges plane*: *plane n-nz*: *n-nz defs*: *assms*)
qed

lemma *octahedron-facet5-edges*:
  defines $v1 \equiv vector[1, 0, 0] :: real\hat{}3$
    and $v2 \equiv vector[0, -1, 0] :: real\hat{}3$
    and $v3 \equiv vector[0, 0, -1] :: real\hat{}3$
    and $n \equiv vector[1, -1, -1] :: real\hat{}3$
  shows $(e$ *face-of* $(convex\ hull\ \{v1, v2, v3\}) \wedge$ *aff-dim* $e = 1) =$
        $(e = convex\ hull\ \{v1, v2\} \vee e = convex\ hull\ \{v1, v3\} \vee e = convex\ hull$
$\{v2, v3\})$
proof –
  have *n-nz*: $n \neq 0$
    unfolding *n-def* **by** (*simp add*: *vector3-eq-0*)
  have *plane*: $\bigwedge x.\ x \in \{v1, v2, v3\} \Longrightarrow n \cdot x = 1$
    by (*auto simp add*: *assms vector3-dot*)
  have *finite*: *finite* $\{v1, v2, v3\}$
    using *assms* **by** *blast*
  show *?thesis*
    by (*facet-edges plane*: *plane n-nz*: *n-nz defs*: *assms*)
qed

lemma *octahedron-facet6-edges*:
  defines $v1 \equiv vector[1, 0, 0] :: real\hat{}3$
    and $v2 \equiv vector[0, -1, 0] :: real\hat{}3$
    and $v3 \equiv vector[0, 0, 1] :: real\hat{}3$
    and $n \equiv vector[-1, 1, -1] :: real\hat{}3$
  shows $(e$ *face-of* $(convex\ hull\ \{v1, v2, v3\}) \wedge$ *aff-dim* $e = 1) =$
        $(e = convex\ hull\ \{v1, v2\} \vee e = convex\ hull\ \{v1, v3\} \vee e = convex\ hull$
$\{v2, v3\})$
proof –
  have *n-nz*: $n \neq 0$
    unfolding *n-def* **by** (*simp add*: *vector3-eq-0*)
  have *plane*: $\bigwedge x.\ x \in \{v1, v2, v3\} \Longrightarrow n \cdot x = -1$
    by (*auto simp add*: *assms vector3-dot*)
  have *finite*: *finite* $\{v1, v2, v3\}$
    using *assms* **by** *blast*
  show *?thesis*
    by (*facet-edges plane*: *plane n-nz*: *n-nz defs*: *assms*)
qed

lemma *octahedron-facet7-edges*:
  defines $v1 \equiv vector[1, 0, 0] :: real\hat{}3$
    and $v2 \equiv vector[0, 1, 0] :: real\hat{}3$
    and $v3 \equiv vector[0, 0, -1] :: real\hat{}3$
    and $n \equiv vector[-1, -1, 1] :: real\hat{}3$
  shows $(e$ *face-of* $(convex\ hull\ \{v1, v2, v3\}) \wedge$ *aff-dim* $e = 1) =$
        $(e = convex\ hull\ \{v1, v2\} \vee e = convex\ hull\ \{v1, v3\} \vee e = convex\ hull$
$\{v2, v3\})$
proof –
  have *n-nz*: $n \neq 0$
    unfolding *n-def* **by** (*simp add*: *vector3-eq-0*)

have *plane*: $\bigwedge x.\ x \in \{v1,\ v2,\ v3\} \Longrightarrow n \cdot x = -1$
by (*auto simp add*: *assms vector3-dot*)
have *finite*: *finite* $\{v1,\ v2,\ v3\}$
using *assms* **by** *blast*
show *?thesis*
by (*facet-edges plane*: *plane n-nz*: *n-nz defs*: *assms*)
qed

lemma *octahedron-facet8-edges*:
defines $v1 \equiv vector[1,\ 0,\ 0] :: real\hat{}3$
and $v2 \equiv vector[0,\ 1,\ 0] :: real\hat{}3$
and $v3 \equiv vector[0,\ 0,\ 1] :: real\hat{}3$
and $n \equiv vector[1,\ 1,\ 1] :: real\hat{}3$
shows $(e\ face\text{-}of\ (convex\ hull\ \{v1,\ v2,\ v3\}) \wedge aff\text{-}dim\ e = 1) =$
$(e = convex\ hull\ \{v1,\ v2\} \vee e = convex\ hull\ \{v1,\ v3\} \vee e = convex\ hull$
$\{v2,\ v3\})$
proof −
have *n-nz*: $n \neq 0$
unfolding *n-def* **by** (*simp add*: *vector3-eq-0*)
have *plane*: $\bigwedge x.\ x \in \{v1,\ v2,\ v3\} \Longrightarrow n \cdot x = 1$
by (*auto simp add*: *assms vector3-dot*)
have *finite*: *finite* $\{v1,\ v2,\ v3\}$
using *assms* **by** *blast*
show *?thesis*
by (*facet-edges plane*: *plane n-nz*: *n-nz defs*: *assms*)
qed

lemmas *octahedron-facet-edges* =
octahedron-facet1-edges octahedron-facet2-edges
octahedron-facet3-edges octahedron-facet4-edges octahedron-facet5-edges
octahedron-facet6-edges octahedron-facet7-edges octahedron-facet8-edges

lemma *octahedron-edges-per-face*:
assumes *f face-of std-octahedron*
and $aff\text{-}dim\ f = 2$
shows $card\ \{e.\ e\ face\text{-}of\ std\text{-}octahedron \wedge aff\text{-}dim\ e = 1 \wedge e \subseteq f\} = 3$
using *conjI*[*OF assms*] **unfolding** *octahedron-facets face-edge-set-eq*[*OF assms*(*1*)]
apply (*elim disjE forw-subst*)
unfolding *octahedron-facet-edges set-cases card-3-iff2*
segment-convex-hull[*symmetric*] *closed-segment-eq*
by (*simp-all add*: *vector3-eq doubleton-eq-iff*)

lemma *octahedron-edges*:
$e\ face\text{-}of\ std\text{-}octahedron \wedge aff\text{-}dim\ e = 1 \longleftrightarrow$
$e = convex\ hull\ \{vector[-\ 1,\ 0,\ 0],\ vector[0,\ -\ 1,\ 0]\} \vee$
$e = convex\ hull\ \{vector[-\ 1,\ 0,\ 0],\ vector[0,\ 1,\ 0]\} \vee$
$e = convex\ hull\ \{vector[-\ 1,\ 0,\ 0],\ vector[0,\ 0,\ -\ 1]\} \vee$
$e = convex\ hull\ \{vector[-\ 1,\ 0,\ 0],\ vector[0,\ 0,\ 1]\} \vee$
$e = convex\ hull\ \{vector[1,\ 0,\ 0],\ vector[0,\ -\ 1,\ 0]\} \vee$

e = *convex hull* {*vector*[*1*, *0*, *0*], *vector*[*0*, *1*, *0*]} ∨
e = *convex hull* {*vector*[*1*, *0*, *0*], *vector*[*0*, *0*, − *1*]} ∨
e = *convex hull* {*vector*[*1*, *0*, *0*], *vector*[*0*, *0*, *1*]} ∨
e = *convex hull* {*vector*[*0*, − *1*, *0*], *vector*[*0*, *0*, − *1*]} ∨
e = *convex hull* {*vector*[*0*, − *1*, *0*], *vector*[*0*, *0*, *1*]} ∨
e = *convex hull* {*vector*[*0*, *1*, *0*], *vector*[*0*, *0*, − *1*]} ∨
e = *convex hull* {*vector*[*0*, *1*, *0*], *vector*[*0*, *0*, *1*]}
(**is** *?lhs* ⟷ *?rhs*)
proof
assume *?lhs*
then obtain *f* **where** *f-facts*: *f face-of std-octahedron* ∧ *aff-dim f* = *2 e face-of f* ∧ *aff-dim e* = *1*
using *octahedron-polyhedron octahedron-fulldim edge-belongs-to-face*
by *blast*
show *?rhs*
using *f-facts* **unfolding** *octahedron-facets*
apply (*elim disjE*) **apply** (*all* ‹*hypsubst-thin*›)
unfolding *octahedron-facet-edges*
by *linarith*+
next
assume *?rhs*
then show *?lhs*
using *octahedron-facets octahedron-facet-edges*
by (*smt* (*verit*, *ccfv-threshold*) *face-of-trans insertI1 insert-commute mem-Collect-eq*)
qed

lemma *octahedron-vertices*:
shows (*v face-of std-octahedron* ∧ *aff-dim v* = *0*) =
(*v* = {*vector* [*1*, *0*, *0*]} ∨
v = {*vector* [−*1*, *0*, *0*]} ∨
v = {*vector* [*0*, *0*, *1*]} ∨
v = {*vector* [*0*, *0*, −*1*]} ∨
v = {*vector* [*0*, *1*, *0*]} ∨
v = {*vector* [*0*, −*1*, *0*]})
(**is** *?lhs* = *?rhs*)
proof
assume ∗: *?lhs*
then obtain *e* **where** *e-facts*: *e face-of std-octahedron* ∧ *aff-dim e* = *1 v face-of e*
using *octahedron-polyhedron octahedron-fulldim vertex-belongs-to-edge*
by *blast*
show *?rhs*
using *e-facts*(*1*)
unfolding *octahedron-edges*
apply (*elim disjE*)
using *vertices-of-edge*[*OF* - ‹*v face-of e*›] ∗
by *metis*+
next
assume *?rhs*

then show *?lhs*
apply (*elim disjE*)
by (*meson aff-dim-sing extreme-point-of-convex-hull-2 face-of-singleton face-of-trans octahedron-edges*)+
qed

lemma *octahedron-edges-per-vertex*:
assumes *v face-of std-octahedron*
and *aff-dim v = 0*
shows *card* {*e. e face-of std-octahedron* ∧ *aff-dim e = 1* ∧ *v* ⊆ *e*} = *4*
unfolding *vertex-edge-set-eq*[*OF assms*(*1*)]
using *conjI*[*OF assms*]
unfolding *octahedron-vertices conj-assoc*[*symmetric*] *octahedron-edges*
by (*elim disjE forw-subst*) *edges-per-vertex*+

7.3 Regularity

lemma *octahedron-congruent-edges*:
assumes *e1 face-of std-octahedron* ∧ *aff-dim e1 = 1*
and *e2 face-of std-octahedron* ∧ *aff-dim e2 = 1*
shows *e1 congruent e2*
proof −
let *?edges* =
{*convex hull* {(*vector*[− *1*, *0*, *0*]::*real^3*), *vector*[*0*, − *1*, *0*]},
convex hull {*vector*[− *1*, *0*, *0*], *vector*[*0*, *1*, *0*]},
convex hull {*vector*[− *1*, *0*, *0*], *vector*[*0*, *0*, − *1*]},
convex hull {*vector*[− *1*, *0*, *0*], *vector*[*0*, *0*, *1*]},
convex hull {*vector*[*1*, *0*, *0*], *vector*[*0*, − *1*, *0*]},
convex hull {*vector*[*1*, *0*, *0*], *vector*[*0*, *1*, *0*]},
convex hull {*vector*[*1*, *0*, *0*], *vector*[*0*, *0*, − *1*]},
convex hull {*vector*[*1*, *0*, *0*], *vector*[*0*, *0*, *1*]},
convex hull {*vector*[*0*, − *1*, *0*], *vector*[*0*, *0*, − *1*]},
convex hull {*vector*[*0*, − *1*, *0*], *vector*[*0*, *0*, *1*]},
convex hull {*vector*[*0*, *1*, *0*], *vector*[*0*, *0*, − *1*]},
convex hull {*vector*[*0*, *1*, *0*], *vector*[*0*, *0*, *1*]}}
have *e1* ∈ *?edges e2* ∈ *?edges*
using *assms* **unfolding** *octahedron-edges*
by *auto*
moreover have ∀ *e*∈*?edges*. *convex hull* {*vector*[− *1*, *0*, *0*], *vector*[*0*, − *1*, *0*]} *congruent e*
apply (*simp only*: *ball-simps*(*7*) *Set.ball-empty simp-thms*(*21*))
unfolding *conj-assoc*[*symmetric*] *segment-convex-hull*[*symmetric*]
apply (*rule conjI*)+
apply (*all* ‹*rule congruent-segments*›)
unfolding *dist-norm vector3-sub*
apply (*simp-all add*: *norm-eq*)
unfolding *vector3-dot*
by *linarith*+
ultimately show *?thesis*

```
    using congruent-set by meson
qed

lemma octahedron-congruent-faces:
  assumes f1 face-of std-octahedron ∧ aff-dim f1 = 2
      and f2 face-of std-octahedron ∧ aff-dim f2 = 2
    shows f1 congruent f2
proof −
  let ?faces =
    {convex hull {(vector[− 1, 0, 0]::real^3), vector[0, − 1, 0], vector[0, 0, − 1]},
      convex hull {vector[− 1, 0, 0], vector[0, − 1, 0], vector[0, 0, 1]},
      convex hull {vector[− 1, 0, 0], vector[0, 1, 0], vector[0, 0, − 1]},
      convex hull {vector[− 1, 0, 0], vector[0, 1, 0], vector[0, 0, 1]},
      convex hull {vector[1, 0, 0], vector[0, − 1, 0], vector[0, 0, − 1]},
      convex hull {vector[1, 0, 0], vector[0, − 1, 0], vector[0, 0, 1]},
      convex hull {vector[1, 0, 0], vector[0, 1, 0], vector[0, 0, − 1]},
      convex hull {vector[1, 0, 0], vector[0, 1, 0], vector[0, 0, 1]}}
  have f1 ∈ ?faces f2 ∈ ?faces
    using assms octahedron-facets
    by auto
  moreover have ∀f∈?faces.
          convex hull {vector[− 1, 0, 0], vector[0, − 1, 0], vector[0, 0, − 1]}
congruent f
    apply (simp only: ball-simps(7) Set.ball-empty simp-thms(21))
    unfolding conj-assoc[symmetric]
    by (rule conjI)+ show-congruence+
  ultimately show ?thesis
    using congruent-set by meson
qed

lemma octahedron-facet1-equiangular:
  defines v1 ≡ vector[−1, 0, 0] :: real^3
  defines v2 ≡ vector[0, −1, 0] :: real^3
  defines v3 ≡ vector[0, 0, −1] :: real^3
  shows equiangular (convex hull {v1, v2, v3}) (pi / 3)
  unfolding equiangular-def
proof (clarify)
  fix e1 e2 v
  assume ∗: e1 face-of convex hull {v1, v2, v3} aff-dim e1 = 1
          e2 face-of convex hull {v1, v2, v3} aff-dim e2 = 1
          e1 ≠ e2 v extreme-point-of e1 v extreme-point-of e2
  have (v = v1 ∧ ((e1 = convex hull {v1, v2} ∧ e2 = convex hull {v1, v3}) ∨
               (e1 = convex hull {v1, v3} ∧ e2 = convex hull {v1, v2}))) ∨
       (v = v2 ∧ ((e1 = convex hull {v2, v1} ∧ e2 = convex hull {v2, v3}) ∨
               (e1 = convex hull {v2, v3} ∧ e2 = convex hull {v2, v1}))) ∨
       (v = v3 ∧ ((e1 = convex hull {v3, v1} ∧ e2 = convex hull {v3, v2}) ∨
               (e1 = convex hull {v3, v2} ∧ e2 = convex hull {v3, v1})))
    using conjI[OF ∗(1,2)] conjI[OF ∗(3,4)] ∗(5−7) unfolding assms octahe-
dron-facet1-edges
```

by (*smt* (*verit*, *best*) *extreme-point-of-convex-hull-2 insert-commute vector3-eq*)
moreover have *vangle* (*v2* − *v1*) (*v3* − *v1*) = *pi* / *3*
vangle (*v1* − *v2*) (*v3* − *v2*) = *pi* / *3*
vangle (*v1* − *v3*) (*v2* − *v3*) = *pi* / *3*
unfolding *assms vector3-sub norm-eq-sqrt-inner vector3-dot vangle-def*
by (*simp-all add*: *vector3-eq-0*)
ultimately show (∃ *a b*. *e1* = *convex hull* {*v*, *a*} ∧ *e2* = *convex hull* {*v*, *b*} ∧
vangle (*a* − *v*) (*b* − *v*) = *pi* / *3*)
using *vangle-commute*
by *metis*
qed

lemma *octahedron-equiangular*:
assumes *f face-of std-octahedron* ∧ *aff-dim f* = *2*
shows *equiangular f* (*pi* / *3*)
proof −
define *f1* **where** *f1* = *convex hull* {*vector*[− *1*, *0*, *0*]::*real^3*, *vector*[*0*, − *1*, *0*],
vector[*0*, *0*, − *1*]}
have *f1 face-of std-octahedron* ∧ *aff-dim f1* = *2*
using *f1-def octahedron-facets* **by** *blast*
then have *f1 congruent f*
using *assms octahedron-congruent-faces* **by** *blast*
then show *?thesis*
using *equiangular-congruent octahedron-facet1-equiangular f1-def*
by *blast*
qed

end

8 Dodecahedron

theory *Dodecahedron*
imports
Computation
HOL−Library.Quadratic-Discriminant
begin

8.1 Definition

definition *std-dodecahedron* :: (*real*, *3*) *vec set*
where *std-dodecahedron* ≡ *convex hull*
{*vector* [*1*, *1*, *1*], *vector* [*1*, *1*, −*1*],
vector [*1*, −*1*, *1*], *vector* [*1*, −*1*, −*1*],
vector [−*1*, *1*, *1*], *vector* [−*1*, *1*, −*1*],
vector [−*1*, −*1*, *1*], *vector* [−*1*, −*1*, −*1*],
vector [*0*, *inverse* φ, φ], *vector*[*0*, *inverse* φ, −φ],
vector [*0*, −*inverse* φ, φ], *vector* [*0*, −*inverse* φ, −φ],
vector [*inverse* φ, φ, *0*], *vector* [*inverse* φ, −φ, *0*],
vector [−*inverse* φ, φ, *0*], *vector* [−*inverse* φ, −φ, *0*],

vector [φ, *0*, *inverse* φ], *vector* [$-\varphi$, *0*, *inverse* φ],
vector [φ, *0*, $-$*inverse* φ], *vector* [$-\varphi$, *0*, $-$*inverse* φ]}

lemma *std-dodecahedron-explicit*:
shows *std-dodecahedron* = *convex hull*
{ *vector*[*1*, *1*, *1*],
vector[*1*, *1*, $-$*1*],
vector[*1*, $-$*1*, *1*],
vector[*1*, $-$*1*, $-$*1*],
vector[$-$*1*, *1*, *1*],
vector[$-$*1*, *1*, $-$*1*],
vector[$-$*1*, $-$*1*, *1*],
vector[$-$*1*, $-$*1*, $-$*1*],
vector[*0*, $-$*1* / *2* + *1* / *2* ∗ *sqrt*(*5*), *1* / *2* + *1* / *2* ∗ *sqrt*(*5*)],
vector[*0*, $-$*1* / *2* + *1* / *2* ∗ *sqrt*(*5*), $-$*1* / *2* + $-$*1* / *2* ∗ *sqrt*(*5*)],
vector[*0*, *1* / *2* + $-$*1* / *2* ∗ *sqrt*(*5*), *1* / *2* + *1* / *2* ∗ *sqrt*(*5*)],
vector[*0*, *1* / *2* + $-$*1* / *2* ∗ *sqrt*(*5*), $-$*1* / *2* + $-$*1* / *2* ∗ *sqrt*(*5*)],
vector[$-$*1* / *2* + *1* / *2* ∗ *sqrt*(*5*), *1* / *2* + *1* / *2* ∗ *sqrt*(*5*), *0*],
vector[$-$*1* / *2* + *1* / *2* ∗ *sqrt*(*5*), $-$*1* / *2* + $-$*1* / *2* ∗ *sqrt*(*5*), *0*],
vector[*1* / *2* + $-$*1* / *2* ∗ *sqrt*(*5*), *1* / *2* + *1* / *2* ∗ *sqrt*(*5*), *0*],
vector[*1* / *2* + $-$*1* / *2* ∗ *sqrt*(*5*), $-$*1* / *2* + $-$*1* / *2* ∗ *sqrt*(*5*), *0*],
vector[*1* / *2* + *1* / *2* ∗ *sqrt*(*5*), *0*, $-$*1* / *2* + *1* / *2* ∗ *sqrt*(*5*)],
vector[$-$*1* / *2* + $-$*1* / *2* ∗ *sqrt*(*5*), *0*, $-$*1* / *2* + *1* / *2* ∗ *sqrt*(*5*)],
vector[*1* / *2* + *1* / *2* ∗ *sqrt*(*5*), *0*, *1* / *2* + $-$*1* / *2* ∗ *sqrt*(*5*)],
vector[$-$*1* / *2* + $-$*1* / *2* ∗ *sqrt*(*5*), *0*, *1* / *2* + $-$*1* / *2* ∗ *sqrt*(*5*)]}
proof $-$
have *inverse* φ = *2* / (*1* + *sqrt 5*) **by** *simp*
also have ... = (*2* ∗ (*1* $-$ *sqrt 5*)) / ((*1* + *sqrt 5*) ∗ (*1* $-$ *sqrt 5*))
by (*metis mult.commute nonzero-mult-divide-mult-cancel-right2 numeral-eq-one-iff real-sqrt-eq-1-iff right-minus-eq*
verit-eq-simplify(*12*))
also have ... = (*2* $-$ *2* ∗ *sqrt 5*) / ($-$*4*)
using *square-diff-square-factored*[*of 1 sqrt 5*] **by** *auto*
finally have *inverse* φ = $-$*1*/*2* + *1*/*2* ∗ *sqrt 5*
by *argo*
moreover have *golden*: φ = *1*/*2* + *1*/*2* ∗ *sqrt 5*
by *auto*
ultimately have *golden-inverse*: *inverse* (*1*/*2* + *1*/*2* ∗ *sqrt 5*) = $-$*1*/*2* + *1*/*2* ∗ *sqrt 5*
by *auto*
show *?thesis*
unfolding *std-dodecahedron-def golden golden-inverse*
by *simp*
qed

lemma *std-dodecahedron-eq*:
shows *std-dodecahedron* = *convex hull*
{ *vector*[*rat5 1 0*, *rat5 1 0*, *rat5 1 0*],
vector[*rat5 1 0*, *rat5 1 0*, *rat5* ($-$*1*) *0*],

vector[*rat5 1 0*, *rat5* (−*1*) *0*, *rat5 1 0*],
vector[*rat5 1 0*, *rat5* (−*1*) *0*, *rat5* (−*1*) *0*],
vector[*rat5* (−*1*) *0*, *rat5 1 0*, *rat5 1 0*],
vector[*rat5* (−*1*) *0*, *rat5 1 0*, *rat5* (−*1*) *0*],
vector[*rat5* (−*1*) *0*, *rat5* (−*1*) *0*, *rat5 1 0*],
vector[*rat5* (−*1*) *0*, *rat5* (−*1*) *0*, *rat5* (−*1*) *0*],
vector[*rat5 0 0*, *rat5* (−*1/2*) (*1/2*), *rat5* (*1/2*) (*1/2*)],
vector[*rat5 0 0*, *rat5* (−*1/2*) (*1/2*), *rat5* (−*1/2*) (−*1/2*)],
vector[*rat5 0 0*, *rat5* (*1/2*) (−*1/2*), *rat5* (*1/2*) (*1/2*)],
vector[*rat5 0 0*, *rat5* (*1/2*) (−*1/2*), *rat5* (−*1/2*) (−*1/2*)],
vector[*rat5* (−*1/2*) (*1/2*), *rat5* (*1/2*) (*1/2*), *rat5 0 0*],
vector[*rat5* (−*1/2*) (*1/2*), *rat5* (−*1/2*) (−*1/2*), *rat5 0 0*],
vector[*rat5* (*1/2*) (−*1/2*), *rat5* (*1/2*) (*1/2*), *rat5 0 0*],
vector[*rat5* (*1/2*) (−*1/2*), *rat5* (−*1/2*) (−*1/2*), *rat5 0 0*],
vector[*rat5* (*1/2*) (*1/2*), *rat5 0 0*, *rat5* (−*1/2*) (*1/2*)],
vector[*rat5* (−*1/2*) (−*1/2*), *rat5 0 0*, *rat5* (−*1/2*) (*1/2*)],
vector[*rat5* (*1/2*) (*1/2*), *rat5 0 0*, *rat5* (*1/2*) (−*1/2*)],
vector[*rat5* (−*1/2*) (−*1/2*), *rat5 0 0*, *rat5* (*1/2*) (−*1/2*)]}
unfolding *std-dodecahedron-explicit rat5-def*
by (*simp add*: *of-rat-divide of-rat-minus*)

lemma *dodecahedron-fulldim*:
shows *aff-dim std-dodecahedron* = *3*
unfolding *std-dodecahedron-def*
apply (*rule polytope-3D*)
by (*simp add*: *vector3-sub vector3-cross vector3-dot vector3-eq-0*)+

lemma *dodecahedron-polyhedron*:
shows *polyhedron std-dodecahedron*
by (*simp add*: *std-dodecahedron-def polytope-convex-hull polytope-imp-polyhedron*)

8.2 Facets, edges, and vertices

lemma *dodecahedron-facets*:
fixes *f* :: (*real^3*) *set*
shows *f face-of std-dodecahedron* ∧ *aff-dim f* = *2* ⟷
f = *convex hull* {*vector*[*rat5* (−*1/2*) (−*1/2*), *rat5 0 0*, *rat5* (−*1/2*) (*1/2*)], *vector*[*rat5* (−*1/2*) (−*1/2*), *rat5 0 0*, *rat5* (*1/2*) (−*1/2*)], *vector*[*rat5* (*1/2*) (−*1/2*), *rat5* (−*1/2*) (−*1/2*), *rat5 0 0*], *vector*[*rat5* (−*1*) *0*, *rat5* (−*1*) *0*, *rat5* (−*1*) *0*], *vector*[*rat5* (−*1*) *0*, *rat5* (−*1*) *0*, *rat5 1 0*]} ∨
f = *convex hull* {*vector*[*rat5* (−*1/2*) (−*1/2*), *rat5 0 0*, *rat5* (−*1/2*) (*1/2*)], *vector*[*rat5* (−*1/2*) (−*1/2*), *rat5 0 0*, *rat5* (*1/2*) (−*1/2*)], *vector*[*rat5* (*1/2*) (−*1/2*), *rat5* (*1/2*) (*1/2*), *rat5 0 0*], *vector*[*rat5* (−*1*) *0*, *rat5 1 0*, *rat5* (−*1*) *0*], *vector*[*rat5* (−*1*) *0*, *rat5 1 0*, *rat5 1 0*]} ∨
f = *convex hull* {*vector*[*rat5* (−*1/2*) (−*1/2*), *rat5 0 0*, *rat5* (−*1/2*) (*1/2*)], *vector*[*rat5* (−*1*) *0*, *rat5* (−*1*) *0*, *rat5 1 0*], *vector*[*rat5* (−*1*) *0*, *rat5 1 0*, *rat5 1 0*], *vector*[*rat5 0 0*, *rat5* (−*1/2*) (*1/2*), *rat5* (*1/2*) (*1/2*)], *vector*[*rat5 0 0*, *rat5* (*1/2*) (−*1/2*), *rat5* (*1/2*) (*1/2*)]} ∨
f = *convex hull* {*vector*[*rat5* (−*1/2*) (−*1/2*), *rat5 0 0*, *rat5* (*1/2*) (−*1/2*)],

vector[*rat5* (−*1*) *0*, *rat5* (−*1*) *0*, *rat5* (−*1*) *0*], *vector*[*rat5* (−*1*) *0*, *rat5 1 0*, *rat5* (−*1*) *0*], *vector*[*rat5 0 0*, *rat5* (−*1*/*2*) (*1*/*2*), *rat5* (−*1*/*2*) (−*1*/*2*)], *vector*[*rat5 0 0*, *rat5* (*1*/*2*) (−*1*/*2*), *rat5* (−*1*/*2*) (−*1*/*2*)]} ∨

f = *convex hull* {*vector*[*rat5* (−*1*/*2*) (*1*/*2*), *rat5* (−*1*/*2*) (−*1*/*2*), *rat5 0 0*], *vector*[*rat5* (*1*/*2*) (−*1*/*2*), *rat5* (−*1*/*2*) (−*1*/*2*), *rat5 0 0*], *vector*[*rat5* (−*1*) *0*, *rat5* (−*1*) *0*, *rat5* (−*1*) *0*], *vector*[*rat5 1 0*, *rat5* (−*1*) *0*, *rat5* (−*1*) *0*], *vector*[*rat5 0 0*, *rat5* (*1*/*2*) (−*1*/*2*), *rat5* (−*1*/*2*) (−*1*/*2*)]} ∨

f = *convex hull* {*vector*[*rat5* (−*1*/*2*) (*1*/*2*), *rat5* (−*1*/*2*) (−*1*/*2*), *rat5 0 0*], *vector*[*rat5* (*1*/*2*) (−*1*/*2*), *rat5* (−*1*/*2*) (−*1*/*2*), *rat5 0 0*], *vector*[*rat5* (−*1*) *0*, *rat5* (−*1*) *0*, *rat5 1 0*], *vector*[*rat5 1 0*, *rat5* (−*1*) *0*, *rat5 1 0*], *vector*[*rat5 0 0*, *rat5* (*1*/*2*) (−*1*/*2*), *rat5* (*1*/*2*) (*1*/*2*)]} ∨

f = *convex hull* {*vector*[*rat5* (−*1*/*2*) (*1*/*2*), *rat5* (−*1*/*2*) (−*1*/*2*), *rat5 0 0*], *vector*[*rat5* (*1*/*2*) (*1*/*2*), *rat5 0 0*, *rat5* (−*1*/*2*) (*1*/*2*)], *vector*[*rat5* (*1*/*2*) (*1*/*2*), *rat5 0 0*, *rat5* (*1*/*2*) (−*1*/*2*)], *vector*[*rat5 1 0*, *rat5* (−*1*) *0*, *rat5* (−*1*) *0*], *vector*[*rat5 1 0*, *rat5* (−*1*) *0*, *rat5 1 0*]} ∨

f = *convex hull* {*vector*[*rat5* (−*1*/*2*) (*1*/*2*), *rat5* (*1*/*2*) (*1*/*2*), *rat5 0 0*], *vector*[*rat5* (*1*/*2*) (−*1*/*2*), *rat5* (*1*/*2*) (*1*/*2*), *rat5 0 0*], *vector*[*rat5* (−*1*) *0*, *rat5 1 0*, *rat5* (−*1*) *0*], *vector*[*rat5 1 0*, *rat5 1 0*, *rat5* (−*1*) *0*], *vector*[*rat5 0 0*, *rat5* (−*1*/*2*) (*1*/*2*), *rat5* (−*1*/*2*) (−*1*/*2*)]} ∨

f = *convex hull* {*vector*[*rat5* (−*1*/*2*) (*1*/*2*), *rat5* (*1*/*2*) (*1*/*2*), *rat5 0 0*], *vector*[*rat5* (*1*/*2*) (−*1*/*2*), *rat5* (*1*/*2*) (*1*/*2*), *rat5 0 0*], *vector*[*rat5* (−*1*) *0*, *rat5 1 0*, *rat5 1 0*], *vector*[*rat5 1 0*, *rat5 1 0*, *rat5 1 0*], *vector*[*rat5 0 0*, *rat5* (−*1*/*2*) (*1*/*2*), *rat5* (*1*/*2*) (*1*/*2*)]} ∨

f = *convex hull* {*vector*[*rat5* (−*1*/*2*) (*1*/*2*), *rat5* (*1*/*2*) (*1*/*2*), *rat5 0 0*], *vector*[*rat5* (*1*/*2*) (*1*/*2*), *rat5 0 0*, *rat5* (−*1*/*2*) (*1*/*2*)], *vector*[*rat5* (*1*/*2*) (*1*/*2*), *rat5 0 0*, *rat5* (*1*/*2*) (−*1*/*2*)], *vector*[*rat5 1 0*, *rat5 1 0*, *rat5* (−*1*) *0*], *vector*[*rat5 1 0*, *rat5 1 0*, *rat5 1 0*]} ∨

f = *convex hull* {*vector*[*rat5* (*1*/*2*) (*1*/*2*), *rat5 0 0*, *rat5* (−*1*/*2*) (*1*/*2*)], *vector*[*rat5 1 0*, *rat5* (−*1*) *0*, *rat5 1 0*], *vector*[*rat5 1 0*, *rat5 1 0*, *rat5 1 0*], *vector*[*rat5 0 0*, *rat5* (−*1*/*2*) (*1*/*2*), *rat5* (*1*/*2*) (*1*/*2*)], *vector*[*rat5 0 0*, *rat5* (*1*/*2*) (−*1*/*2*), *rat5* (*1*/*2*) (*1*/*2*)]} ∨

f = *convex hull* {*vector*[*rat5* (*1*/*2*) (*1*/*2*), *rat5 0 0*, *rat5* (*1*/*2*) (−*1*/*2*)], *vector*[*rat5 1 0*, *rat5* (−*1*) *0*, *rat5* (−*1*) *0*], *vector*[*rat5 1 0*, *rat5 1 0*, *rat5* (−*1*) *0*], *vector*[*rat5 0 0*, *rat5* (−*1*/*2*) (*1*/*2*), *rat5* (−*1*/*2*) (−*1*/*2*)], *vector*[*rat5 0 0*, *rat5* (*1*/*2*) (−*1*/*2*), *rat5* (−*1*/*2*) (−*1*/*2*)]}

unfolding *std-dodecahedron-eq*

apply (*simp only*: *compute-faces-2 finite-insert finite.emptyI*)

apply (*simp only*: *compute-faces-2-step-1 bex-empty simp-thms*(*31*))

apply (*simp only*: *compute-faces-2-step-2 bex-empty simp-thms*(*31*))

apply (*simp only*: *empty-iff ball-insert bex-empty bex-simps*(*5*) *ball-simps*(*5*) *simp-thms*(*21*,*31*))

apply (*simp only*: *vector3-sub vector3-cross vector3-eq-0 rat5-add rat5-sub rat5-mul rat5-eq*)

apply (*simp add*: *vector3-eq-0 vector3-dot rat5-add rat5-sub rat5-mul rat5-le rat5-eq rat5-eq-0 Let-def*)

— very slow, ca 600s

apply (*simp only*: *insert-commute*)

by *linarith*

lemma *dodecahedron-facet1-edges*:
defines *v1* ≡ *vector*[*rat5* (−*1*/*2*) (−*1*/*2*), *rat5 0 0*, *rat5* (−*1*/*2*) (*1*/*2*)] :: *real^3*
and *v2* ≡ *vector*[*rat5* (−*1*/*2*) (−*1*/*2*), *rat5 0 0*, *rat5* (*1*/*2*) (−*1*/*2*)] :: *real^3*
and *v3* ≡ *vector*[*rat5* (*1*/*2*) (−*1*/*2*), *rat5* (−*1*/*2*) (−*1*/*2*), *rat5 0 0*] :: *real^3*
and *v4* ≡ *vector*[*rat5* (−*1*) *0*, *rat5* (−*1*) *0*, *rat5* (−*1*) *0*] :: *real^3*
and *v5* ≡ *vector*[*rat5* (−*1*) *0*, *rat5* (−*1*) *0*, *rat5 1 0*] :: *real^3*
and *n* ≡ *vector*[*rat5* (−*2*) *0*, *rat5 1* (−*1*), *rat5 0 0*] :: *real^3*
shows (*e face-of* (*convex hull* {*v1*, *v2*, *v3*, *v4*, *v5*}) ∧ *aff-dim e* = *1*) =
(*e* = *convex hull* {*v1*, *v2*} ∨ *e* = *convex hull* {*v1*, *v5*} ∨ *e* = *convex hull* {*v2*, *v4*} ∨ *e* = *convex hull* {*v3*, *v4*} ∨ *e* = *convex hull* {*v3*, *v5*})
proof −
have *n-nz*: *n* ≠ *0*
unfolding *n-def* **by** (*simp add*: *vector3-eq-0 rat5-eq-0*)
have *plane*: ⋀*x*. *x* ∈ {*v1*, *v2*, *v3*, *v4*, *v5*} ⟹ *n* · *x* = *rat5 1 1*
unfolding *assms*
by (*auto simp add*: *vector3-dot rat5-mul rat5-add*)
have *finite*: *finite* {*v1*, *v2*, *v3*, *v4*, *v5*}
using *assms* **by** *blast*
show *?thesis*
by (*facet-edges plane*: *plane n-nz*: *n-nz defs*: *assms*)
qed

lemma *dodecahedron-facet2-edges*:
defines *v1* ≡ *vector*[*rat5* (−*1*/*2*) (−*1*/*2*), *rat5 0 0*, *rat5* (−*1*/*2*) (*1*/*2*)] :: *real^3*
and *v2* ≡ *vector*[*rat5* (−*1*/*2*) (−*1*/*2*), *rat5 0 0*, *rat5* (*1*/*2*) (−*1*/*2*)] :: *real^3*
and *v3* ≡ *vector*[*rat5* (*1*/*2*) (−*1*/*2*), *rat5* (*1*/*2*) (*1*/*2*), *rat5 0 0*] :: *real^3*
and *v4* ≡ *vector*[*rat5* (−*1*) *0*, *rat5 1 0*, *rat5* (−*1*) *0*] :: *real^3*
and *v5* ≡ *vector*[*rat5* (−*1*) *0*, *rat5 1 0*, *rat5 1 0*] :: *real^3*
and *n* ≡ *vector*[*rat5 2 0*, *rat5 1* (−*1*), *rat5 0 0*] :: *real^3*
shows (*e face-of* (*convex hull* {*v1*, *v2*, *v3*, *v4*, *v5*}) ∧ *aff-dim e* = *1*) =
(*e* = *convex hull* {*v1*, *v2*} ∨ *e* = *convex hull* {*v1*, *v5*} ∨ *e* = *convex hull* {*v2*, *v4*} ∨ *e* = *convex hull* {*v3*, *v4*} ∨ *e* = *convex hull* {*v3*, *v5*})
proof −
have *n-nz*: *n* ≠ *0*
unfolding *n-def* **by** (*simp add*: *vector3-eq-0 rat5-eq-0*)
have *plane*: ⋀*x*. *x* ∈ {*v1*, *v2*, *v3*, *v4*, *v5*} ⟹ *n* · *x* = *rat5* (−*1*) (−*1*)
unfolding *assms*
by (*auto simp add*: *vector3-dot rat5-mul rat5-add*)
have *finite*: *finite* {*v1*, *v2*, *v3*, *v4*, *v5*}
using *assms* **by** *blast*
show *?thesis*
by (*facet-edges plane*: *plane n-nz*: *n-nz defs*: *assms*)
qed

lemma *dodecahedron-facet3-edges*:
defines *v1* ≡ *vector*[*rat5* (−*1*/*2*) (−*1*/*2*), *rat5 0 0*, *rat5* (−*1*/*2*) (*1*/*2*)] :: *real^3*
and *v2* ≡ *vector*[*rat5* (−*1*) *0*, *rat5* (−*1*) *0*, *rat5 1 0*] :: *real^3*
and *v3* ≡ *vector*[*rat5* (−*1*) *0*, *rat5 1 0*, *rat5 1 0*] :: *real^3*
and *v4* ≡ *vector*[*rat5 0 0*, *rat5* (−*1*/*2*) (*1*/*2*), *rat5* (*1*/*2*) (*1*/*2*)] :: *real^3*

and *v5* ≡ *vector*[*rat5 0 0*, *rat5* (*1/2*) (−*1/2*), *rat5* (*1/2*) (*1/2*)] :: *real^3*
and *n* ≡ *vector*[*rat5* (−*3*) *1*, *rat5 0 0*, *rat5* (−*1*) *1*] :: *real^3*
shows (*e face-of* (*convex hull* {*v1*, *v2*, *v3*, *v4*, *v5*}) ∧ *aff-dim e* = *1*) =
(*e* = *convex hull* {*v1*, *v2*} ∨ *e* = *convex hull* {*v1*, *v3*} ∨ *e* = *convex hull* {*v2*, *v5*} ∨ *e* = *convex hull* {*v3*, *v4*} ∨ *e* = *convex hull* {*v4*, *v5*})
proof −
have *n-nz*: *n* ≠ *0*
unfolding *n-def* **by** (*simp add*: *vector3-eq-0 rat5-eq-0*)
have *plane*: ⋀*x*. *x* ∈ {*v1*, *v2*, *v3*, *v4*, *v5*} ⟹ *n* · *x* = *rat5 2 0*
unfolding *assms*
by (*auto simp add*: *vector3-dot rat5-mul rat5-add*)
have *finite*: *finite* {*v1*, *v2*, *v3*, *v4*, *v5*}
using *assms* **by** *blast*
show *?thesis*
by (*facet-edges plane*: *plane n-nz*: *n-nz defs*: *assms*)
qed

lemma *dodecahedron-facet4-edges*:
defines *v1* ≡ *vector*[*rat5* (−*1/2*) (−*1/2*), *rat5 0 0*, *rat5* (*1/2*) (−*1/2*)] :: *real^3*
and *v2* ≡ *vector*[*rat5* (−*1*) *0*, *rat5* (−*1*) *0*, *rat5* (−*1*) *0*] :: *real^3*
and *v3* ≡ *vector*[*rat5* (−*1*) *0*, *rat5 1 0*, *rat5* (−*1*) *0*] :: *real^3*
and *v4* ≡ *vector*[*rat5 0 0*, *rat5* (−*1/2*) (*1/2*), *rat5* (−*1/2*) (−*1/2*)] :: *real^3*
and *v5* ≡ *vector*[*rat5 0 0*, *rat5* (*1/2*) (−*1/2*), *rat5* (−*1/2*) (−*1/2*)] :: *real^3*
and *n* ≡ *vector*[*rat5 3* (−*1*), *rat5 0 0*, *rat5* (−*1*) *1*] :: *real^3*
shows (*e face-of* (*convex hull* {*v1*, *v2*, *v3*, *v4*, *v5*}) ∧ *aff-dim e* = *1*) =
(*e* = *convex hull* {*v1*, *v2*} ∨ *e* = *convex hull* {*v1*, *v3*} ∨ *e* = *convex hull* {*v2*, *v5*} ∨ *e* = *convex hull* {*v3*, *v4*} ∨ *e* = *convex hull* {*v4*, *v5*})
proof −
have *n-nz*: *n* ≠ *0*
unfolding *n-def* **by** (*simp add*: *vector3-eq-0 rat5-eq-0*)
have *plane*: ⋀*x*. *x* ∈ {*v1*, *v2*, *v3*, *v4*, *v5*} ⟹ *n* · *x* = *rat5* (−*2*) *0*
unfolding *assms*
by (*auto simp add*: *vector3-dot rat5-mul rat5-add*)
have *finite*: *finite* {*v1*, *v2*, *v3*, *v4*, *v5*}
using *assms* **by** *blast*
show *?thesis*
by (*facet-edges plane*: *plane n-nz*: *n-nz defs*: *assms*)
qed

lemma *dodecahedron-facet5-edges*:
defines *v1* ≡ *vector*[*rat5* (−*1/2*) (*1/2*), *rat5* (−*1/2*) (−*1/2*), *rat5 0 0*] :: *real^3*
and *v2* ≡ *vector*[*rat5* (*1/2*) (−*1/2*), *rat5* (−*1/2*) (−*1/2*), *rat5 0 0*] :: *real^3*
and *v3* ≡ *vector*[*rat5* (−*1*) *0*, *rat5* (−*1*) *0*, *rat5* (−*1*) *0*] :: *real^3*
and *v4* ≡ *vector*[*rat5 1 0*, *rat5* (−*1*) *0*, *rat5* (−*1*) *0*] :: *real^3*
and *v5* ≡ *vector*[*rat5 0 0*, *rat5* (*1/2*) (−*1/2*), *rat5* (−*1/2*) (−*1/2*)] :: *real^3*
and *n* ≡ *vector*[*rat5 0 0*, *rat5 1* (−*1*), *rat5* (−*3*) *1*] :: *real^3*
shows (*e face-of* (*convex hull* {*v1*, *v2*, *v3*, *v4*, *v5*}) ∧ *aff-dim e* = *1*) =
(*e* = *convex hull* {*v1*, *v2*} ∨ *e* = *convex hull* {*v1*, *v4*} ∨ *e* = *convex hull* {*v2*, *v3*} ∨ *e* = *convex hull* {*v3*, *v5*} ∨ *e* = *convex hull* {*v4*, *v5*})

proof −
  have *n-nz*: $n \neq 0$
    unfolding *n-def* **by** (*simp add*: *vector3-eq-0 rat5-eq-0*)
  have *plane*: $\bigwedge x.\ x \in \{v1,\ v2,\ v3,\ v4,\ v5\} \Longrightarrow n \cdot x = rat5\ 2\ 0$
    unfolding *assms*
    by (*auto simp add*: *vector3-dot rat5-mul rat5-add*)
  have *finite*: *finite* $\{v1,\ v2,\ v3,\ v4,\ v5\}$
    using *assms* **by** *blast*
  show *?thesis*
    by (*facet-edges plane*: *plane n-nz*: *n-nz defs*: *assms*)
qed

lemma *dodecahedron-facet6-edges*:
  defines $v1 \equiv vector[rat5\ (-1/2)\ (1/2),\ rat5\ (-1/2)\ (-1/2),\ rat5\ 0\ 0] :: real\hat{}3$
    and $v2 \equiv vector[rat5\ (1/2)\ (-1/2),\ rat5\ (-1/2)\ (-1/2),\ rat5\ 0\ 0] :: real\hat{}3$
    and $v3 \equiv vector[rat5\ (-1)\ 0,\ rat5\ (-1)\ 0,\ rat5\ 1\ 0] :: real\hat{}3$
    and $v4 \equiv vector[rat5\ 1\ 0,\ rat5\ (-1)\ 0,\ rat5\ 1\ 0] :: real\hat{}3$
    and $v5 \equiv vector[rat5\ 0\ 0,\ rat5\ (1/2)\ (-1/2),\ rat5\ (1/2)\ (1/2)] :: real\hat{}3$
    and $n \equiv vector[rat5\ 0\ 0,\ rat5\ (-1)\ 1,\ rat5\ (-3)\ 1] :: real\hat{}3$
  shows (e *face-of* (*convex hull* $\{v1,\ v2,\ v3,\ v4,\ v5\}$) $\wedge$ *aff-dim* $e = 1$) =
    ($e =$ *convex hull* $\{v1,\ v2\} \vee e =$ *convex hull* $\{v1,\ v4\} \vee e =$ *convex hull* $\{v2,\ v3\} \vee e =$ *convex hull* $\{v3,\ v5\} \vee e =$ *convex hull* $\{v4,\ v5\}$)
proof −
  have *n-nz*: $n \neq 0$
    unfolding *n-def* **by** (*simp add*: *vector3-eq-0 rat5-eq-0*)
  have *plane*: $\bigwedge x.\ x \in \{v1,\ v2,\ v3,\ v4,\ v5\} \Longrightarrow n \cdot x = rat5\ (-2)\ 0$
    unfolding *assms*
    by (*auto simp add*: *vector3-dot rat5-mul rat5-add*)
  have *finite*: *finite* $\{v1,\ v2,\ v3,\ v4,\ v5\}$
    using *assms* **by** *blast*
  show *?thesis*
    by (*facet-edges plane*: *plane n-nz*: *n-nz defs*: *assms*)
qed

lemma *dodecahedron-facet7-edges*:
  defines $v1 \equiv vector[rat5\ (-1/2)\ (1/2),\ rat5\ (-1/2)\ (-1/2),\ rat5\ 0\ 0] :: real\hat{}3$
    and $v2 \equiv vector[rat5\ (1/2)\ (1/2),\ rat5\ 0\ 0,\ rat5\ (-1/2)\ (1/2)] :: real\hat{}3$
    and $v3 \equiv vector[rat5\ (1/2)\ (1/2),\ rat5\ 0\ 0,\ rat5\ (1/2)\ (-1/2)] :: real\hat{}3$
    and $v4 \equiv vector[rat5\ 1\ 0,\ rat5\ (-1)\ 0,\ rat5\ (-1)\ 0] :: real\hat{}3$
    and $v5 \equiv vector[rat5\ 1\ 0,\ rat5\ (-1)\ 0,\ rat5\ 1\ 0] :: real\hat{}3$
    and $n \equiv vector[rat5\ (-2)\ 0,\ rat5\ (-1)\ 1,\ rat5\ 0\ 0] :: real\hat{}3$
  shows (e *face-of* (*convex hull* $\{v1,\ v2,\ v3,\ v4,\ v5\}$) $\wedge$ *aff-dim* $e = 1$) =
    ($e =$ *convex hull* $\{v1,\ v4\} \vee e =$ *convex hull* $\{v1,\ v5\} \vee e =$ *convex hull* $\{v2,\ v3\} \vee e =$ *convex hull* $\{v2,\ v5\} \vee e =$ *convex hull* $\{v3,\ v4\}$)
proof −
  have *n-nz*: $n \neq 0$
    unfolding *n-def* **by** (*simp add*: *vector3-eq-0 rat5-eq-0*)
  have *plane*: $\bigwedge x.\ x \in \{v1,\ v2,\ v3,\ v4,\ v5\} \Longrightarrow n \cdot x = rat5\ (-1)\ (-1)$
    unfolding *assms*

by (*auto simp add*: *vector3-dot rat5-mul rat5-add*)
  have *finite*: *finite* {*v1*, *v2*, *v3*, *v4*, *v5*}
    using *assms* **by** *blast*
  show *?thesis*
    by (*facet-edges plane*: *plane n-nz*: *n-nz defs*: *assms*)
qed

lemma *dodecahedron-facet8-edges*:
  defines *v1* ≡ *vector*[*rat5* (−*1*/*2*) (*1*/*2*), *rat5* (*1*/*2*) (*1*/*2*), *rat5 0 0*] :: *real^3*
    and *v2* ≡ *vector*[*rat5* (*1*/*2*) (−*1*/*2*), *rat5* (*1*/*2*) (*1*/*2*), *rat5 0 0*] :: *real^3*
    and *v3* ≡ *vector*[*rat5* (−*1*) *0*, *rat5 1 0*, *rat5* (−*1*) *0*] :: *real^3*
    and *v4* ≡ *vector*[*rat5 1 0*, *rat5 1 0*, *rat5* (−*1*) *0*] :: *real^3*
    and *v5* ≡ *vector*[*rat5 0 0*, *rat5* (−*1*/*2*) (*1*/*2*), *rat5* (−*1*/*2*) (−*1*/*2*)] :: *real^3*
    and *n* ≡ *vector*[*rat5 0 0*, *rat5 1* (−*1*), *rat5 3* (−*1*)] :: *real^3*
  shows (*e face-of* (*convex hull* {*v1*, *v2*, *v3*, *v4*, *v5*}) ∧ *aff-dim e* = *1*) =
      (*e* = *convex hull* {*v1*, *v2*} ∨ *e* = *convex hull* {*v1*, *v4*} ∨ *e* = *convex hull* {*v2*, *v3*} ∨ *e* = *convex hull* {*v3*, *v5*} ∨ *e* = *convex hull* {*v4*, *v5*})
proof −
  have *n-nz*: *n* ≠ *0*
    unfolding *n-def* **by** (*simp add*: *vector3-eq-0 rat5-eq-0*)
  have *plane*: ⋀*x*. *x* ∈ {*v1*, *v2*, *v3*, *v4*, *v5*} ⟹ *n* · *x* = *rat5* (−*2*) *0*
    unfolding *assms*
    by (*auto simp add*: *vector3-dot rat5-mul rat5-add*)
  have *finite*: *finite* {*v1*, *v2*, *v3*, *v4*, *v5*}
    using *assms* **by** *blast*
  show *?thesis*
    by (*facet-edges plane*: *plane n-nz*: *n-nz defs*: *assms*)
qed

lemma *dodecahedron-facet9-edges*:
  defines *v1* ≡ *vector*[*rat5* (−*1*/*2*) (*1*/*2*), *rat5* (*1*/*2*) (*1*/*2*), *rat5 0 0*] :: *real^3*
    and *v2* ≡ *vector*[*rat5* (*1*/*2*) (−*1*/*2*), *rat5* (*1*/*2*) (*1*/*2*), *rat5 0 0*] :: *real^3*
    and *v3* ≡ *vector*[*rat5* (−*1*) *0*, *rat5 1 0*, *rat5 1 0*] :: *real^3*
    and *v4* ≡ *vector*[*rat5 1 0*, *rat5 1 0*, *rat5 1 0*] :: *real^3*
    and *v5* ≡ *vector*[*rat5 0 0*, *rat5* (−*1*/*2*) (*1*/*2*), *rat5* (*1*/*2*) (*1*/*2*)] :: *real^3*
    and *n* ≡ *vector*[*rat5 0 0*, *rat5* (−*1*) *1*, *rat5 3* (−*1*)] :: *real^3*
  shows (*e face-of* (*convex hull* {*v1*, *v2*, *v3*, *v4*, *v5*}) ∧ *aff-dim e* = *1*) =
      (*e* = *convex hull* {*v1*, *v2*} ∨ *e* = *convex hull* {*v1*, *v4*} ∨ *e* = *convex hull* {*v2*, *v3*} ∨ *e* = *convex hull* {*v3*, *v5*} ∨ *e* = *convex hull* {*v4*, *v5*})
proof −
  have *n-nz*: *n* ≠ *0*
    unfolding *n-def* **by** (*simp add*: *vector3-eq-0 rat5-eq-0*)
  have *plane*: ⋀*x*. *x* ∈ {*v1*, *v2*, *v3*, *v4*, *v5*} ⟹ *n* · *x* = *rat5 2 0*
    unfolding *assms*
    by (*auto simp add*: *vector3-dot rat5-mul rat5-add*)
  have *finite*: *finite* {*v1*, *v2*, *v3*, *v4*, *v5*}
    using *assms* **by** *blast*
  show *?thesis*
    by (*facet-edges plane*: *plane n-nz*: *n-nz defs*: *assms*)

qed

lemma *dodecahedron-facet10-edges*:
  defines $v1 \equiv vector[rat5\ (-1/2)\ (1/2),\ rat5\ (1/2)\ (1/2),\ rat5\ 0\ 0] :: real\hat{\ }3$
    and $v2 \equiv vector[rat5\ (1/2)\ (1/2),\ rat5\ 0\ 0,\ rat5\ (-1/2)\ (1/2)] :: real\hat{\ }3$
    and $v3 \equiv vector[rat5\ (1/2)\ (1/2),\ rat5\ 0\ 0,\ rat5\ (1/2)\ (-1/2)] :: real\hat{\ }3$
    and $v4 \equiv vector[rat5\ 1\ 0,\ rat5\ 1\ 0,\ rat5\ (-1)\ 0] :: real\hat{\ }3$
    and $v5 \equiv vector[rat5\ 1\ 0,\ rat5\ 1\ 0,\ rat5\ 1\ 0] :: real\hat{\ }3$
    and $n \equiv vector[rat5\ 2\ 0,\ rat5\ (-1)\ 1,\ rat5\ 0\ 0] :: real\hat{\ }3$
   shows $(e\ face\text{-}of\ (convex\ hull\ \{v1,\ v2,\ v3,\ v4,\ v5\}) \wedge aff\text{-}dim\ e = 1) =$
      $(e = convex\ hull\ \{v1,\ v4\} \vee e = convex\ hull\ \{v1,\ v5\} \vee e = convex\ hull$
$\{v2,\ v3\} \vee e = convex\ hull\ \{v2,\ v5\} \vee e = convex\ hull\ \{v3,\ v4\})$
proof –
  have *n-nz*: $n \neq 0$
   unfolding *n-def* **by** (*simp add*: *vector3-eq-0 rat5-eq-0*)
  have *plane*: $\bigwedge x.\ x \in \{v1,\ v2,\ v3,\ v4,\ v5\} \Longrightarrow n \cdot x = rat5\ 1\ 1$
   unfolding *assms*
   by (*auto simp add*: *vector3-dot rat5-mul rat5-add*)
  have *finite*: *finite* $\{v1,\ v2,\ v3,\ v4,\ v5\}$
   using *assms* **by** *blast*
  show *?thesis*
   by (*facet-edges plane*: *plane n-nz*: *n-nz defs*: *assms*)
qed

lemma *dodecahedron-facet11-edges*:
  defines $v1 \equiv vector[rat5\ (1/2)\ (1/2),\ rat5\ 0\ 0,\ rat5\ (-1/2)\ (1/2)] :: real\hat{\ }3$
    and $v2 \equiv vector[rat5\ 1\ 0,\ rat5\ (-1)\ 0,\ rat5\ 1\ 0] :: real\hat{\ }3$
    and $v3 \equiv vector[rat5\ 1\ 0,\ rat5\ 1\ 0,\ rat5\ 1\ 0] :: real\hat{\ }3$
    and $v4 \equiv vector[rat5\ 0\ 0,\ rat5\ (-1/2)\ (1/2),\ rat5\ (1/2)\ (1/2)] :: real\hat{\ }3$
    and $v5 \equiv vector[rat5\ 0\ 0,\ rat5\ (1/2)\ (-1/2),\ rat5\ (1/2)\ (1/2)] :: real\hat{\ }3$
    and $n \equiv vector[rat5\ (-3)\ 1,\ rat5\ 0\ 0,\ rat5\ 1\ (-1)] :: real\hat{\ }3$
   shows $(e\ face\text{-}of\ (convex\ hull\ \{v1,\ v2,\ v3,\ v4,\ v5\}) \wedge aff\text{-}dim\ e = 1) =$
      $(e = convex\ hull\ \{v1,\ v2\} \vee e = convex\ hull\ \{v1,\ v3\} \vee e = convex\ hull$
$\{v2,\ v5\} \vee e = convex\ hull\ \{v3,\ v4\} \vee e = convex\ hull\ \{v4,\ v5\})$
proof –
  have *n-nz*: $n \neq 0$
   unfolding *n-def* **by** (*simp add*: *vector3-eq-0 rat5-eq-0*)
  have *plane*: $\bigwedge x.\ x \in \{v1,\ v2,\ v3,\ v4,\ v5\} \Longrightarrow n \cdot x = rat5\ (-2)\ 0$
   unfolding *assms*
   by (*auto simp add*: *vector3-dot rat5-mul rat5-add*)
  have *finite*: *finite* $\{v1,\ v2,\ v3,\ v4,\ v5\}$
   using *assms* **by** *blast*
  show *?thesis*
   by (*facet-edges plane*: *plane n-nz*: *n-nz defs*: *assms*)
qed

lemma *dodecahedron-facet12-edges*:
  defines $v1 \equiv vector[rat5\ (1/2)\ (1/2),\ rat5\ 0\ 0,\ rat5\ (1/2)\ (-1/2)] :: real\hat{\ }3$
    and $v2 \equiv vector[rat5\ 1\ 0,\ rat5\ (-1)\ 0,\ rat5\ (-1)\ 0] :: real\hat{\ }3$

and *v3* ≡ *vector*[*rat5 1 0*, *rat5 1 0*, *rat5* (−*1*) *0*] :: *real^3*
and *v4* ≡ *vector*[*rat5 0 0*, *rat5* (−*1*/*2*) (*1*/*2*), *rat5* (−*1*/*2*) (−*1*/*2*)] :: *real^3*
and *v5* ≡ *vector*[*rat5 0 0*, *rat5* (*1*/*2*) (−*1*/*2*), *rat5* (−*1*/*2*) (−*1*/*2*)] :: *real^3*
and *n* ≡ *vector*[*rat5 3* (−*1*), *rat5 0 0*, *rat5 1* (−*1*)] :: *real^3*
shows (*e face-of* (*convex hull* {*v1*, *v2*, *v3*, *v4*, *v5*}) ∧ *aff-dim e* = *1*) =
(*e* = *convex hull* {*v1*, *v2*} ∨ *e* = *convex hull* {*v1*, *v3*} ∨ *e* = *convex hull* {*v2*, *v5*} ∨ *e* = *convex hull* {*v3*, *v4*} ∨ *e* = *convex hull* {*v4*, *v5*})
proof −
have *n-nz*: *n* ≠ *0*
unfolding *n-def* **by** (*simp add*: *vector3-eq-0 rat5-eq-0*)
have *plane*: ⋀*x*. *x* ∈ {*v1*, *v2*, *v3*, *v4*, *v5*} ⟹ *n* · *x* = *rat5 2 0*
unfolding *assms*
by (*auto simp add*: *vector3-dot rat5-mul rat5-add*)
have *finite*: *finite* {*v1*, *v2*, *v3*, *v4*, *v5*}
using *assms* **by** *blast*
show *?thesis*
by (*facet-edges plane*: *plane n-nz*: *n-nz defs*: *assms*)
qed

lemmas *dodecahedron-facet-edges* =
dodecahedron-facet1-edges dodecahedron-facet2-edges dodecahedron-facet3-edges
dodecahedron-facet4-edges dodecahedron-facet5-edges dodecahedron-facet6-edges
dodecahedron-facet7-edges dodecahedron-facet8-edges dodecahedron-facet9-edges
dodecahedron-facet10-edges dodecahedron-facet11-edges dodecahedron-facet12-edges

lemma *dodecahedron-edges-per-face*:
assumes *f face-of std-dodecahedron*
and *aff-dim f* = *2*
shows *card* {*e*. *e face-of std-dodecahedron* ∧ *aff-dim e* = *1* ∧ *e* ⊆ *f*} = *5*
using *conjI*[*OF assms*] **unfolding** *dodecahedron-facets face-edge-set-eq*[*OF assms*(*1*)]
apply (*elim disjE forw-subst*)
unfolding *dodecahedron-facet-edges set-cases card-5-iff*
segment-convex-hull[*symmetric*] *closed-segment-eq*
by (*simp-all add*: *vector3-eq doubleton-eq-iff rat5-eq*)

lemma *dodecahedron-edges*:
e face-of std-dodecahedron ∧ *aff-dim e* = *1* ⟷
e = *convex hull* {*vector*[*rat5* (−*1*/*2*) (−*1*/*2*), *rat5 0 0*, *rat5* (−*1*/*2*) (*1*/*2*)], *vector*[*rat5* (−*1*/*2*) (−*1*/*2*), *rat5 0 0*, *rat5* (*1*/*2*) (−*1*/*2*)]} ∨
e = *convex hull* {*vector*[*rat5* (−*1*/*2*) (−*1*/*2*), *rat5 0 0*, *rat5* (−*1*/*2*) (*1*/*2*)], *vector*[*rat5* (−*1*) *0*, *rat5* (−*1*) *0*, *rat5 1 0*]} ∨
e = *convex hull* {*vector*[*rat5* (−*1*/*2*) (−*1*/*2*), *rat5 0 0*, *rat5* (−*1*/*2*) (*1*/*2*)], *vector*[*rat5* (−*1*) *0*, *rat5 1 0*, *rat5 1 0*]} ∨
e = *convex hull* {*vector*[*rat5* (−*1*/*2*) (−*1*/*2*), *rat5 0 0*, *rat5* (*1*/*2*) (−*1*/*2*)], *vector*[*rat5* (−*1*) *0*, *rat5* (−*1*) *0*, *rat5* (−*1*) *0*]} ∨
e = *convex hull* {*vector*[*rat5* (−*1*/*2*) (−*1*/*2*), *rat5 0 0*, *rat5* (*1*/*2*) (−*1*/*2*)], *vector*[*rat5* (−*1*) *0*, *rat5 1 0*, *rat5* (−*1*) *0*]} ∨
e = *convex hull* {*vector*[*rat5* (−*1*/*2*) (*1*/*2*), *rat5* (−*1*/*2*) (−*1*/*2*), *rat5 0 0*], *vector*[*rat5* (*1*/*2*) (−*1*/*2*), *rat5* (−*1*/*2*) (−*1*/*2*), *rat5 0 0*]} ∨

e = *convex hull* {*vector*[*rat5* (−*1*/*2*) (*1*/*2*), *rat5* (−*1*/*2*) (−*1*/*2*), *rat5* *0* *0*], *vector*[*rat5* *1* *0*, *rat5* (−*1*) *0*, *rat5* (−*1*) *0*]} ∨
e = *convex hull* {*vector*[*rat5* (−*1*/*2*) (*1*/*2*), *rat5* (−*1*/*2*) (−*1*/*2*), *rat5* *0* *0*], *vector*[*rat5* *1* *0*, *rat5* (−*1*) *0*, *rat5* *1* *0*]} ∨
e = *convex hull* {*vector*[*rat5* (−*1*/*2*) (*1*/*2*), *rat5* (*1*/*2*) (*1*/*2*), *rat5* *0* *0*], *vector*[*rat5* (*1*/*2*) (−*1*/*2*), *rat5* (*1*/*2*) (*1*/*2*), *rat5* *0* *0*]} ∨
e = *convex hull* {*vector*[*rat5* (−*1*/*2*) (*1*/*2*), *rat5* (*1*/*2*) (*1*/*2*), *rat5* *0* *0*], *vector*[*rat5* *1* *0*, *rat5* *1* *0*, *rat5* (−*1*) *0*]} ∨
e = *convex hull* {*vector*[*rat5* (−*1*/*2*) (*1*/*2*), *rat5* (*1*/*2*) (*1*/*2*), *rat5* *0* *0*], *vector*[*rat5* *1* *0*, *rat5* *1* *0*, *rat5* *1* *0*]} ∨
e = *convex hull* {*vector*[*rat5* (*1*/*2*) (−*1*/*2*), *rat5* (−*1*/*2*) (−*1*/*2*), *rat5* *0* *0*], *vector*[*rat5* (−*1*) *0*, *rat5* (−*1*) *0*, *rat5* (−*1*) *0*]} ∨
e = *convex hull* {*vector*[*rat5* (*1*/*2*) (−*1*/*2*), *rat5* (−*1*/*2*) (−*1*/*2*), *rat5* *0* *0*], *vector*[*rat5* (−*1*) *0*, *rat5* (−*1*) *0*, *rat5* *1* *0*]} ∨
e = *convex hull* {*vector*[*rat5* (*1*/*2*) (−*1*/*2*), *rat5* (*1*/*2*) (*1*/*2*), *rat5* *0* *0*], *vector*[*rat5* (−*1*) *0*, *rat5* *1* *0*, *rat5* (−*1*) *0*]} ∨
e = *convex hull* {*vector*[*rat5* (*1*/*2*) (−*1*/*2*), *rat5* (*1*/*2*) (*1*/*2*), *rat5* *0* *0*], *vector*[*rat5* (−*1*) *0*, *rat5* *1* *0*, *rat5* *1* *0*]} ∨
e = *convex hull* {*vector*[*rat5* (*1*/*2*) (*1*/*2*), *rat5* *0* *0*, *rat5* (−*1*/*2*) (*1*/*2*)], *vector*[*rat5* (*1*/*2*) (*1*/*2*), *rat5* *0* *0*, *rat5* (*1*/*2*) (−*1*/*2*)]} ∨
e = *convex hull* {*vector*[*rat5* (*1*/*2*) (*1*/*2*), *rat5* *0* *0*, *rat5* (−*1*/*2*) (*1*/*2*)], *vector*[*rat5* *1* *0*, *rat5* (−*1*) *0*, *rat5* *1* *0*]} ∨
e = *convex hull* {*vector*[*rat5* (*1*/*2*) (*1*/*2*), *rat5* *0* *0*, *rat5* (−*1*/*2*) (*1*/*2*)], *vector*[*rat5* *1* *0*, *rat5* *1* *0*, *rat5* *1* *0*]} ∨
e = *convex hull* {*vector*[*rat5* (*1*/*2*) (*1*/*2*), *rat5* *0* *0*, *rat5* (*1*/*2*) (−*1*/*2*)], *vector*[*rat5* *1* *0*, *rat5* (−*1*) *0*, *rat5* (−*1*) *0*]} ∨
e = *convex hull* {*vector*[*rat5* (*1*/*2*) (*1*/*2*), *rat5* *0* *0*, *rat5* (*1*/*2*) (−*1*/*2*)], *vector*[*rat5* *1* *0*, *rat5* *1* *0*, *rat5* (−*1*) *0*]} ∨
e = *convex hull* {*vector*[*rat5* (−*1*) *0*, *rat5* (−*1*) *0*, *rat5* (−*1*) *0*], *vector*[*rat5* *0* *0*, *rat5* (*1*/*2*) (−*1*/*2*), *rat5* (−*1*/*2*) (−*1*/*2*)]} ∨
e = *convex hull* {*vector*[*rat5* (−*1*) *0*, *rat5* (−*1*) *0*, *rat5* *1* *0*], *vector*[*rat5* *0* *0*, *rat5* (*1*/*2*) (−*1*/*2*), *rat5* (*1*/*2*) (*1*/*2*)]} ∨
e = *convex hull* {*vector*[*rat5* (−*1*) *0*, *rat5* *1* *0*, *rat5* (−*1*) *0*], *vector*[*rat5* *0* *0*, *rat5* (−*1*/*2*) (*1*/*2*), *rat5* (−*1*/*2*) (−*1*/*2*)]} ∨
e = *convex hull* {*vector*[*rat5* (−*1*) *0*, *rat5* *1* *0*, *rat5* *1* *0*], *vector*[*rat5* *0* *0*, *rat5* (−*1*/*2*) (*1*/*2*), *rat5* (*1*/*2*) (*1*/*2*)]} ∨
e = *convex hull* {*vector*[*rat5* *1* *0*, *rat5* (−*1*) *0*, *rat5* (−*1*) *0*], *vector*[*rat5* *0* *0*, *rat5* (*1*/*2*) (−*1*/*2*), *rat5* (−*1*/*2*) (−*1*/*2*)]} ∨
e = *convex hull* {*vector*[*rat5* *1* *0*, *rat5* (−*1*) *0*, *rat5* *1* *0*], *vector*[*rat5* *0* *0*, *rat5* (*1*/*2*) (−*1*/*2*), *rat5* (*1*/*2*) (*1*/*2*)]} ∨
e = *convex hull* {*vector*[*rat5* *1* *0*, *rat5* *1* *0*, *rat5* (−*1*) *0*], *vector*[*rat5* *0* *0*, *rat5* (−*1*/*2*) (*1*/*2*), *rat5* (−*1*/*2*) (−*1*/*2*)]} ∨
e = *convex hull* {*vector*[*rat5* *1* *0*, *rat5* *1* *0*, *rat5* *1* *0*], *vector*[*rat5* *0* *0*, *rat5* (−*1*/*2*) (*1*/*2*), *rat5* (*1*/*2*) (*1*/*2*)]} ∨
e = *convex hull* {*vector*[*rat5* *0* *0*, *rat5* (−*1*/*2*) (*1*/*2*), *rat5* (−*1*/*2*) (−*1*/*2*)], *vector*[*rat5* *0* *0*, *rat5* (*1*/*2*) (−*1*/*2*), *rat5* (−*1*/*2*) (−*1*/*2*)]} ∨
e = *convex hull* {*vector*[*rat5* *0* *0*, *rat5* (−*1*/*2*) (*1*/*2*), *rat5* (*1*/*2*) (*1*/*2*)], *vector*[*rat5* *0* *0*, *rat5* (*1*/*2*) (−*1*/*2*), *rat5* (*1*/*2*) (*1*/*2*)]}
(**is** *?lhs* ⟷ *?rhs*)

proof
  assume *?lhs*
  then obtain *f* **where** *f-facts*: *f face-of std-dodecahedron* $\wedge$ *aff-dim f* = *2 e face-of f* $\wedge$ *aff-dim e* = *1*
    using *dodecahedron-polyhedron dodecahedron-fulldim edge-belongs-to-face*
    by *blast*
  show *?rhs*
    using *f-facts* **unfolding** *dodecahedron-facets*
    apply (*elim disjE*) **apply** (*all* ‹*hypsubst-thin*›)
    unfolding *dodecahedron-facet-edges*
    by *linarith+*
next
  assume *?rhs*
  then show *?lhs*
    using *dodecahedron-facets dodecahedron-facet-edges*
   by (*smt* (*verit*, *ccfv-threshold*) *face-of-trans insertI1 insert-commute mem-Collect-eq*)
qed

lemma *dodecahedron-vertices*:
  shows (*v face-of std-dodecahedron* $\wedge$ *aff-dim v* = *0*) =
      (*v* = {*vector*[*rat5 1 0*, *rat5 1 0*, *rat5 1 0*]} $\vee$
       v = {*vector*[*rat5 1 0*, *rat5 1 0*, *rat5* (−*1*) *0*]} $\vee$
       v = {*vector*[*rat5 1 0*, *rat5* (−*1*) *0*, *rat5 1 0*]} $\vee$
       v = {*vector*[*rat5 1 0*, *rat5* (−*1*) *0*, *rat5* (−*1*) *0*]} $\vee$
       v = {*vector*[*rat5* (−*1*) *0*, *rat5 1 0*, *rat5 1 0*]} $\vee$
       v = {*vector*[*rat5* (−*1*) *0*, *rat5 1 0*, *rat5* (−*1*) *0*]} $\vee$
       v = {*vector*[*rat5* (−*1*) *0*, *rat5* (−*1*) *0*, *rat5 1 0*]} $\vee$
       v = {*vector*[*rat5* (−*1*) *0*, *rat5* (−*1*) *0*, *rat5* (−*1*) *0*]} $\vee$
       v = {*vector*[*rat5 0 0*, *rat5* (−*1/2*) (*1/2*), *rat5* (*1/2*) (*1/2*)]} $\vee$
       v = {*vector*[*rat5 0 0*, *rat5* (−*1/2*) (*1/2*), *rat5* (−*1/2*) (−*1/2*)]} $\vee$
       v = {*vector*[*rat5 0 0*, *rat5* (*1/2*) (−*1/2*), *rat5* (*1/2*) (*1/2*)]} $\vee$
       v = {*vector*[*rat5 0 0*, *rat5* (*1/2*) (−*1/2*), *rat5* (−*1/2*) (−*1/2*)]} $\vee$
       v = {*vector*[*rat5* (−*1/2*) (*1/2*), *rat5* (*1/2*) (*1/2*), *rat5 0 0*]} $\vee$
       v = {*vector*[*rat5* (−*1/2*) (*1/2*), *rat5* (−*1/2*) (−*1/2*), *rat5 0 0*]} $\vee$
       v = {*vector*[*rat5* (*1/2*) (−*1/2*), *rat5* (*1/2*) (*1/2*), *rat5 0 0*]} $\vee$
       v = {*vector*[*rat5* (*1/2*) (−*1/2*), *rat5* (−*1/2*) (−*1/2*), *rat5 0 0*]} $\vee$
       v = {*vector*[*rat5* (*1/2*) (*1/2*), *rat5 0 0*, *rat5* (−*1/2*) (*1/2*)]} $\vee$
       v = {*vector*[*rat5* (−*1/2*) (−*1/2*), *rat5 0 0*, *rat5* (−*1/2*) (*1/2*)]} $\vee$
       v = {*vector*[*rat5* (*1/2*) (*1/2*), *rat5 0 0*, *rat5* (*1/2*) (−*1/2*)]} $\vee$
       v = {*vector*[*rat5* (−*1/2*) (−*1/2*), *rat5 0 0*, *rat5* (*1/2*) (−*1/2*)]})
  (**is** *?lhs* = *?rhs*)
proof
  assume ∗: *?lhs*
  then obtain *e* **where** *e face-of std-dodecahedron aff-dim e* = *1 v face-of e*
    using *dodecahedron-polyhedron dodecahedron-fulldim vertex-belongs-to-edge*
    by *blast*
  show *?rhs*
    using *conjI*[*OF* ‹*e face-of std-dodecahedron*› ‹*aff-dim e* = *1*›]
    unfolding *dodecahedron-edges*

using *vertices-of-edge*[*OF* - ‹*v face-of e*›] ∗
by *metis*+
next
assume *?rhs*
then show *?lhs*
apply (*elim disjE forw-subst*)
by (*smt* (*z3*) *aff-dim-sing extreme-point-of-segment face-of-singleton face-of-trans segment-convex-hull dodecahedron-edges*)+
qed

lemma *dodecahedron-edges-per-vertex*:
assumes *v face-of std-dodecahedron*
and *aff-dim v* = *0*
shows *card* {*e*. *e face-of std-dodecahedron* ∧ *aff-dim e* = *1* ∧ *v* ⊆ *e*} = *3*
unfolding *vertex-edge-set-eq*[*OF assms*(*1*)]
using *conjI*[*OF assms*]
unfolding *dodecahedron-vertices conj-assoc*[*symmetric*] *dodecahedron-edges*
by (*elim disjE forw-subst*) *edges-per-vertex*+

8.3 Regularity

lemma *dodecahedron-congruent-edges*:
assumes *e1 face-of std-dodecahedron* ∧ *aff-dim e1* = *1*
and *e2 face-of std-dodecahedron* ∧ *aff-dim e2* = *1*
shows *e1 congruent e2*
proof −
let *?edges* =
{*convex hull* {(*vector*[*rat5* (−*1*/*2*) (−*1*/*2*), *rat5 0 0*, *rat5* (−*1*/*2*) (*1*/*2*)]::*real*^*3*), *vector*[*rat5* (−*1*/*2*) (−*1*/*2*), *rat5 0 0*, *rat5* (*1*/*2*) (−*1*/*2*)]},
convex hull {*vector*[*rat5* (−*1*/*2*) (−*1*/*2*), *rat5 0 0*, *rat5* (−*1*/*2*) (*1*/*2*)], *vector*[*rat5* (−*1*) *0*, *rat5* (−*1*) *0*, *rat5 1 0*]},
convex hull {*vector*[*rat5* (−*1*/*2*) (−*1*/*2*), *rat5 0 0*, *rat5* (−*1*/*2*) (*1*/*2*)], *vector*[*rat5* (−*1*) *0*, *rat5 1 0*, *rat5 1 0*]},
convex hull {*vector*[*rat5* (−*1*/*2*) (−*1*/*2*), *rat5 0 0*, *rat5* (*1*/*2*) (−*1*/*2*)], *vector*[*rat5* (−*1*) *0*, *rat5* (−*1*) *0*, *rat5* (−*1*) *0*]},
convex hull {*vector*[*rat5* (−*1*/*2*) (−*1*/*2*), *rat5 0 0*, *rat5* (*1*/*2*) (−*1*/*2*)], *vector*[*rat5* (−*1*) *0*, *rat5 1 0*, *rat5* (−*1*) *0*]},
convex hull {*vector*[*rat5* (−*1*/*2*) (*1*/*2*), *rat5* (−*1*/*2*) (−*1*/*2*), *rat5 0 0*], *vector*[*rat5* (*1*/*2*) (−*1*/*2*), *rat5* (−*1*/*2*) (−*1*/*2*), *rat5 0 0*]},
convex hull {*vector*[*rat5* (−*1*/*2*) (*1*/*2*), *rat5* (−*1*/*2*) (−*1*/*2*), *rat5 0 0*], *vector*[*rat5 1 0*, *rat5* (−*1*) *0*, *rat5* (−*1*) *0*]},
convex hull {*vector*[*rat5* (−*1*/*2*) (*1*/*2*), *rat5* (−*1*/*2*) (−*1*/*2*), *rat5 0 0*], *vector*[*rat5 1 0*, *rat5* (−*1*) *0*, *rat5 1 0*]},
convex hull {*vector*[*rat5* (−*1*/*2*) (*1*/*2*), *rat5* (*1*/*2*) (*1*/*2*), *rat5 0 0*], *vector*[*rat5* (*1*/*2*) (−*1*/*2*), *rat5* (*1*/*2*) (*1*/*2*), *rat5 0 0*]},
convex hull {*vector*[*rat5* (−*1*/*2*) (*1*/*2*), *rat5* (*1*/*2*) (*1*/*2*), *rat5 0 0*], *vector*[*rat5 1 0*, *rat5 1 0*, *rat5* (−*1*) *0*]},
convex hull {*vector*[*rat5* (−*1*/*2*) (*1*/*2*), *rat5* (*1*/*2*) (*1*/*2*), *rat5 0 0*], *vector*[*rat5 1 0*, *rat5 1 0*, *rat5 1 0*]},

convex hull {*vector*[*rat5* (*1/2*) (−*1/2*), *rat5* (−*1/2*) (−*1/2*), *rat5 0 0*], *vector*[*rat5* (−*1*) *0*, *rat5* (−*1*) *0*, *rat5* (−*1*) *0*]},
convex hull {*vector*[*rat5* (*1/2*) (−*1/2*), *rat5* (−*1/2*) (−*1/2*), *rat5 0 0*], *vector*[*rat5* (−*1*) *0*, *rat5* (−*1*) *0*, *rat5 1 0*]},
convex hull {*vector*[*rat5* (*1/2*) (−*1/2*), *rat5* (*1/2*) (*1/2*), *rat5 0 0*], *vector*[*rat5* (−*1*) *0*, *rat5 1 0*, *rat5* (−*1*) *0*]},
convex hull {*vector*[*rat5* (*1/2*) (−*1/2*), *rat5* (*1/2*) (*1/2*), *rat5 0 0*], *vector*[*rat5* (−*1*) *0*, *rat5 1 0*, *rat5 1 0*]},
convex hull {*vector*[*rat5* (*1/2*) (*1/2*), *rat5 0 0*, *rat5* (−*1/2*) (*1/2*)], *vector*[*rat5* (*1/2*) (*1/2*), *rat5 0 0*, *rat5* (*1/2*) (−*1/2*)]},
convex hull {*vector*[*rat5* (*1/2*) (*1/2*), *rat5 0 0*, *rat5* (−*1/2*) (*1/2*)], *vector*[*rat5 1 0*, *rat5* (−*1*) *0*, *rat5 1 0*]},
convex hull {*vector*[*rat5* (*1/2*) (*1/2*), *rat5 0 0*, *rat5* (−*1/2*) (*1/2*)], *vector*[*rat5 1 0*, *rat5 1 0*, *rat5 1 0*]},
convex hull {*vector*[*rat5* (*1/2*) (*1/2*), *rat5 0 0*, *rat5* (*1/2*) (−*1/2*)], *vector*[*rat5 1 0*, *rat5* (−*1*) *0*, *rat5* (−*1*) *0*]},
convex hull {*vector*[*rat5* (*1/2*) (*1/2*), *rat5 0 0*, *rat5* (*1/2*) (−*1/2*)], *vector*[*rat5 1 0*, *rat5 1 0*, *rat5* (−*1*) *0*]},
convex hull {*vector*[*rat5* (−*1*) *0*, *rat5* (−*1*) *0*, *rat5* (−*1*) *0*], *vector*[*rat5 0 0*, *rat5* (*1/2*) (−*1/2*), *rat5* (−*1/2*) (−*1/2*)]},
convex hull {*vector*[*rat5* (−*1*) *0*, *rat5* (−*1*) *0*, *rat5 1 0*], *vector*[*rat5 0 0*, *rat5* (*1/2*) (−*1/2*), *rat5* (*1/2*) (*1/2*)]},
convex hull {*vector*[*rat5* (−*1*) *0*, *rat5 1 0*, *rat5* (−*1*) *0*], *vector*[*rat5 0 0*, *rat5* (−*1/2*) (*1/2*), *rat5* (−*1/2*) (−*1/2*)]},
convex hull {*vector*[*rat5* (−*1*) *0*, *rat5 1 0*, *rat5 1 0*], *vector*[*rat5 0 0*, *rat5* (−*1/2*) (*1/2*), *rat5* (*1/2*) (*1/2*)]},
convex hull {*vector*[*rat5 1 0*, *rat5* (−*1*) *0*, *rat5* (−*1*) *0*], *vector*[*rat5 0 0*, *rat5* (*1/2*) (−*1/2*), *rat5* (−*1/2*) (−*1/2*)]},
convex hull {*vector*[*rat5 1 0*, *rat5* (−*1*) *0*, *rat5 1 0*], *vector*[*rat5 0 0*, *rat5* (*1/2*) (−*1/2*), *rat5* (*1/2*) (*1/2*)]},
convex hull {*vector*[*rat5 1 0*, *rat5 1 0*, *rat5* (−*1*) *0*], *vector*[*rat5 0 0*, *rat5* (−*1/2*) (*1/2*), *rat5* (−*1/2*) (−*1/2*)]},
convex hull {*vector*[*rat5 1 0*, *rat5 1 0*, *rat5 1 0*], *vector*[*rat5 0 0*, *rat5* (−*1/2*) (*1/2*), *rat5* (*1/2*) (*1/2*)]},
convex hull {*vector*[*rat5 0 0*, *rat5* (−*1/2*) (*1/2*), *rat5* (−*1/2*) (−*1/2*)], *vector*[*rat5 0 0*, *rat5* (*1/2*) (−*1/2*), *rat5* (−*1/2*) (−*1/2*)]},
convex hull {*vector*[*rat5 0 0*, *rat5* (−*1/2*) (*1/2*), *rat5* (*1/2*) (*1/2*)], *vector*[*rat5 0 0*, *rat5* (*1/2*) (−*1/2*), *rat5* (*1/2*) (*1/2*)]}}
have *e1* ∈ *?edges e2* ∈ *?edges*
using *assms* **unfolding** *dodecahedron-edges*
by *auto*
moreover have ∀ *e*∈*?edges*.
convex hull {*vector*[*rat5* (−*1/2*) (−*1/2*), *rat5 0 0*, *rat5* (−*1/2*) (*1/2*)], *vector*[*rat5* (−*1/2*) (−*1/2*), *rat5 0 0*, *rat5* (*1/2*) (−*1/2*)]} *congruent e*
apply (*simp only*: *ball-simps*(*7*) *Set.ball-empty simp-thms*(*21*))
unfolding *conj-assoc*[*symmetric*] *segment-convex-hull*[*symmetric*]
apply (*rule conjI*)+
apply (*all* ‹*rule congruent-segments*›)
unfolding *dist-norm vector3-sub*

apply (*simp-all add*: *norm-eq rat5-sub del*: *cancel-comm-monoid-add-class.diff-cancel diff-self*)
by (*simp-all add*: *vector3-dot rat5-mul rat5-add rat5-eq*)
ultimately show *?thesis*
using *congruent-set* **by** *meson*
qed

lemma *dodecahedron-congruent-faces*:
assumes *f1 face-of std-dodecahedron* $\wedge$ *aff-dim f1* = *2*
and *f2 face-of std-dodecahedron* $\wedge$ *aff-dim f2* = *2*
shows *f1 congruent f2*
proof −
let *?faces* =
{*convex hull* {(*vector*[*rat5* (−*1*/*2*) (−*1*/*2*), *rat5 0 0*, *rat5* (−*1*/*2*) (*1*/*2*)]::*real^3*), *vector*[*rat5* (−*1*/*2*) (−*1*/*2*), *rat5 0 0*, *rat5* (*1*/*2*) (−*1*/*2*)], *vector*[*rat5* (*1*/*2*) (−*1*/*2*), *rat5* (−*1*/*2*) (−*1*/*2*), *rat5 0 0*], *vector*[*rat5* (−*1*) *0*, *rat5* (−*1*) *0*, *rat5* (−*1*) *0*], *vector*[*rat5* (−*1*) *0*, *rat5* (−*1*) *0*, *rat5 1 0*]},
convex hull {*vector*[*rat5* (−*1*/*2*) (−*1*/*2*), *rat5 0 0*, *rat5* (−*1*/*2*) (*1*/*2*)], *vector*[*rat5* (−*1*/*2*) (−*1*/*2*), *rat5 0 0*, *rat5* (*1*/*2*) (−*1*/*2*)], *vector*[*rat5* (*1*/*2*) (−*1*/*2*), *rat5* (*1*/*2*) (*1*/*2*), *rat5 0 0*], *vector*[*rat5* (−*1*) *0*, *rat5 1 0*, *rat5* (−*1*) *0*], *vector*[*rat5* (−*1*) *0*, *rat5 1 0*, *rat5 1 0*]},
convex hull {*vector*[*rat5* (−*1*/*2*) (−*1*/*2*), *rat5 0 0*, *rat5* (−*1*/*2*) (*1*/*2*)], *vector*[*rat5* (−*1*) *0*, *rat5* (−*1*) *0*, *rat5 1 0*], *vector*[*rat5* (−*1*) *0*, *rat5 1 0*, *rat5 1 0*], *vector*[*rat5 0 0*, *rat5* (−*1*/*2*) (*1*/*2*), *rat5* (*1*/*2*) (*1*/*2*)], *vector*[*rat5 0 0*, *rat5* (*1*/*2*) (−*1*/*2*), *rat5* (*1*/*2*) (*1*/*2*)]},
convex hull {*vector*[*rat5* (−*1*/*2*) (−*1*/*2*), *rat5 0 0*, *rat5* (*1*/*2*) (−*1*/*2*)], *vector*[*rat5* (−*1*) *0*, *rat5* (−*1*) *0*, *rat5* (−*1*) *0*], *vector*[*rat5* (−*1*) *0*, *rat5 1 0*, *rat5* (−*1*) *0*], *vector*[*rat5 0 0*, *rat5* (−*1*/*2*) (*1*/*2*), *rat5* (−*1*/*2*) (−*1*/*2*)], *vector*[*rat5 0 0*, *rat5* (*1*/*2*) (−*1*/*2*), *rat5* (−*1*/*2*) (−*1*/*2*)]},
convex hull {*vector*[*rat5* (−*1*/*2*) (*1*/*2*), *rat5* (−*1*/*2*) (−*1*/*2*), *rat5 0 0*], *vector*[*rat5* (*1*/*2*) (−*1*/*2*), *rat5* (−*1*/*2*) (−*1*/*2*), *rat5 0 0*], *vector*[*rat5* (−*1*) *0*, *rat5* (−*1*) *0*, *rat5* (−*1*) *0*], *vector*[*rat5 1 0*, *rat5* (−*1*) *0*, *rat5* (−*1*) *0*], *vector*[*rat5 0 0*, *rat5* (*1*/*2*) (−*1*/*2*), *rat5* (−*1*/*2*) (−*1*/*2*)]},
convex hull {*vector*[*rat5* (−*1*/*2*) (*1*/*2*), *rat5* (−*1*/*2*) (−*1*/*2*), *rat5 0 0*], *vector*[*rat5* (*1*/*2*) (−*1*/*2*), *rat5* (−*1*/*2*) (−*1*/*2*), *rat5 0 0*], *vector*[*rat5* (−*1*) *0*, *rat5* (−*1*) *0*, *rat5 1 0*], *vector*[*rat5 1 0*, *rat5* (−*1*) *0*, *rat5 1 0*], *vector*[*rat5 0 0*, *rat5* (*1*/*2*) (−*1*/*2*), *rat5* (*1*/*2*) (*1*/*2*)]},
convex hull {*vector*[*rat5* (−*1*/*2*) (*1*/*2*), *rat5* (−*1*/*2*) (−*1*/*2*), *rat5 0 0*], *vector*[*rat5* (*1*/*2*) (*1*/*2*), *rat5 0 0*, *rat5* (−*1*/*2*) (*1*/*2*)], *vector*[*rat5* (*1*/*2*) (*1*/*2*), *rat5 0 0*, *rat5* (*1*/*2*) (−*1*/*2*)], *vector*[*rat5 1 0*, *rat5* (−*1*) *0*, *rat5* (−*1*) *0*], *vector*[*rat5 1 0*, *rat5* (−*1*) *0*, *rat5 1 0*]},
convex hull {*vector*[*rat5* (−*1*/*2*) (*1*/*2*), *rat5* (*1*/*2*) (*1*/*2*), *rat5 0 0*], *vector*[*rat5* (*1*/*2*) (−*1*/*2*), *rat5* (*1*/*2*) (*1*/*2*), *rat5 0 0*], *vector*[*rat5* (−*1*) *0*, *rat5 1 0*, *rat5* (−*1*) *0*], *vector*[*rat5 1 0*, *rat5 1 0*, *rat5* (−*1*) *0*], *vector*[*rat5 0 0*, *rat5* (−*1*/*2*) (*1*/*2*), *rat5* (−*1*/*2*) (−*1*/*2*)]},
convex hull {*vector*[*rat5* (−*1*/*2*) (*1*/*2*), *rat5* (*1*/*2*) (*1*/*2*), *rat5 0 0*], *vector*[*rat5* (*1*/*2*) (−*1*/*2*), *rat5* (*1*/*2*) (*1*/*2*), *rat5 0 0*], *vector*[*rat5* (−*1*) *0*, *rat5 1 0*, *rat5 1 0*], *vector*[*rat5 1 0*, *rat5 1 0*, *rat5 1 0*], *vector*[*rat5 0 0*, *rat5* (−*1*/*2*) (*1*/*2*), *rat5* (*1*/*2*) (*1*/*2*)]},

convex hull {*vector*[*rat5* (−*1*/*2*) (*1*/*2*), *rat5* (*1*/*2*) (*1*/*2*), *rat5 0 0*], *vector*[*rat5* (*1*/*2*) (*1*/*2*), *rat5 0 0*, *rat5* (−*1*/*2*) (*1*/*2*)], *vector*[*rat5* (*1*/*2*) (*1*/*2*), *rat5 0 0*, *rat5* (*1*/*2*) (−*1*/*2*)], *vector*[*rat5 1 0*, *rat5 1 0*, *rat5* (−*1*) *0*], *vector*[*rat5 1 0*, *rat5 1 0*, *rat5 1 0*]},

convex hull {*vector*[*rat5* (*1*/*2*) (*1*/*2*), *rat5 0 0*, *rat5* (−*1*/*2*) (*1*/*2*)], *vector*[*rat5 1 0*, *rat5* (−*1*) *0*, *rat5 1 0*], *vector*[*rat5 1 0*, *rat5 1 0*, *rat5 1 0*], *vector*[*rat5 0 0*, *rat5* (−*1*/*2*) (*1*/*2*), *rat5* (*1*/*2*) (*1*/*2*)], *vector*[*rat5 0 0*, *rat5* (*1*/*2*) (−*1*/*2*), *rat5* (*1*/*2*) (*1*/*2*)]},

convex hull {*vector*[*rat5* (*1*/*2*) (*1*/*2*), *rat5 0 0*, *rat5* (*1*/*2*) (−*1*/*2*)], *vector*[*rat5 1 0*, *rat5* (−*1*) *0*, *rat5* (−*1*) *0*], *vector*[*rat5 1 0*, *rat5 1 0*, *rat5* (−*1*) *0*], *vector*[*rat5 0 0*, *rat5* (−*1*/*2*) (*1*/*2*), *rat5* (−*1*/*2*) (−*1*/*2*)], *vector*[*rat5 0 0*, *rat5* (*1*/*2*) (−*1*/*2*), *rat5* (−*1*/*2*) (−*1*/*2*)]}}

```
  have f1 ∈ ?faces f2 ∈ ?faces
    using assms dodecahedron-facets
    by auto
  moreover have ∀f∈?faces.
```

convex hull {*vector*[*rat5* (−*1*/*2*) (−*1*/*2*), *rat5 0 0*, *rat5* (−*1*/*2*) (*1*/*2*)], *vector*[*rat5* (−*1*/*2*) (−*1*/*2*), *rat5 0 0*, *rat5* (*1*/*2*) (−*1*/*2*)], *vector*[*rat5* (*1*/*2*) (−*1*/*2*), *rat5* (−*1*/*2*) (−*1*/*2*), *rat5 0 0*], *vector*[*rat5* (−*1*) *0*, *rat5* (−*1*) *0*, *rat5* (−*1*) *0*], *vector*[*rat5* (−*1*) *0*, *rat5* (−*1*) *0*, *rat5 1 0*]} *congruent f*

```
    apply (simp only: ball-simps(7) Set.ball-empty simp-thms(21))
    unfolding conj-assoc[symmetric]
    by (rule conjI)+ show-congruence+
  ultimately show ?thesis
    using congruent-set by meson
qed

lemma dodecahedron-facet1-equiangular:
  defines v1 ≡ vector[rat5 (−1/2) (−1/2), rat5 0 0, rat5 (−1/2) (1/2)] :: real^3
  defines v2 ≡ vector[rat5 (−1/2) (−1/2), rat5 0 0, rat5 (1/2) (−1/2)] :: real^3
  defines v3 ≡ vector[rat5 (1/2) (−1/2), rat5 (−1/2) (−1/2), rat5 0 0] :: real^3
  defines v4 ≡ vector[rat5 (−1) 0, rat5 (−1) 0, rat5 (−1) 0] :: real^3
  defines v5 ≡ vector[rat5 (−1) 0, rat5 (−1) 0, rat5 1 0] :: real^3
  shows equiangular (convex hull {v1, v2, v3, v4, v5}) (3 ∗ pi / 5)
  unfolding equiangular-def
proof (clarify)
  fix e1 e2 v
  assume ∗: e1 face-of convex hull {v1, v2, v3, v4, v5} aff-dim e1 = 1
           e2 face-of convex hull {v1, v2, v3, v4, v5} aff-dim e2 = 1
           e1 ≠ e2 v extreme-point-of e1 v extreme-point-of e2
  have arccos (1/4 − 1/4 ∗ sqrt 5) = 3 ∗ pi / 5
  proof (rule arccos-unique)
    show 0 ≤ 3 ∗ pi / 5 by simp
    show 3 ∗ pi / 5 ≤ pi by simp

    define t where t = 3 ∗ pi / 5
    define c where c = cos t
    define s where s = sin t
```

have *cos* $(3 * t) = cos (2 * t + t)$ **by** *simp*
also have ... $= (2 * c\hat{}2 - 1) * c - 2 * s\hat{}2 * c$
unfolding *cos-add c-def s-def*
by (*simp add*: *cos-double-cos power2-eq-square sin-double*)
finally have *cos-3t*: $cos (3 * t) = 4 * c\hat{}3 - 3 * c$
unfolding *sin-squared-eq c-def s-def* **by** *algebra*

have $sin (3 * t) = sin (2 * t + t)$ **by** *simp*
also have ... $= 2 * s * c\hat{}2 + (1 - 2 * s\hat{}2) * s$
unfolding *sin-add c-def s-def*
by (*simp add*: *cos-double-sin power2-eq-square sin-double*)
finally have *sin-3t*: $sin (3 * t) = 3 * s - 4 * s\hat{}3$
unfolding *cos-squared-eq c-def s-def* **by** *algebra*

have $cos (5 * t) = cos (3 * t + 2 * t)$ **by** *simp*
also have ... $= cos (3 * t) * cos (2 * t) - sin (3 * t) * sin (2 * t)$
using *cos-add* **by** *blast*
also have ... $= (4 * c\hat{}3 - 3 * c) * (2 * c\hat{}2 - 1) - (3 * s - 4 * s\hat{}3) * 2 * s * c$
by (*simp add*: *cos-3t sin-3t c-def s-def cos-double-cos sin-double*)
also have ... $= (4 * c\hat{}3 - 3 * c) * (2 * c\hat{}2 - 1) - 6 * s\hat{}2 * c + 8 * (s\hat{}2)\hat{}2 * c$
by *algebra*
also have ... $= (4 * c\hat{}3 - 3 * c) * (2 * c\hat{}2 - 1) - 6 * (1 - c\hat{}2) * c + 8 * (1 - c\hat{}2)\hat{}2 * c$
unfolding *sin-squared-eq c-def s-def* **by** *blast*
finally have $cos (5 * t) = 16 * c\hat{}5 - 20 * c\hat{}3 + 5 * c$
by *algebra*
moreover have $cos (5 * t) = -1$
unfolding *t-def* **by** *simp*
ultimately have $16 * c\hat{}5 - 20 * c\hat{}3 + 5 * c + 1 = 0$
by *simp*
then have $(c + 1) * (4 * c\hat{}2 - 2 * c - 1)\hat{}2 = 0$
by *algebra*
then have $c + 1 = 0 \vee 4 * c\hat{}2 - 2 * c - 1 = 0$
by *simp*

have $\exists!x.\ 0 \leq x \wedge x \leq pi \wedge cos\ x = -\ 1$
using *cos-total*[*of* -1] **by** *linarith*
moreover have $0 \leq pi \wedge pi \leq pi \wedge cos\ pi = -1$
using *cos-pi* **by** *simp*
moreover have $3 * pi / 5 \neq pi\ 0 \leq 3 * pi / 5 \wedge 3 * pi / 5 \leq pi$
by *auto*
ultimately have $c \neq -1$
unfolding *c-def t-def* **by** *blast*
then have $c + 1 \neq 0$
by *linarith*

then have $4 * c\hat{}2 + (-2) * c + (-1) = 0$

using ‹$c + 1 = 0 \lor 4 * c\hat{\ }2 - 2 * c - 1 = 0$›
by *simp*
moreover have *discrim* 4 (-2) $(-1) = 20$
unfolding *discrim-def* **by** *simp*
moreover have *sqrt* $20 = 2 *$ *sqrt* 5
by (*metis numeral-times-numeral real-sqrt-four real-sqrt-mult semiring-norm*(12,14))
ultimately have ($c = (2 + 2 *$ *sqrt* $5) / (2 * 4) \lor$
$c = (2 - 2 *$ *sqrt* $5) / (2 * 4)$)
using *discriminant-iff*[*of* 4 *c* −2 −1]
by *auto*
then have $c = 1/4 + 1/4 *$ *sqrt* $5 \lor c = 1/4 - 1/4 *$ *sqrt* 5
by *fastforce*
moreover have $c < 0$
unfolding *c-def t-def* **using** *cos-lt-zero-pi*[*of* 3 ∗ *pi* / 5]
by *simp*
ultimately show *cos* $(3 * pi / 5) = 1 / 4 - 1 / 4 *$ *sqrt* 5
unfolding *c-def t-def*
by (*metis add-nonneg-nonneg less-le not-le real-rat5-le-sqrt-cases zero-less-divide-1-iff zero-less-numeral*)
qed
then have *arccos-3pi5*: *arccos* (*rat5* (1/4) (−(1/4))) = 3 ∗ *pi* / 5 **unfolding** *rat5-def*
by (*simp add*: *of-rat-divide of-rat-minus*)
have *rat5* 6 (− 2) ≥ 0
unfolding *rat5-0-1*(1)[*symmetric*] *rat5-le*
by *simp*
have ($v = v1 \land$ ((*e1* = *convex hull* {*v1*, *v2*} ∧ *e2* = *convex hull* {*v1*, *v5*}) ∨
(*e1* = *convex hull* {*v1*, *v5*} ∧ *e2* = *convex hull* {*v1*, *v2*}))) ∨
($v = v2 \land$ ((*e1* = *convex hull* {*v2*, *v1*} ∧ *e2* = *convex hull* {*v2*, *v4*}) ∨
(*e1* = *convex hull* {*v2*, *v4*} ∧ *e2* = *convex hull* {*v2*, *v1*}))) ∨
($v = v3 \land$ ((*e1* = *convex hull* {*v3*, *v4*} ∧ *e2* = *convex hull* {*v3*, *v5*}) ∨
(*e1* = *convex hull* {*v3*, *v5*} ∧ *e2* = *convex hull* {*v3*, *v4*}))) ∨
($v = v4 \land$ ((*e1* = *convex hull* {*v4*, *v2*} ∧ *e2* = *convex hull* {*v4*, *v3*}) ∨
(*e1* = *convex hull* {*v4*, *v3*} ∧ *e2* = *convex hull* {*v4*, *v2*}))) ∨
($v = v5 \land$ ((*e1* = *convex hull* {*v5*, *v1*} ∧ *e2* = *convex hull* {*v5*, *v3*}) ∨
(*e1* = *convex hull* {*v5*, *v3*} ∧ *e2* = *convex hull* {*v5*, *v1*})))
using *conjI*[*OF* ∗(1,2)] *conjI*[*OF* ∗(3,4)] ∗(5−7) **unfolding** *assms dodecahedron-facet1-edges*
by (*smt* (*verit*) *divide-cancel-right divide-eq-0-iff extreme-point-of-convex-hull-2 insert-commute*
rat5-eq solids-vectors-to-rat5(10) *vector3-neq*)
moreover have *vangle* ($v2 - v1$) ($v5 - v1$) = 3 ∗ *pi* / 5
vangle ($v1 - v2$) ($v4 - v2$) = 3 ∗ *pi* / 5
vangle ($v4 - v3$) ($v5 - v3$) = 3 ∗ *pi* / 5
vangle ($v2 - v4$) ($v3 - v4$) = 3 ∗ *pi* / 5
vangle ($v1 - v5$) ($v3 - v5$) = 3 ∗ *pi* / 5
unfolding *assms vector3-sub norm-eq-sqrt-inner vector3-dot vangle-def rat5-sub rat5-mul*
by (*simp-all add*: ‹*rat5* 6 (− 2) ≥ 0› *arccos-3pi5 vector3-eq-0 rat5-sub rat5-mul*

rat5-add rat5-eq-0 rat5-div)
ultimately show ($\exists$ *a b. e1* = *convex hull* {*v*, *a*} $\wedge$ *e2* = *convex hull* {*v*, *b*} $\wedge$
vangle (*a* − *v*) (*b* − *v*) = *3* ∗ *pi* / *5*)
using *vangle-commute*
by *metis*
qed

lemma *dodecahedron-equiangular*:
assumes *f face-of std-dodecahedron* $\wedge$ *aff-dim f* = *2*
shows *equiangular f* (*3* ∗ *pi* / *5*)
proof −
define *f1* **where** *f1* = *convex hull* {(*vector*[*rat5* (−*1*/*2*) (−*1*/*2*), *rat5 0 0*, *rat5* (−*1*/*2*) (*1*/*2*)]::*real*^*3*), *vector*[*rat5* (−*1*/*2*) (−*1*/*2*), *rat5 0 0*, *rat5* (*1*/*2*) (−*1*/*2*)], *vector*[*rat5* (*1*/*2*) (−*1*/*2*), *rat5* (−*1*/*2*) (−*1*/*2*), *rat5 0 0*], *vector*[*rat5* (−*1*) *0*, *rat5* (−*1*) *0*, *rat5* (−*1*) *0*], *vector*[*rat5* (−*1*) *0*, *rat5* (−*1*) *0*, *rat5 1 0*]}
have *f1 face-of std-dodecahedron* $\wedge$ *aff-dim f1* = *2*
using *f1-def dodecahedron-facets* **by** *blast*
then have *f1 congruent f*
using *assms dodecahedron-congruent-faces* **by** *blast*
then show *?thesis*
using *equiangular-congruent dodecahedron-facet1-equiangular f1-def*
by *blast*
qed

end

9 Icosahedron

theory *Icosahedron*
imports
Computation
begin

9.1 Definition

definition *std-icosahedron* :: (*real*, *3*) *vec set*
where *std-icosahedron* ≡ *convex hull*
{*vector* [*0*, *1*, φ], *vector* [*0*, *1*, −φ],
vector [*0*, −*1*, φ], *vector* [*0*, −*1*, −φ],
vector [*1*, φ, *0*], *vector* [*1*, −φ, *0*],
vector [−*1*, φ, *0*], *vector* [−*1*, −φ, *0*],
vector [φ, *0*, *1*], *vector* [−φ, *0*, *1*],
vector [φ, *0*, −*1*], *vector* [−φ, *0*, −*1*]}

lemma *std-icosahedron-eq*:
shows *std-icosahedron* = *convex hull*
{ *vector*[*rat5 0 0*, *rat5 1 0*, *rat5* (*1*/*2*) (*1*/*2*)],
vector[*rat5 0 0*, *rat5 1 0*, *rat5* (−*1*/*2*) (−*1*/*2*)],
vector[*rat5 0 0*, *rat5* (−*1*) *0*, *rat5* (*1*/*2*) (*1*/*2*)],

vector[*rat5 0 0*, *rat5* (−*1*) *0*, *rat5* (−*1*/*2*) (−*1*/*2*)],
vector[*rat5 1 0*, *rat5* (*1*/*2*) (*1*/*2*), *rat5 0 0*],
vector[*rat5 1 0*, *rat5* (−*1*/*2*) (−*1*/*2*), *rat5 0 0*],
vector[*rat5* (−*1*) *0*, *rat5* (*1*/*2*) (*1*/*2*), *rat5 0 0*],
vector[*rat5* (−*1*) *0*, *rat5* (−*1*/*2*) (−*1*/*2*), *rat5 0 0*],
vector[*rat5* (*1*/*2*) (*1*/*2*), *rat5 0 0*, *rat5 1 0*],
vector[*rat5* (−*1*/*2*) (−*1*/*2*), *rat5 0 0*, *rat5 1 0*],
vector[*rat5* (*1*/*2*) (*1*/*2*), *rat5 0 0*, *rat5* (−*1*) *0*],
vector[*rat5* (−*1*/*2*) (−*1*/*2*), *rat5 0 0*, *rat5* (−*1*) *0*]}
unfolding *std-icosahedron-def rat5-def*
by (*simp add*: *add-divide-distrib of-rat-divide of-rat-minus*)

lemma *icosahedron-fulldim*:
shows *aff-dim std-icosahedron* = *3*
unfolding *std-icosahedron-def*
apply (*rule polytope-3D*)
by (*simp add*: *vector3-sub vector3-cross vector3-dot vector3-eq-0*,
smt (*verit*, *ccfv-threshold*) *real-sqrt-ge-0-iff*)+

lemma *icosahedron-polyhedron*:
shows *polyhedron std-icosahedron*
by (*simp add*: *std-icosahedron-def polytope-convex-hull polytope-imp-polyhedron*)

9.2 Facets, edges, and vertices

lemma *icosahedron-facets*:
fixes *f* :: (*real*^*3*) *set*
shows *f face-of std-icosahedron* ∧ *aff-dim f* = *2* ⟷
f = *convex hull* {*vector*[*rat5* (−*1*/*2*) (−*1*/*2*), *rat5 0 0*, *rat5* (−*1*) *0*], *vector*[*rat5* (−*1*/*2*) (−*1*/*2*), *rat5 0 0*, *rat5 1 0*], *vector*[*rat5* (−*1*) *0*, *rat5* (−*1*/*2*) (−*1*/*2*), *rat5 0 0*]} ∨
f = *convex hull* {*vector*[*rat5* (−*1*/*2*) (−*1*/*2*), *rat5 0 0*, *rat5* (−*1*) *0*], *vector*[*rat5* (−*1*/*2*) (−*1*/*2*), *rat5 0 0*, *rat5 1 0*], *vector*[*rat5* (−*1*) *0*, *rat5* (*1*/*2*) (*1*/*2*), *rat5 0 0*]} ∨
f = *convex hull* {*vector*[*rat5* (−*1*/*2*) (−*1*/*2*), *rat5 0 0*, *rat5* (−*1*) *0*], *vector*[*rat5* (−*1*) *0*, *rat5* (−*1*/*2*) (−*1*/*2*), *rat5 0 0*], *vector*[*rat5 0 0*, *rat5* (−*1*) *0*, *rat5* (−*1*/*2*) (−*1*/*2*)]} ∨
f = *convex hull* {*vector*[*rat5* (−*1*/*2*) (−*1*/*2*), *rat5 0 0*, *rat5* (−*1*) *0*], *vector*[*rat5* (−*1*) *0*, *rat5* (*1*/*2*) (*1*/*2*), *rat5 0 0*], *vector*[*rat5 0 0*, *rat5 1 0*, *rat5* (−*1*/*2*) (−*1*/*2*)]} ∨
f = *convex hull* {*vector*[*rat5* (−*1*/*2*) (−*1*/*2*), *rat5 0 0*, *rat5* (−*1*) *0*], *vector*[*rat5 0 0*, *rat5* (−*1*) *0*, *rat5* (−*1*/*2*) (−*1*/*2*)], *vector*[*rat5 0 0*, *rat5 1 0*, *rat5* (−*1*/*2*) (−*1*/*2*)]} ∨
f = *convex hull* {*vector*[*rat5* (−*1*/*2*) (−*1*/*2*), *rat5 0 0*, *rat5 1 0*], *vector*[*rat5* (−*1*) *0*, *rat5* (−*1*/*2*) (−*1*/*2*), *rat5 0 0*], *vector*[*rat5 0 0*, *rat5* (−*1*) *0*, *rat5* (*1*/*2*) (*1*/*2*)]} ∨
f = *convex hull* {*vector*[*rat5* (−*1*/*2*) (−*1*/*2*), *rat5 0 0*, *rat5 1 0*], *vector*[*rat5* (−*1*) *0*, *rat5* (*1*/*2*) (*1*/*2*), *rat5 0 0*], *vector*[*rat5 0 0*, *rat5 1 0*, *rat5* (*1*/*2*) (*1*/*2*)]} ∨

f = *convex hull* {*vector*[*rat5* (−*1*/*2*) (−*1*/*2*), *rat5 0 0*, *rat5 1 0*], *vector*[*rat5 0 0*, *rat5* (−*1*) *0*, *rat5* (*1*/*2*) (*1*/*2*)], *vector*[*rat5 0 0*, *rat5 1 0*, *rat5* (*1*/*2*) (*1*/*2*)]} ∨

f = *convex hull* {*vector*[*rat5* (*1*/*2*) (*1*/*2*), *rat5 0 0*, *rat5* (−*1*) *0*], *vector*[*rat5* (*1*/*2*) (*1*/*2*), *rat5 0 0*, *rat5 1 0*], *vector*[*rat5 1 0*, *rat5* (−*1*/*2*) (−*1*/*2*), *rat5 0 0*]} ∨

f = *convex hull* {*vector*[*rat5* (*1*/*2*) (*1*/*2*), *rat5 0 0*, *rat5* (−*1*) *0*], *vector*[*rat5* (*1*/*2*) (*1*/*2*), *rat5 0 0*, *rat5 1 0*], *vector*[*rat5 1 0*, *rat5* (*1*/*2*) (*1*/*2*), *rat5 0 0*]} ∨

f = *convex hull* {*vector*[*rat5* (*1*/*2*) (*1*/*2*), *rat5 0 0*, *rat5* (−*1*) *0*], *vector*[*rat5 1 0*, *rat5* (−*1*/*2*) (−*1*/*2*), *rat5 0 0*], *vector*[*rat5 0 0*, *rat5* (−*1*) *0*, *rat5* (−*1*/*2*) (−*1*/*2*)]} ∨

f = *convex hull* {*vector*[*rat5* (*1*/*2*) (*1*/*2*), *rat5 0 0*, *rat5* (−*1*) *0*], *vector*[*rat5 1 0*, *rat5* (*1*/*2*) (*1*/*2*), *rat5 0 0*], *vector*[*rat5 0 0*, *rat5 1 0*, *rat5* (−*1*/*2*) (−*1*/*2*)]} ∨

f = *convex hull* {*vector*[*rat5* (*1*/*2*) (*1*/*2*), *rat5 0 0*, *rat5* (−*1*) *0*], *vector*[*rat5 0 0*, *rat5* (−*1*) *0*, *rat5* (−*1*/*2*) (−*1*/*2*)], *vector*[*rat5 0 0*, *rat5 1 0*, *rat5* (−*1*/*2*) (−*1*/*2*)]} ∨

f = *convex hull* {*vector*[*rat5* (*1*/*2*) (*1*/*2*), *rat5 0 0*, *rat5 1 0*], *vector*[*rat5 1 0*, *rat5* (−*1*/*2*) (−*1*/*2*), *rat5 0 0*], *vector*[*rat5 0 0*, *rat5* (−*1*) *0*, *rat5* (*1*/*2*) (*1*/*2*)]} ∨

f = *convex hull* {*vector*[*rat5* (*1*/*2*) (*1*/*2*), *rat5 0 0*, *rat5 1 0*], *vector*[*rat5 1 0*, *rat5* (*1*/*2*) (*1*/*2*), *rat5 0 0*], *vector*[*rat5 0 0*, *rat5 1 0*, *rat5* (*1*/*2*) (*1*/*2*)]} ∨

f = *convex hull* {*vector*[*rat5* (*1*/*2*) (*1*/*2*), *rat5 0 0*, *rat5 1 0*], *vector*[*rat5 0 0*, *rat5* (−*1*) *0*, *rat5* (*1*/*2*) (*1*/*2*)], *vector*[*rat5 0 0*, *rat5 1 0*, *rat5* (*1*/*2*) (*1*/*2*)]} ∨

f = *convex hull* {*vector*[*rat5* (−*1*) *0*, *rat5* (−*1*/*2*) (−*1*/*2*), *rat5 0 0*], *vector*[*rat5 1 0*, *rat5* (−*1*/*2*) (−*1*/*2*), *rat5 0 0*], *vector*[*rat5 0 0*, *rat5* (−*1*) *0*, *rat5* (−*1*/*2*) (−*1*/*2*)]} ∨

f = *convex hull* {*vector*[*rat5* (−*1*) *0*, *rat5* (−*1*/*2*) (−*1*/*2*), *rat5 0 0*], *vector*[*rat5 1 0*, *rat5* (−*1*/*2*) (−*1*/*2*), *rat5 0 0*], *vector*[*rat5 0 0*, *rat5* (−*1*) *0*, *rat5* (*1*/*2*) (*1*/*2*)]} ∨

f = *convex hull* {*vector*[*rat5* (−*1*) *0*, *rat5* (*1*/*2*) (*1*/*2*), *rat5 0 0*], *vector*[*rat5 1 0*, *rat5* (*1*/*2*) (*1*/*2*), *rat5 0 0*], *vector*[*rat5 0 0*, *rat5 1 0*, *rat5* (−*1*/*2*) (−*1*/*2*)]} ∨

f = *convex hull* {*vector*[*rat5* (−*1*) *0*, *rat5* (*1*/*2*) (*1*/*2*), *rat5 0 0*], *vector*[*rat5 1 0*, *rat5* (*1*/*2*) (*1*/*2*), *rat5 0 0*], *vector*[*rat5 0 0*, *rat5 1 0*, *rat5* (*1*/*2*) (*1*/*2*)]}

unfolding *std-icosahedron-eq*
apply (*simp only*: *compute-faces-2 finite-insert finite.emptyI*)
apply (*simp only*: *compute-faces-2-step-1 bex-empty simp-thms*(*31*))
apply (*simp only*: *compute-faces-2-step-2 bex-empty simp-thms*(*31*))
apply (*simp only*: *empty-iff ball-insert bex-empty bex-simps*(*5*) *ball-simps*(*5*) *simp-thms*(*21*,*31*))
apply (*simp only*: *vector3-sub vector3-cross vector3-eq-0 rat5-add rat5-sub rat5-mul rat5-eq*)
apply (*simp add*: *vector3-eq-0 vector3-dot rat5-add rat5-sub rat5-mul rat5-le rat5-eq rat5-eq-0 Let-def*)
— very slow, ca 60s
apply (*simp only*: *insert-commute*)
by *linarith*

lemma *icosahedron-facet1-edges*:
  defines $v1 \equiv vector[rat5\ (-1/2)\ (-1/2),\ rat5\ 0\ 0,\ rat5\ (-1)\ 0] :: real\hat{}3$
    and $v2 \equiv vector[rat5\ (-1/2)\ (-1/2),\ rat5\ 0\ 0,\ rat5\ 1\ 0] :: real\hat{}3$
    and $v3 \equiv vector[rat5\ (-1)\ 0,\ rat5\ (-1/2)\ (-1/2),\ rat5\ 0\ 0] :: real\hat{}3$
    and $n \equiv vector[rat5\ 1\ 1,\ rat5\ (-1)\ 1,\ rat5\ 0\ 0] :: real\hat{}3$
   shows $(e\ face\text{-}of\ (convex\ hull\ \{v1,\ v2,\ v3\}) \wedge aff\text{-}dim\ e = 1) =$
      $(e = convex\ hull\ \{v1,\ v2\} \vee e = convex\ hull\ \{v1,\ v3\} \vee e = convex\ hull\ \{v2,\ v3\})$
proof –
  have *n-nz*: $n \neq 0$
   unfolding *n-def* **by** (*simp add*: *vector3-eq-0 rat5-eq-0*)
  have *plane*: $\bigwedge x.\ x \in \{v1,\ v2,\ v3\} \Longrightarrow n \cdot x = rat5\ (-3)\ (-1)$
   unfolding *v1-def v2-def v3-def n-def*
   by (*auto simp add*: *vector3-dot rat5-mul rat5-add*)
  have *finite*: *finite* $\{v1,\ v2,\ v3\}$
   using *assms* **by** *blast*
  show *?thesis*
   by (*facet-edges plane*: *plane n-nz*: *n-nz defs*: *assms*)
qed

lemma *icosahedron-facet2-edges*:
  defines $v1 \equiv vector[rat5\ (-1/2)\ (-1/2),\ rat5\ 0\ 0,\ rat5\ (-1)\ 0] :: real\hat{}3$
    and $v2 \equiv vector[rat5\ (-1/2)\ (-1/2),\ rat5\ 0\ 0,\ rat5\ 1\ 0] :: real\hat{}3$
    and $v3 \equiv vector[rat5\ (-1)\ 0,\ rat5\ (1/2)\ (1/2),\ rat5\ 0\ 0] :: real\hat{}3$
    and $n \equiv vector[rat5\ (-1)\ (-1),\ rat5\ (-1)\ 1,\ rat5\ 0\ 0] :: real\hat{}3$
   shows $(e\ face\text{-}of\ (convex\ hull\ \{v1,\ v2,\ v3\}) \wedge aff\text{-}dim\ e = 1) =$
      $(e = convex\ hull\ \{v1,\ v2\} \vee e = convex\ hull\ \{v1,\ v3\} \vee e = convex\ hull\ \{v2,\ v3\})$
proof –
  have *n-nz*: $n \neq 0$
   unfolding *n-def* **by** (*simp add*: *vector3-eq-0 rat5-eq-0*)
  have *plane*: $\bigwedge x.\ x \in \{v1,\ v2,\ v3\} \Longrightarrow n \cdot x = rat5\ 3\ 1$
   unfolding *v1-def v2-def v3-def n-def*
   by (*auto simp add*: *vector3-dot rat5-mul rat5-add*)
  have *finite*: *finite* $\{v1,\ v2,\ v3\}$
   using *assms* **by** *blast*
  show *?thesis*
   by (*facet-edges plane*: *plane n-nz*: *n-nz defs*: *assms*)
qed

lemma *icosahedron-facet3-edges*:
  defines $v1 \equiv vector[rat5\ (-1/2)\ (-1/2),\ rat5\ 0\ 0,\ rat5\ (-1)\ 0] :: real\hat{}3$
    and $v2 \equiv vector[rat5\ (-1)\ 0,\ rat5\ (-1/2)\ (-1/2),\ rat5\ 0\ 0] :: real\hat{}3$
    and $v3 \equiv vector[rat5\ 0\ 0,\ rat5\ (-1)\ 0,\ rat5\ (-1/2)\ (-1/2)] :: real\hat{}3$
    and $n \equiv vector[rat5\ 2\ 0,\ rat5\ 2\ 0,\ rat5\ 2\ 0] :: real\hat{}3$
   shows $(e\ face\text{-}of\ (convex\ hull\ \{v1,\ v2,\ v3\}) \wedge aff\text{-}dim\ e = 1) =$
      $(e = convex\ hull\ \{v1,\ v2\} \vee e = convex\ hull\ \{v1,\ v3\} \vee e = convex\ hull\ \{v2,\ v3\})$

proof −
  have *n-nz*: $n \neq 0$
    unfolding *n-def* **by** (*simp add*: *vector3-eq-0 rat5-eq-0*)
  have *plane*: $\bigwedge x.\ x \in \{v1,\ v2,\ v3\} \Longrightarrow n \cdot x = rat5\ (-3)\ (-1)$
    unfolding *v1-def v2-def v3-def n-def*
    by (*auto simp add*: *vector3-dot rat5-mul rat5-add*)
  have *finite*: *finite* $\{v1,\ v2,\ v3\}$
    using *assms* **by** *blast*
  show *?thesis*
    by (*facet-edges plane*: *plane n-nz*: *n-nz defs*: *assms*)
qed

lemma *icosahedron-facet4-edges*:
  defines $v1 \equiv vector[rat5\ (-1/2)\ (-1/2),\ rat5\ 0\ 0,\ rat5\ (-1)\ 0] :: real\hat{}3$
    and $v2 \equiv vector[rat5\ (-1)\ 0,\ rat5\ (1/2)\ (1/2),\ rat5\ 0\ 0] :: real\hat{}3$
    and $v3 \equiv vector[rat5\ 0\ 0,\ rat5\ 1\ 0,\ rat5\ (-1/2)\ (-1/2)] :: real\hat{}3$
    and $n \equiv vector[rat5\ (-2)\ 0,\ rat5\ 2\ 0,\ rat5\ (-2)\ 0] :: real\hat{}3$
  shows (e *face-of* (*convex hull* $\{v1,\ v2,\ v3\}) \wedge$ *aff-dim* $e = 1) =$
    ($e =$ *convex hull* $\{v1,\ v2\} \vee e =$ *convex hull* $\{v1,\ v3\} \vee e =$ *convex hull* $\{v2,\ v3\})$
proof −
  have *n-nz*: $n \neq 0$
    unfolding *n-def* **by** (*simp add*: *vector3-eq-0 rat5-eq-0*)
  have *plane*: $\bigwedge x.\ x \in \{v1,\ v2,\ v3\} \Longrightarrow n \cdot x = rat5\ 3\ 1$
    unfolding *v1-def v2-def v3-def n-def*
    by (*auto simp add*: *vector3-dot rat5-mul rat5-add*)
  have *finite*: *finite* $\{v1,\ v2,\ v3\}$
    using *assms* **by** *blast*
  show *?thesis*
    by (*facet-edges plane*: *plane n-nz*: *n-nz defs*: *assms*)
qed

lemma *icosahedron-facet5-edges*:
  defines $v1 \equiv vector[rat5\ (-1/2)\ (-1/2),\ rat5\ 0\ 0,\ rat5\ (-1)\ 0] :: real\hat{}3$
    and $v2 \equiv vector[rat5\ 0\ 0,\ rat5\ (-1)\ 0,\ rat5\ (-1/2)\ (-1/2)] :: real\hat{}3$
    and $v3 \equiv vector[rat5\ 0\ 0,\ rat5\ 1\ 0,\ rat5\ (-1/2)\ (-1/2)] :: real\hat{}3$
    and $n \equiv vector[rat5\ (-1)\ 1,\ rat5\ 0\ 0,\ rat5\ 1\ 1] :: real\hat{}3$
  shows (e *face-of* (*convex hull* $\{v1,\ v2,\ v3\}) \wedge$ *aff-dim* $e = 1) =$
    ($e =$ *convex hull* $\{v1,\ v2\} \vee e =$ *convex hull* $\{v1,\ v3\} \vee e =$ *convex hull* $\{v2,\ v3\})$
proof −
  have *n-nz*: $n \neq 0$
    unfolding *n-def* **by** (*simp add*: *vector3-eq-0 rat5-eq-0*)
  have *plane*: $\bigwedge x.\ x \in \{v1,\ v2,\ v3\} \Longrightarrow n \cdot x = rat5\ (-3)\ (-1)$
    unfolding *v1-def v2-def v3-def n-def*
    by (*auto simp add*: *vector3-dot rat5-mul rat5-add*)
  have *finite*: *finite* $\{v1,\ v2,\ v3\}$
    using *assms* **by** *blast*
  show *?thesis*

by (*facet-edges plane*: *plane n-nz*: *n-nz defs*: *assms*)
qed

lemma *icosahedron-facet6-edges*:
defines $v1 \equiv vector[rat5\ (-1/2)\ (-1/2),\ rat5\ 0\ 0,\ rat5\ 1\ 0] :: real^3$
and $v2 \equiv vector[rat5\ (-1)\ 0,\ rat5\ (-1/2)\ (-1/2),\ rat5\ 0\ 0] :: real^3$
and $v3 \equiv vector[rat5\ 0\ 0,\ rat5\ (-1)\ 0,\ rat5\ (1/2)\ (1/2)] :: real^3$
and $n \equiv vector[rat5\ (-2)\ 0,\ rat5\ (-2)\ 0,\ rat5\ 2\ 0] :: real^3$
shows $(e\ face\text{-}of\ (convex\ hull\ \{v1,\ v2,\ v3\}) \wedge aff\text{-}dim\ e = 1) =$
$(e = convex\ hull\ \{v1,\ v2\} \vee e = convex\ hull\ \{v1,\ v3\} \vee e = convex\ hull$
$\{v2,\ v3\})$
proof −
have *n-nz*: $n \neq 0$
unfolding *n-def* **by** (*simp add*: *vector3-eq-0 rat5-eq-0*)
have *plane*: $\bigwedge x.\ x \in \{v1,\ v2,\ v3\} \Longrightarrow n \cdot x = rat5\ 3\ 1$
unfolding *v1-def v2-def v3-def n-def*
by (*auto simp add*: *vector3-dot rat5-mul rat5-add*)
have *finite*: *finite* $\{v1,\ v2,\ v3\}$
using *assms* **by** *blast*
show *?thesis*
by (*facet-edges plane*: *plane n-nz*: *n-nz defs*: *assms*)
qed

lemma *icosahedron-facet7-edges*:
defines $v1 \equiv vector[rat5\ (-1/2)\ (-1/2),\ rat5\ 0\ 0,\ rat5\ 1\ 0] :: real^3$
and $v2 \equiv vector[rat5\ (-1)\ 0,\ rat5\ (1/2)\ (1/2),\ rat5\ 0\ 0] :: real^3$
and $v3 \equiv vector[rat5\ 0\ 0,\ rat5\ 1\ 0,\ rat5\ (1/2)\ (1/2)] :: real^3$
and $n \equiv vector[rat5\ 2\ 0,\ rat5\ (-2)\ 0,\ rat5\ (-2)\ 0] :: real^3$
shows $(e\ face\text{-}of\ (convex\ hull\ \{v1,\ v2,\ v3\}) \wedge aff\text{-}dim\ e = 1) =$
$(e = convex\ hull\ \{v1,\ v2\} \vee e = convex\ hull\ \{v1,\ v3\} \vee e = convex\ hull$
$\{v2,\ v3\})$
proof −
have *n-nz*: $n \neq 0$
unfolding *n-def* **by** (*simp add*: *vector3-eq-0 rat5-eq-0*)
have *plane*: $\bigwedge x.\ x \in \{v1,\ v2,\ v3\} \Longrightarrow n \cdot x = rat5\ (-3)\ (-1)$
unfolding *v1-def v2-def v3-def n-def*
by (*auto simp add*: *vector3-dot rat5-mul rat5-add*)
have *finite*: *finite* $\{v1,\ v2,\ v3\}$
using *assms* **by** *blast*
show *?thesis*
by (*facet-edges plane*: *plane n-nz*: *n-nz defs*: *assms*)
qed

lemma *icosahedron-facet8-edges*:
defines $v1 \equiv vector[rat5\ (-1/2)\ (-1/2),\ rat5\ 0\ 0,\ rat5\ 1\ 0] :: real^3$
and $v2 \equiv vector[rat5\ 0\ 0,\ rat5\ (-1)\ 0,\ rat5\ (1/2)\ (1/2)] :: real^3$
and $v3 \equiv vector[rat5\ 0\ 0,\ rat5\ 1\ 0,\ rat5\ (1/2)\ (1/2)] :: real^3$
and $n \equiv vector[rat5\ 1\ (-1),\ rat5\ 0\ 0,\ rat5\ 1\ 1] :: real^3$
shows $(e\ face\text{-}of\ (convex\ hull\ \{v1,\ v2,\ v3\}) \wedge aff\text{-}dim\ e = 1) =$

$(e = \textit{convex hull } \{v1,\ v2\} \lor e = \textit{convex hull } \{v1,\ v3\} \lor e = \textit{convex hull }$
$\{v2,\ v3\})$
proof −
  have *n-nz*: $n \neq 0$
    unfolding *n-def* **by** (*simp add*: *vector3-eq-0 rat5-eq-0*)
  have *plane*: $\bigwedge x.\ x \in \{v1,\ v2,\ v3\} \Longrightarrow n \cdot x = \textit{rat5 } 3\ 1$
    unfolding *v1-def v2-def v3-def n-def*
    by (*auto simp add*: *vector3-dot rat5-mul rat5-add*)
  have *finite*: *finite* $\{v1,\ v2,\ v3\}$
    using *assms* **by** *blast*
  show *?thesis*
    by (*facet-edges plane*: *plane n-nz*: *n-nz defs*: *assms*)
qed

lemma *icosahedron-facet9-edges*:
  defines $v1 \equiv \textit{vector}[\textit{rat5 } (1/2)\ (1/2),\ \textit{rat5 } 0\ 0,\ \textit{rat5 } (-1)\ 0] :: \textit{real}\hat{\ }3$
    and $v2 \equiv \textit{vector}[\textit{rat5 } (1/2)\ (1/2),\ \textit{rat5 } 0\ 0,\ \textit{rat5 } 1\ 0] :: \textit{real}\hat{\ }3$
    and $v3 \equiv \textit{vector}[\textit{rat5 } 1\ 0,\ \textit{rat5 } (-1/2)\ (-1/2),\ \textit{rat5 } 0\ 0] :: \textit{real}\hat{\ }3$
    and $n \equiv \textit{vector}[\textit{rat5 } 1\ 1,\ \textit{rat5 } 1\ (-1),\ \textit{rat5 } 0\ 0] :: \textit{real}\hat{\ }3$
  shows $(e \textit{ face-of } (\textit{convex hull } \{v1,\ v2,\ v3\}) \land \textit{aff-dim } e = 1) =$
    $(e = \textit{convex hull } \{v1,\ v2\} \lor e = \textit{convex hull } \{v1,\ v3\} \lor e = \textit{convex hull }$
$\{v2,\ v3\})$
proof −
  have *n-nz*: $n \neq 0$
    unfolding *n-def* **by** (*simp add*: *vector3-eq-0 rat5-eq-0*)
  have *plane*: $\bigwedge x.\ x \in \{v1,\ v2,\ v3\} \Longrightarrow n \cdot x = \textit{rat5 } 3\ 1$
    unfolding *v1-def v2-def v3-def n-def*
    by (*auto simp add*: *vector3-dot rat5-mul rat5-add*)
  have *finite*: *finite* $\{v1,\ v2,\ v3\}$
    using *assms* **by** *blast*
  show *?thesis*
    by (*facet-edges plane*: *plane n-nz*: *n-nz defs*: *assms*)
qed

lemma *icosahedron-facet10-edges*:
  defines $v1 \equiv \textit{vector}[\textit{rat5 } (1/2)\ (1/2),\ \textit{rat5 } 0\ 0,\ \textit{rat5 } (-1)\ 0] :: \textit{real}\hat{\ }3$
    and $v2 \equiv \textit{vector}[\textit{rat5 } (1/2)\ (1/2),\ \textit{rat5 } 0\ 0,\ \textit{rat5 } 1\ 0] :: \textit{real}\hat{\ }3$
    and $v3 \equiv \textit{vector}[\textit{rat5 } 1\ 0,\ \textit{rat5 } (1/2)\ (1/2),\ \textit{rat5 } 0\ 0] :: \textit{real}\hat{\ }3$
    and $n \equiv \textit{vector}[\textit{rat5 } (-1)\ (-1),\ \textit{rat5 } 1\ (-1),\ \textit{rat5 } 0\ 0] :: \textit{real}\hat{\ }3$
  shows $(e \textit{ face-of } (\textit{convex hull } \{v1,\ v2,\ v3\}) \land \textit{aff-dim } e = 1) =$
    $(e = \textit{convex hull } \{v1,\ v2\} \lor e = \textit{convex hull } \{v1,\ v3\} \lor e = \textit{convex hull }$
$\{v2,\ v3\})$
proof −
  have *n-nz*: $n \neq 0$
    unfolding *n-def* **by** (*simp add*: *vector3-eq-0 rat5-eq-0*)
  have *plane*: $\bigwedge x.\ x \in \{v1,\ v2,\ v3\} \Longrightarrow n \cdot x = \textit{rat5 } (-3)\ (-1)$
    unfolding *v1-def v2-def v3-def n-def*
    by (*auto simp add*: *vector3-dot rat5-mul rat5-add*)
  have *finite*: *finite* $\{v1,\ v2,\ v3\}$

using *assms* **by** *blast*
show *?thesis*
by (*facet-edges plane*: *plane n-nz*: *n-nz defs*: *assms*)
qed

lemma *icosahedron-facet11-edges*:
defines $v1 \equiv vector[rat5\ (1/2)\ (1/2),\ rat5\ 0\ 0,\ rat5\ (-1)\ 0] :: real\verb|^|3$
and $v2 \equiv vector[rat5\ 1\ 0,\ rat5\ (-1/2)\ (-1/2),\ rat5\ 0\ 0] :: real\verb|^|3$
and $v3 \equiv vector[rat5\ 0\ 0,\ rat5\ (-1)\ 0,\ rat5\ (-1/2)\ (-1/2)] :: real\verb|^|3$
and $n \equiv vector[rat5\ 2\ 0,\ rat5\ (-2)\ 0,\ rat5\ (-2)\ 0] :: real\verb|^|3$
shows (e *face-of* (*convex hull* $\{v1, v2, v3\}$) $\wedge$ *aff-dim* $e = 1$) =
(e = *convex hull* $\{v1, v2\}$ $\vee$ e = *convex hull* $\{v1, v3\}$ $\vee$ e = *convex hull* $\{v2, v3\}$)
proof −
have *n-nz*: $n \neq 0$
unfolding *n-def* **by** (*simp add*: *vector3-eq-0 rat5-eq-0*)
have *plane*: $\bigwedge x.\ x \in \{v1, v2, v3\} \Longrightarrow n \cdot x = rat5\ 3\ 1$
unfolding *v1-def v2-def v3-def n-def*
by (*auto simp add*: *vector3-dot rat5-mul rat5-add*)
have *finite*: *finite* $\{v1, v2, v3\}$
using *assms* **by** *blast*
show *?thesis*
by (*facet-edges plane*: *plane n-nz*: *n-nz defs*: *assms*)
qed

lemma *icosahedron-facet12-edges*:
defines $v1 \equiv vector[rat5\ (1/2)\ (1/2),\ rat5\ 0\ 0,\ rat5\ (-1)\ 0] :: real\verb|^|3$
and $v2 \equiv vector[rat5\ 1\ 0,\ rat5\ (1/2)\ (1/2),\ rat5\ 0\ 0] :: real\verb|^|3$
and $v3 \equiv vector[rat5\ 0\ 0,\ rat5\ 1\ 0,\ rat5\ (-1/2)\ (-1/2)] :: real\verb|^|3$
and $n \equiv vector[rat5\ (-2)\ 0,\ rat5\ (-2)\ 0,\ rat5\ 2\ 0] :: real\verb|^|3$
shows (e *face-of* (*convex hull* $\{v1, v2, v3\}$) $\wedge$ *aff-dim* $e = 1$) =
(e = *convex hull* $\{v1, v2\}$ $\vee$ e = *convex hull* $\{v1, v3\}$ $\vee$ e = *convex hull* $\{v2, v3\}$)
proof −
have *n-nz*: $n \neq 0$
unfolding *n-def* **by** (*simp add*: *vector3-eq-0 rat5-eq-0*)
have *plane*: $\bigwedge x.\ x \in \{v1, v2, v3\} \Longrightarrow n \cdot x = rat5\ (-3)\ (-1)$
unfolding *v1-def v2-def v3-def n-def*
by (*auto simp add*: *vector3-dot rat5-mul rat5-add*)
have *finite*: *finite* $\{v1, v2, v3\}$
using *assms* **by** *blast*
show *?thesis*
by (*facet-edges plane*: *plane n-nz*: *n-nz defs*: *assms*)
qed

lemma *icosahedron-facet13-edges*:
defines $v1 \equiv vector[rat5\ (1/2)\ (1/2),\ rat5\ 0\ 0,\ rat5\ (-1)\ 0] :: real\verb|^|3$
and $v2 \equiv vector[rat5\ 0\ 0,\ rat5\ (-1)\ 0,\ rat5\ (-1/2)\ (-1/2)] :: real\verb|^|3$
and $v3 \equiv vector[rat5\ 0\ 0,\ rat5\ 1\ 0,\ rat5\ (-1/2)\ (-1/2)] :: real\verb|^|3$

and $n \equiv$ *vector*[*rat5* (-1) *1*, *rat5 0 0*, *rat5* (-1) (-1)] :: *real^3*
shows (*e face-of* (*convex hull* {*v1*, *v2*, *v3*}) $\wedge$ *aff-dim e* = *1*) =
(*e* = *convex hull* {*v1*, *v2*} $\vee$ *e* = *convex hull* {*v1*, *v3*} $\vee$ *e* = *convex hull*
{*v2*, *v3*})
proof −
have *n-nz*: $n \neq 0$
unfolding *n-def* **by** (*simp add*: *vector3-eq-0 rat5-eq-0*)
have *plane*: $\bigwedge x.\ x \in \{v1, v2, v3\} \Longrightarrow n \cdot x =$ *rat5 3 1*
unfolding *v1-def v2-def v3-def n-def*
by (*auto simp add*: *vector3-dot rat5-mul rat5-add*)
have *finite*: *finite* {*v1*, *v2*, *v3*}
using *assms* **by** *blast*
show *?thesis*
by (*facet-edges plane*: *plane n-nz*: *n-nz defs*: *assms*)
qed

lemma *icosahedron-facet14-edges*:
defines $v1 \equiv$ *vector*[*rat5* $(1/2)$ $(1/2)$, *rat5 0 0*, *rat5 1 0*] :: *real^3*
and $v2 \equiv$ *vector*[*rat5 1 0*, *rat5* $(-1/2)$ $(-1/2)$, *rat5 0 0*] :: *real^3*
and $v3 \equiv$ *vector*[*rat5 0 0*, *rat5* (-1) *0*, *rat5* $(1/2)$ $(1/2)$] :: *real^3*
and $n \equiv$ *vector*[*rat5* (-2) *0*, *rat5 2 0*, *rat5* (-2) *0*] :: *real^3*
shows (*e face-of* (*convex hull* {*v1*, *v2*, *v3*}) $\wedge$ *aff-dim e* = *1*) =
(*e* = *convex hull* {*v1*, *v2*} $\vee$ *e* = *convex hull* {*v1*, *v3*} $\vee$ *e* = *convex hull*
{*v2*, *v3*})
proof −
have *n-nz*: $n \neq 0$
unfolding *n-def* **by** (*simp add*: *vector3-eq-0 rat5-eq-0*)
have *plane*: $\bigwedge x.\ x \in \{v1, v2, v3\} \Longrightarrow n \cdot x =$ *rat5* (-3) (-1)
unfolding *v1-def v2-def v3-def n-def*
by (*auto simp add*: *vector3-dot rat5-mul rat5-add*)
have *finite*: *finite* {*v1*, *v2*, *v3*}
using *assms* **by** *blast*
show *?thesis*
by (*facet-edges plane*: *plane n-nz*: *n-nz defs*: *assms*)
qed

lemma *icosahedron-facet15-edges*:
defines $v1 \equiv$ *vector*[*rat5* $(1/2)$ $(1/2)$, *rat5 0 0*, *rat5 1 0*] :: *real^3*
and $v2 \equiv$ *vector*[*rat5 1 0*, *rat5* $(1/2)$ $(1/2)$, *rat5 0 0*] :: *real^3*
and $v3 \equiv$ *vector*[*rat5 0 0*, *rat5 1 0*, *rat5* $(1/2)$ $(1/2)$] :: *real^3*
and $n \equiv$ *vector*[*rat5 2 0*, *rat5 2 0*, *rat5 2 0*] :: *real^3*
shows (*e face-of* (*convex hull* {*v1*, *v2*, *v3*}) $\wedge$ *aff-dim e* = *1*) =
(*e* = *convex hull* {*v1*, *v2*} $\vee$ *e* = *convex hull* {*v1*, *v3*} $\vee$ *e* = *convex hull*
{*v2*, *v3*})
proof −
have *n-nz*: $n \neq 0$
unfolding *n-def* **by** (*simp add*: *vector3-eq-0 rat5-eq-0*)
have *plane*: $\bigwedge x.\ x \in \{v1, v2, v3\} \Longrightarrow n \cdot x =$ *rat5 3 1*
unfolding *v1-def v2-def v3-def n-def*

    by (*auto simp add*: *vector3-dot rat5-mul rat5-add*)
  have *finite*: *finite* {*v1*, *v2*, *v3*}
    using *assms* **by** *blast*
  show *?thesis*
    by (*facet-edges plane*: *plane n-nz*: *n-nz defs*: *assms*)
qed

lemma *icosahedron-facet16-edges*:
  defines $v1 \equiv vector[rat5\ (1/2)\ (1/2),\ rat5\ 0\ 0,\ rat5\ 1\ 0] :: real\hat{\ }3$
      and $v2 \equiv vector[rat5\ 0\ 0,\ rat5\ (-1)\ 0,\ rat5\ (1/2)\ (1/2)] :: real\hat{\ }3$
      and $v3 \equiv vector[rat5\ 0\ 0,\ rat5\ 1\ 0,\ rat5\ (1/2)\ (1/2)] :: real\hat{\ }3$
      and $n \equiv vector[rat5\ 1\ (-1),\ rat5\ 0\ 0,\ rat5\ (-1)\ (-1)] :: real\hat{\ }3$
    shows (*e face-of* (*convex hull* {*v1*, *v2*, *v3*}) ∧ *aff-dim e* = *1*) =
          (*e* = *convex hull* {*v1*, *v2*} ∨ *e* = *convex hull* {*v1*, *v3*} ∨ *e* = *convex hull*
{*v2*, *v3*})
proof −
  have *n-nz*: $n \neq 0$
    unfolding *n-def* **by** (*simp add*: *vector3-eq-0 rat5-eq-0*)
  have *plane*: $\bigwedge x.\ x \in \{v1,\ v2,\ v3\} \Longrightarrow n \cdot x = rat5\ (-3)\ (-1)$
    unfolding *v1-def v2-def v3-def n-def*
    by (*auto simp add*: *vector3-dot rat5-mul rat5-add*)
  have *finite*: *finite* {*v1*, *v2*, *v3*}
    using *assms* **by** *blast*
  show *?thesis*
    by (*facet-edges plane*: *plane n-nz*: *n-nz defs*: *assms*)
qed

lemma *icosahedron-facet17-edges*:
  defines $v1 \equiv vector[rat5\ (-1)\ 0,\ rat5\ (-1/2)\ (-1/2),\ rat5\ 0\ 0] :: real\hat{\ }3$
      and $v2 \equiv vector[rat5\ 1\ 0,\ rat5\ (-1/2)\ (-1/2),\ rat5\ 0\ 0] :: real\hat{\ }3$
      and $v3 \equiv vector[rat5\ 0\ 0,\ rat5\ (-1)\ 0,\ rat5\ (-1/2)\ (-1/2)] :: real\hat{\ }3$
      and $n \equiv vector[rat5\ 0\ 0,\ rat5\ 1\ 1,\ rat5\ (-1)\ 1] :: real\hat{\ }3$
    shows (*e face-of* (*convex hull* {*v1*, *v2*, *v3*}) ∧ *aff-dim e* = *1*) =
          (*e* = *convex hull* {*v1*, *v2*} ∨ *e* = *convex hull* {*v1*, *v3*} ∨ *e* = *convex hull*
{*v2*, *v3*})
proof −
  have *n-nz*: $n \neq 0$
    unfolding *n-def* **by** (*simp add*: *vector3-eq-0 rat5-eq-0*)
  have *plane*: $\bigwedge x.\ x \in \{v1,\ v2,\ v3\} \Longrightarrow n \cdot x = rat5\ (-3)\ (-1)$
    unfolding *v1-def v2-def v3-def n-def*
    by (*auto simp add*: *vector3-dot rat5-mul rat5-add*)
  have *finite*: *finite* {*v1*, *v2*, *v3*}
    using *assms* **by** *blast*
  show *?thesis*
    by (*facet-edges plane*: *plane n-nz*: *n-nz defs*: *assms*)
qed

lemma *icosahedron-facet18-edges*:
  defines $v1 \equiv vector[rat5\ (-1)\ 0,\ rat5\ (-1/2)\ (-1/2),\ rat5\ 0\ 0] :: real\hat{\ }3$

and *v2* ≡ *vector*[*rat5 1 0*, *rat5* (−*1*/*2*) (−*1*/*2*), *rat5 0 0*] :: *real^3*
and *v3* ≡ *vector*[*rat5 0 0*, *rat5* (−*1*) *0*, *rat5* (*1*/*2*) (*1*/*2*)] :: *real^3*
and *n* ≡ *vector*[*rat5 0 0*, *rat5* (−*1*) (−*1*), *rat5* (−*1*) *1*] :: *real^3*
shows (*e face-of* (*convex hull* {*v1*, *v2*, *v3*}) ∧ *aff-dim e* = *1*) =
(*e* = *convex hull* {*v1*, *v2*} ∨ *e* = *convex hull* {*v1*, *v3*} ∨ *e* = *convex hull* {*v2*, *v3*})
proof −
have *n-nz*: *n* ≠ *0*
unfolding *n-def* **by** (*simp add*: *vector3-eq-0 rat5-eq-0*)
have *plane*: ⋀*x*. *x* ∈ {*v1*, *v2*, *v3*} ⟹ *n* · *x* = *rat5 3 1*
unfolding *v1-def v2-def v3-def n-def*
by (*auto simp add*: *vector3-dot rat5-mul rat5-add*)
have *finite*: *finite* {*v1*, *v2*, *v3*}
using *assms* **by** *blast*
show *?thesis*
by (*facet-edges plane*: *plane n-nz*: *n-nz defs*: *assms*)
qed

lemma *icosahedron-facet19-edges*:
defines *v1* ≡ *vector*[*rat5* (−*1*) *0*, *rat5* (*1*/*2*) (*1*/*2*), *rat5 0 0*] :: *real^3*
and *v2* ≡ *vector*[*rat5 1 0*, *rat5* (*1*/*2*) (*1*/*2*), *rat5 0 0*] :: *real^3*
and *v3* ≡ *vector*[*rat5 0 0*, *rat5 1 0*, *rat5* (−*1*/*2*) (−*1*/*2*)] :: *real^3*
and *n* ≡ *vector*[*rat5 0 0*, *rat5 1 1*, *rat5 1* (−*1*)] :: *real^3*
shows (*e face-of* (*convex hull* {*v1*, *v2*, *v3*}) ∧ *aff-dim e* = *1*) =
(*e* = *convex hull* {*v1*, *v2*} ∨ *e* = *convex hull* {*v1*, *v3*} ∨ *e* = *convex hull* {*v2*, *v3*})
proof −
have *n-nz*: *n* ≠ *0*
unfolding *n-def* **by** (*simp add*: *vector3-eq-0 rat5-eq-0*)
have *plane*: ⋀*x*. *x* ∈ {*v1*, *v2*, *v3*} ⟹ *n* · *x* = *rat5 3 1*
unfolding *v1-def v2-def v3-def n-def*
by (*auto simp add*: *vector3-dot rat5-mul rat5-add*)
have *finite*: *finite* {*v1*, *v2*, *v3*}
using *assms* **by** *blast*
show *?thesis*
by (*facet-edges plane*: *plane n-nz*: *n-nz defs*: *assms*)
qed

lemma *icosahedron-facet20-edges*:
defines *v1* ≡ *vector*[*rat5* (−*1*) *0*, *rat5* (*1*/*2*) (*1*/*2*), *rat5 0 0*] :: *real^3*
and *v2* ≡ *vector*[*rat5 1 0*, *rat5* (*1*/*2*) (*1*/*2*), *rat5 0 0*] :: *real^3*
and *v3* ≡ *vector*[*rat5 0 0*, *rat5 1 0*, *rat5* (*1*/*2*) (*1*/*2*)] :: *real^3*
and *n* ≡ *vector*[*rat5 0 0*, *rat5* (−*1*) (−*1*), *rat5 1* (−*1*)] :: *real^3*
shows (*e face-of* (*convex hull* {*v1*, *v2*, *v3*}) ∧ *aff-dim e* = *1*) =
(*e* = *convex hull* {*v1*, *v2*} ∨ *e* = *convex hull* {*v1*, *v3*} ∨ *e* = *convex hull* {*v2*, *v3*})
proof −
have *n-nz*: *n* ≠ *0*
unfolding *n-def* **by** (*simp add*: *vector3-eq-0 rat5-eq-0*)

have *plane*: $\bigwedge x.\ x \in \{v1,\ v2,\ v3\} \Longrightarrow n \cdot x = rat5\ (-3)\ (-1)$
  unfolding *v1-def v2-def v3-def n-def*
  by (*auto simp add*: *vector3-dot rat5-mul rat5-add*)
have *finite*: *finite* $\{v1,\ v2,\ v3\}$
  using *assms* **by** *blast*
show *?thesis*
  by (*facet-edges plane*: *plane n-nz*: *n-nz defs*: *assms*)
qed

lemmas *icosahedron-facet-edges* =
icosahedron-facet1-edges icosahedron-facet2-edges icosahedron-facet3-edges
icosahedron-facet4-edges icosahedron-facet5-edges icosahedron-facet6-edges
icosahedron-facet7-edges icosahedron-facet8-edges icosahedron-facet9-edges
icosahedron-facet10-edges icosahedron-facet11-edges icosahedron-facet12-edges
icosahedron-facet13-edges icosahedron-facet14-edges icosahedron-facet15-edges
icosahedron-facet16-edges icosahedron-facet17-edges icosahedron-facet18-edges
icosahedron-facet19-edges icosahedron-facet20-edges

lemma *icosahedron-edges-per-face*:
assumes *f face-of std-icosahedron*
  and *aff-dim f* = *2*
shows *card* $\{e.\ e$ *face-of std-icosahedron* $\wedge$ *aff-dim e* = *1* $\wedge\ e \subseteq f\}$ = *3*
using *conjI*[*OF assms*] **unfolding** *icosahedron-facets face-edge-set-eq*[*OF assms*(*1*)]
apply (*elim disjE forw-subst*)
unfolding *icosahedron-facet-edges set-cases card-3-iff2*
  segment-convex-hull[*symmetric*] *closed-segment-eq*
by (*simp-all add*: *vector3-eq doubleton-eq-iff rat5-eq*)

lemma *icosahedron-edges*:
e face-of std-icosahedron $\wedge$ *aff-dim e* = *1* $\longleftrightarrow$
e = *convex hull* {*vector*[*rat5* (−*1*/*2*) (−*1*/*2*), *rat5 0 0*, *rat5* (−*1*) *0*], *vector*[*rat5* (−*1*/*2*) (−*1*/*2*), *rat5 0 0*, *rat5 1 0*]} ∨
e = *convex hull* {*vector*[*rat5* (−*1*/*2*) (−*1*/*2*), *rat5 0 0*, *rat5* (−*1*) *0*], *vector*[*rat5* (−*1*) *0*, *rat5* (−*1*/*2*) (−*1*/*2*), *rat5 0 0*]} ∨
e = *convex hull* {*vector*[*rat5* (−*1*/*2*) (−*1*/*2*), *rat5 0 0*, *rat5* (−*1*) *0*], *vector*[*rat5* (−*1*) *0*, *rat5* (*1*/*2*) (*1*/*2*), *rat5 0 0*]} ∨
e = *convex hull* {*vector*[*rat5* (−*1*/*2*) (−*1*/*2*), *rat5 0 0*, *rat5* (−*1*) *0*], *vector*[*rat5 0 0*, *rat5* (−*1*) *0*, *rat5* (−*1*/*2*) (−*1*/*2*)]} ∨
e = *convex hull* {*vector*[*rat5* (−*1*/*2*) (−*1*/*2*), *rat5 0 0*, *rat5* (−*1*) *0*], *vector*[*rat5 0 0*, *rat5 1 0*, *rat5* (−*1*/*2*) (−*1*/*2*)]} ∨
e = *convex hull* {*vector*[*rat5* (−*1*/*2*) (−*1*/*2*), *rat5 0 0*, *rat5 1 0*], *vector*[*rat5* (−*1*) *0*, *rat5* (−*1*/*2*) (−*1*/*2*), *rat5 0 0*]} ∨
e = *convex hull* {*vector*[*rat5* (−*1*/*2*) (−*1*/*2*), *rat5 0 0*, *rat5 1 0*], *vector*[*rat5* (−*1*) *0*, *rat5* (*1*/*2*) (*1*/*2*), *rat5 0 0*]} ∨
e = *convex hull* {*vector*[*rat5* (−*1*/*2*) (−*1*/*2*), *rat5 0 0*, *rat5 1 0*], *vector*[*rat5 0 0*, *rat5* (−*1*) *0*, *rat5* (*1*/*2*) (*1*/*2*)]} ∨
e = *convex hull* {*vector*[*rat5* (−*1*/*2*) (−*1*/*2*), *rat5 0 0*, *rat5 1 0*], *vector*[*rat5 0 0*, *rat5 1 0*, *rat5* (*1*/*2*) (*1*/*2*)]} ∨
e = *convex hull* {*vector*[*rat5* (*1*/*2*) (*1*/*2*), *rat5 0 0*, *rat5* (−*1*) *0*], *vector*[*rat5*

$(1/2)$ $(1/2)$, $rat5\ 0\ 0$, $rat5\ 1\ 0$]} ∨
  e = convex hull {vector[rat5 (1/2) (1/2), rat5 0 0, rat5 (−1) 0], vector[rat5 1 0, rat5 (−1/2) (−1/2), rat5 0 0]} ∨
  e = convex hull {vector[rat5 (1/2) (1/2), rat5 0 0, rat5 (−1) 0], vector[rat5 1 0, rat5 (1/2) (1/2), rat5 0 0]} ∨
  e = convex hull {vector[rat5 (1/2) (1/2), rat5 0 0, rat5 (−1) 0], vector[rat5 0 0, rat5 (−1) 0, rat5 (−1/2) (−1/2)]} ∨
  e = convex hull {vector[rat5 (1/2) (1/2), rat5 0 0, rat5 (−1) 0], vector[rat5 0 0, rat5 1 0, rat5 (−1/2) (−1/2)]} ∨
  e = convex hull {vector[rat5 (1/2) (1/2), rat5 0 0, rat5 1 0], vector[rat5 1 0, rat5 (−1/2) (−1/2), rat5 0 0]} ∨
  e = convex hull {vector[rat5 (1/2) (1/2), rat5 0 0, rat5 1 0], vector[rat5 1 0, rat5 (1/2) (1/2), rat5 0 0]} ∨
  e = convex hull {vector[rat5 (1/2) (1/2), rat5 0 0, rat5 1 0], vector[rat5 0 0, rat5 (−1) 0, rat5 (1/2) (1/2)]} ∨
  e = convex hull {vector[rat5 (1/2) (1/2), rat5 0 0, rat5 1 0], vector[rat5 0 0, rat5 1 0, rat5 (1/2) (1/2)]} ∨
  e = convex hull {vector[rat5 (−1) 0, rat5 (−1/2) (−1/2), rat5 0 0], vector[rat5 1 0, rat5 (−1/2) (−1/2), rat5 0 0]} ∨
  e = convex hull {vector[rat5 (−1) 0, rat5 (−1/2) (−1/2), rat5 0 0], vector[rat5 0 0, rat5 (−1) 0, rat5 (−1/2) (−1/2)]} ∨
  e = convex hull {vector[rat5 (−1) 0, rat5 (−1/2) (−1/2), rat5 0 0], vector[rat5 0 0, rat5 (−1) 0, rat5 (1/2) (1/2)]} ∨
  e = convex hull {vector[rat5 (−1) 0, rat5 (1/2) (1/2), rat5 0 0], vector[rat5 1 0, rat5 (1/2) (1/2), rat5 0 0]} ∨
  e = convex hull {vector[rat5 (−1) 0, rat5 (1/2) (1/2), rat5 0 0], vector[rat5 0 0, rat5 1 0, rat5 (−1/2) (−1/2)]} ∨
  e = convex hull {vector[rat5 (−1) 0, rat5 (1/2) (1/2), rat5 0 0], vector[rat5 0 0, rat5 1 0, rat5 (1/2) (1/2)]} ∨
  e = convex hull {vector[rat5 1 0, rat5 (−1/2) (−1/2), rat5 0 0], vector[rat5 0 0, rat5 (−1) 0, rat5 (−1/2) (−1/2)]} ∨
  e = convex hull {vector[rat5 1 0, rat5 (−1/2) (−1/2), rat5 0 0], vector[rat5 0 0, rat5 (−1) 0, rat5 (1/2) (1/2)]} ∨
  e = convex hull {vector[rat5 1 0, rat5 (1/2) (1/2), rat5 0 0], vector[rat5 0 0, rat5 1 0, rat5 (−1/2) (−1/2)]} ∨
  e = convex hull {vector[rat5 1 0, rat5 (1/2) (1/2), rat5 0 0], vector[rat5 0 0, rat5 1 0, rat5 (1/2) (1/2)]} ∨
  e = convex hull {vector[rat5 0 0, rat5 (−1) 0, rat5 (−1/2) (−1/2)], vector[rat5 0 0, rat5 1 0, rat5 (−1/2) (−1/2)]} ∨
  e = convex hull {vector[rat5 0 0, rat5 (−1) 0, rat5 (1/2) (1/2)], vector[rat5 0 0, rat5 1 0, rat5 (1/2) (1/2)]}
 (**is** ?lhs ⟷ ?rhs)
proof
 assume ?lhs
 then obtain f **where** f-facts: f face-of std-icosahedron ∧ aff-dim f = 2 e face-of f ∧ aff-dim e = 1
  using icosahedron-polyhedron icosahedron-fulldim edge-belongs-to-face
  by blast
 show ?rhs

using *f-facts* **unfolding** *icosahedron-facets*
apply (*elim disjE*) **apply** (*all* ‹*hypsubst-thin*›)
unfolding *icosahedron-facet-edges*
by *linarith+*
next
assume *?rhs*
then show *?lhs*
using *icosahedron-facets icosahedron-facet-edges*

by (*smt* (*verit*, *ccfv-threshold*) *face-of-trans insertI1 insert-commute mem-Collect-eq*)
qed

lemma *icosahedron-vertices*:
shows (v *face-of std-icosahedron* $\wedge$ *aff-dim* $v = 0$) =
(v = {*vector*[*rat5 0 0*, *rat5 1 0*, *rat5* (*1/2*) (*1/2*)]} $\vee$
v = {*vector*[*rat5 0 0*, *rat5 1 0*, *rat5* (−*1/2*) (−*1/2*)]} $\vee$
v = {*vector*[*rat5 0 0*, *rat5* (−*1*) *0*, *rat5* (*1/2*) (*1/2*)]} $\vee$
v = {*vector*[*rat5 0 0*, *rat5* (−*1*) *0*, *rat5* (−*1/2*) (−*1/2*)]} $\vee$
v = {*vector*[*rat5 1 0*, *rat5* (*1/2*) (*1/2*), *rat5 0 0*]} $\vee$
v = {*vector*[*rat5 1 0*, *rat5* (−*1/2*) (−*1/2*), *rat5 0 0*]} $\vee$
v = {*vector*[*rat5* (−*1*) *0*, *rat5* (*1/2*) (*1/2*), *rat5 0 0*]} $\vee$
v = {*vector*[*rat5* (−*1*) *0*, *rat5* (−*1/2*) (−*1/2*), *rat5 0 0*]} $\vee$
v = {*vector*[*rat5* (*1/2*) (*1/2*), *rat5 0 0*, *rat5 1 0*]} $\vee$
v = {*vector*[*rat5* (−*1/2*) (−*1/2*), *rat5 0 0*, *rat5 1 0*]} $\vee$
v = {*vector*[*rat5* (*1/2*) (*1/2*), *rat5 0 0*, *rat5* (−*1*) *0*]} $\vee$
v = {*vector*[*rat5* (−*1/2*) (−*1/2*), *rat5 0 0*, *rat5* (−*1*) *0*]})
(**is** *?lhs* = *?rhs*)
proof
assume ∗: *?lhs*
then obtain e **where** e *face-of std-icosahedron aff-dim* $e = 1$ v *face-of* e
using *icosahedron-polyhedron icosahedron-fulldim vertex-belongs-to-edge*
by *blast*
show *?rhs*
using *conjI*[*OF* ‹*e face-of std-icosahedron*› ‹*aff-dim e* = *1*›]
unfolding *icosahedron-edges*
using *vertices-of-edge*[*OF* - ‹*v face-of e*›] ∗
by *metis+*
next
assume *?rhs*
then show *?lhs*
apply (*elim disjE forw-subst*)
by (*smt* (*z3*) *aff-dim-sing extreme-point-of-segment face-of-singleton face-of-trans segment-convex-hull icosahedron-edges*)+
qed

lemma *icosahedron-edges-per-vertex*:
assumes v *face-of std-icosahedron*
and *aff-dim* $v = 0$
shows *card* {e. e *face-of std-icosahedron* $\wedge$ *aff-dim* $e = 1 \wedge v \subseteq e$} = *5*

unfolding *vertex-edge-set-eq*[*OF assms*(*1*)]
using *conjI*[*OF assms*]
unfolding *icosahedron-vertices conj-assoc*[*symmetric*] *icosahedron-edges*
by (*elim disjE forw-subst*) *edges-per-vertex*+

9.3 Regularity

lemma *icosahedron-congruent-edges*:
assumes *e1 face-of std-icosahedron* ∧ *aff-dim e1* = *1*
and *e2 face-of std-icosahedron* ∧ *aff-dim e2* = *1*
shows *e1 congruent e2*
proof −
let *?edges* =
{*convex hull* {(*vector*[*rat5* (−*1*/*2*) (−*1*/*2*), *rat5 0 0*, *rat5* (−*1*) *0*]::*real^3*), *vector*[*rat5* (−*1*/*2*) (−*1*/*2*), *rat5 0 0*, *rat5 1 0*]},
convex hull {*vector*[*rat5* (−*1*/*2*) (−*1*/*2*), *rat5 0 0*, *rat5* (−*1*) *0*], *vector*[*rat5* (−*1*) *0*, *rat5* (−*1*/*2*) (−*1*/*2*), *rat5 0 0*]},
convex hull {*vector*[*rat5* (−*1*/*2*) (−*1*/*2*), *rat5 0 0*, *rat5* (−*1*) *0*], *vector*[*rat5* (−*1*) *0*, *rat5* (*1*/*2*) (*1*/*2*), *rat5 0 0*]},
convex hull {*vector*[*rat5* (−*1*/*2*) (−*1*/*2*), *rat5 0 0*, *rat5* (−*1*) *0*], *vector*[*rat5 0 0*, *rat5* (−*1*) *0*, *rat5* (−*1*/*2*) (−*1*/*2*)]},
convex hull {*vector*[*rat5* (−*1*/*2*) (−*1*/*2*), *rat5 0 0*, *rat5* (−*1*) *0*], *vector*[*rat5 0 0*, *rat5 1 0*, *rat5* (−*1*/*2*) (−*1*/*2*)]},
convex hull {*vector*[*rat5* (−*1*/*2*) (−*1*/*2*), *rat5 0 0*, *rat5 1 0*], *vector*[*rat5* (−*1*) *0*, *rat5* (−*1*/*2*) (−*1*/*2*), *rat5 0 0*]},
convex hull {*vector*[*rat5* (−*1*/*2*) (−*1*/*2*), *rat5 0 0*, *rat5 1 0*], *vector*[*rat5* (−*1*) *0*, *rat5* (*1*/*2*) (*1*/*2*), *rat5 0 0*]},
convex hull {*vector*[*rat5* (−*1*/*2*) (−*1*/*2*), *rat5 0 0*, *rat5 1 0*], *vector*[*rat5 0 0*, *rat5* (−*1*) *0*, *rat5* (*1*/*2*) (*1*/*2*)]},
convex hull {*vector*[*rat5* (−*1*/*2*) (−*1*/*2*), *rat5 0 0*, *rat5 1 0*], *vector*[*rat5 0 0*, *rat5 1 0*, *rat5* (*1*/*2*) (*1*/*2*)]},
convex hull {*vector*[*rat5* (*1*/*2*) (*1*/*2*), *rat5 0 0*, *rat5* (−*1*) *0*], *vector*[*rat5* (*1*/*2*) (*1*/*2*), *rat5 0 0*, *rat5 1 0*]},
convex hull {*vector*[*rat5* (*1*/*2*) (*1*/*2*), *rat5 0 0*, *rat5* (−*1*) *0*], *vector*[*rat5 1 0*, *rat5* (−*1*/*2*) (−*1*/*2*), *rat5 0 0*]},
convex hull {*vector*[*rat5* (*1*/*2*) (*1*/*2*), *rat5 0 0*, *rat5* (−*1*) *0*], *vector*[*rat5 1 0*, *rat5* (*1*/*2*) (*1*/*2*), *rat5 0 0*]},
convex hull {*vector*[*rat5* (*1*/*2*) (*1*/*2*), *rat5 0 0*, *rat5* (−*1*) *0*], *vector*[*rat5 0 0*, *rat5* (−*1*) *0*, *rat5* (−*1*/*2*) (−*1*/*2*)]},
convex hull {*vector*[*rat5* (*1*/*2*) (*1*/*2*), *rat5 0 0*, *rat5* (−*1*) *0*], *vector*[*rat5 0 0*, *rat5 1 0*, *rat5* (−*1*/*2*) (−*1*/*2*)]},
convex hull {*vector*[*rat5* (*1*/*2*) (*1*/*2*), *rat5 0 0*, *rat5 1 0*], *vector*[*rat5 1 0*, *rat5* (−*1*/*2*) (−*1*/*2*), *rat5 0 0*]},
convex hull {*vector*[*rat5* (*1*/*2*) (*1*/*2*), *rat5 0 0*, *rat5 1 0*], *vector*[*rat5 1 0*, *rat5* (*1*/*2*) (*1*/*2*), *rat5 0 0*]},
convex hull {*vector*[*rat5* (*1*/*2*) (*1*/*2*), *rat5 0 0*, *rat5 1 0*], *vector*[*rat5 0 0*, *rat5* (−*1*) *0*, *rat5* (*1*/*2*) (*1*/*2*)]},
convex hull {*vector*[*rat5* (*1*/*2*) (*1*/*2*), *rat5 0 0*, *rat5 1 0*], *vector*[*rat5 0 0*, *rat5 1 0*, *rat5* (*1*/*2*) (*1*/*2*)]},

convex hull {*vector*[*rat5* (−*1*) *0*, *rat5* (−*1/2*) (−*1/2*), *rat5 0 0*], *vector*[*rat5 1 0*, *rat5* (−*1/2*) (−*1/2*), *rat5 0 0*]},
convex hull {*vector*[*rat5* (−*1*) *0*, *rat5* (−*1/2*) (−*1/2*), *rat5 0 0*], *vector*[*rat5 0 0*, *rat5* (−*1*) *0*, *rat5* (−*1/2*) (−*1/2*)]},
convex hull {*vector*[*rat5* (−*1*) *0*, *rat5* (−*1/2*) (−*1/2*), *rat5 0 0*], *vector*[*rat5 0 0*, *rat5* (−*1*) *0*, *rat5* (*1/2*) (*1/2*)]},
convex hull {*vector*[*rat5* (−*1*) *0*, *rat5* (*1/2*) (*1/2*), *rat5 0 0*], *vector*[*rat5 1 0*, *rat5* (*1/2*) (*1/2*), *rat5 0 0*]},
convex hull {*vector*[*rat5* (−*1*) *0*, *rat5* (*1/2*) (*1/2*), *rat5 0 0*], *vector*[*rat5 0 0*, *rat5 1 0*, *rat5* (−*1/2*) (−*1/2*)]},
convex hull {*vector*[*rat5* (−*1*) *0*, *rat5* (*1/2*) (*1/2*), *rat5 0 0*], *vector*[*rat5 0 0*, *rat5 1 0*, *rat5* (*1/2*) (*1/2*)]},
convex hull {*vector*[*rat5 1 0*, *rat5* (−*1/2*) (−*1/2*), *rat5 0 0*], *vector*[*rat5 0 0*, *rat5* (−*1*) *0*, *rat5* (−*1/2*) (−*1/2*)]},
convex hull {*vector*[*rat5 1 0*, *rat5* (−*1/2*) (−*1/2*), *rat5 0 0*], *vector*[*rat5 0 0*, *rat5* (−*1*) *0*, *rat5* (*1/2*) (*1/2*)]},
convex hull {*vector*[*rat5 1 0*, *rat5* (*1/2*) (*1/2*), *rat5 0 0*], *vector*[*rat5 0 0*, *rat5 1 0*, *rat5* (−*1/2*) (−*1/2*)]},
convex hull {*vector*[*rat5 1 0*, *rat5* (*1/2*) (*1/2*), *rat5 0 0*], *vector*[*rat5 0 0*, *rat5 1 0*, *rat5* (*1/2*) (*1/2*)]},
convex hull {*vector*[*rat5 0 0*, *rat5* (−*1*) *0*, *rat5* (−*1/2*) (−*1/2*)], *vector*[*rat5 0 0*, *rat5 1 0*, *rat5* (−*1/2*) (−*1/2*)]},
convex hull {*vector*[*rat5 0 0*, *rat5* (−*1*) *0*, *rat5* (*1/2*) (*1/2*)], *vector*[*rat5 0 0*, *rat5 1 0*, *rat5* (*1/2*) (*1/2*)]}}
have *e1* ∈ *?edges e2* ∈ *?edges*
using *assms* **unfolding** *icosahedron-edges*
by *auto*
moreover have ∀ *e*∈*?edges*.
convex hull {*vector*[*rat5* (−*1/2*) (−*1/2*), *rat5 0 0*, *rat5* (−*1*) *0*], *vector*[*rat5* (−*1/2*) (−*1/2*), *rat5 0 0*, *rat5 1 0*]} *congruent e*
apply (*simp only*: *ball-simps*(*7*) *Set.ball-empty simp-thms*(*21*))
unfolding *conj-assoc*[*symmetric*] *segment-convex-hull*[*symmetric*]
apply (*rule conjI*)+
apply (*all* ‹*rule congruent-segments*›)
unfolding *dist-norm vector3-sub*
apply (*simp-all add*: *norm-eq rat5-sub del*: *cancel-comm-monoid-add-class.diff-cancel diff-self*)
by (*simp-all add*: *vector3-dot rat5-mul rat5-add rat5-eq*)
ultimately show *?thesis*
using *congruent-set* **by** *meson*
qed

lemma *icosahedron-congruent-faces*:
assumes *f1 face-of std-icosahedron* ∧ *aff-dim f1* = *2*
and *f2 face-of std-icosahedron* ∧ *aff-dim f2* = *2*
shows *f1 congruent f2*
proof −
let *?faces* =
{*convex hull* {(*vector*[*rat5* (−*1/2*) (−*1/2*), *rat5 0 0*, *rat5* (−*1*) *0*]::*real^3*),

vector[*rat5* (−*1*/*2*) (−*1*/*2*), *rat5 0 0*, *rat5 1 0*], *vector*[*rat5* (−*1*) *0*, *rat5* (−*1*/*2*) (−*1*/*2*), *rat5 0 0*]},

convex hull {*vector*[*rat5* (−*1*/*2*) (−*1*/*2*), *rat5 0 0*, *rat5* (−*1*) *0*], *vector*[*rat5* (−*1*/*2*) (−*1*/*2*), *rat5 0 0*, *rat5 1 0*], *vector*[*rat5* (−*1*) *0*, *rat5* (*1*/*2*) (*1*/*2*), *rat5 0 0*]},

convex hull {*vector*[*rat5* (−*1*/*2*) (−*1*/*2*), *rat5 0 0*, *rat5* (−*1*) *0*], *vector*[*rat5* (−*1*) *0*, *rat5* (−*1*/*2*) (−*1*/*2*), *rat5 0 0*], *vector*[*rat5 0 0*, *rat5* (−*1*) *0*, *rat5* (−*1*/*2*) (−*1*/*2*)]},

convex hull {*vector*[*rat5* (−*1*/*2*) (−*1*/*2*), *rat5 0 0*, *rat5* (−*1*) *0*], *vector*[*rat5* (−*1*) *0*, *rat5* (*1*/*2*) (*1*/*2*), *rat5 0 0*], *vector*[*rat5 0 0*, *rat5 1 0*, *rat5* (−*1*/*2*) (−*1*/*2*)]},

convex hull {*vector*[*rat5* (−*1*/*2*) (−*1*/*2*), *rat5 0 0*, *rat5* (−*1*) *0*], *vector*[*rat5 0 0*, *rat5* (−*1*) *0*, *rat5* (−*1*/*2*) (−*1*/*2*)], *vector*[*rat5 0 0*, *rat5 1 0*, *rat5* (−*1*/*2*) (−*1*/*2*)]},

convex hull {*vector*[*rat5* (−*1*/*2*) (−*1*/*2*), *rat5 0 0*, *rat5 1 0*], *vector*[*rat5* (−*1*) *0*, *rat5* (−*1*/*2*) (−*1*/*2*), *rat5 0 0*], *vector*[*rat5 0 0*, *rat5* (−*1*) *0*, *rat5* (*1*/*2*) (*1*/*2*)]},

convex hull {*vector*[*rat5* (−*1*/*2*) (−*1*/*2*), *rat5 0 0*, *rat5 1 0*], *vector*[*rat5* (−*1*) *0*, *rat5* (*1*/*2*) (*1*/*2*), *rat5 0 0*], *vector*[*rat5 0 0*, *rat5 1 0*, *rat5* (*1*/*2*) (*1*/*2*)]},

convex hull {*vector*[*rat5* (−*1*/*2*) (−*1*/*2*), *rat5 0 0*, *rat5 1 0*], *vector*[*rat5 0 0*, *rat5* (−*1*) *0*, *rat5* (*1*/*2*) (*1*/*2*)], *vector*[*rat5 0 0*, *rat5 1 0*, *rat5* (*1*/*2*) (*1*/*2*)]},

convex hull {*vector*[*rat5* (*1*/*2*) (*1*/*2*), *rat5 0 0*, *rat5* (−*1*) *0*], *vector*[*rat5* (*1*/*2*) (*1*/*2*), *rat5 0 0*, *rat5 1 0*], *vector*[*rat5 1 0*, *rat5* (−*1*/*2*) (−*1*/*2*), *rat5 0 0*]},

convex hull {*vector*[*rat5* (*1*/*2*) (*1*/*2*), *rat5 0 0*, *rat5* (−*1*) *0*], *vector*[*rat5* (*1*/*2*) (*1*/*2*), *rat5 0 0*, *rat5 1 0*], *vector*[*rat5 1 0*, *rat5* (*1*/*2*) (*1*/*2*), *rat5 0 0*]},

convex hull {*vector*[*rat5* (*1*/*2*) (*1*/*2*), *rat5 0 0*, *rat5* (−*1*) *0*], *vector*[*rat5 1 0*, *rat5* (−*1*/*2*) (−*1*/*2*), *rat5 0 0*], *vector*[*rat5 0 0*, *rat5* (−*1*) *0*, *rat5* (−*1*/*2*) (−*1*/*2*)]},

convex hull {*vector*[*rat5* (*1*/*2*) (*1*/*2*), *rat5 0 0*, *rat5* (−*1*) *0*], *vector*[*rat5 1 0*, *rat5* (*1*/*2*) (*1*/*2*), *rat5 0 0*], *vector*[*rat5 0 0*, *rat5 1 0*, *rat5* (−*1*/*2*) (−*1*/*2*)]},

convex hull {*vector*[*rat5* (*1*/*2*) (*1*/*2*), *rat5 0 0*, *rat5* (−*1*) *0*], *vector*[*rat5 0 0*, *rat5* (−*1*) *0*, *rat5* (−*1*/*2*) (−*1*/*2*)], *vector*[*rat5 0 0*, *rat5 1 0*, *rat5* (−*1*/*2*) (−*1*/*2*)]},

convex hull {*vector*[*rat5* (*1*/*2*) (*1*/*2*), *rat5 0 0*, *rat5 1 0*], *vector*[*rat5 1 0*, *rat5* (−*1*/*2*) (−*1*/*2*), *rat5 0 0*], *vector*[*rat5 0 0*, *rat5* (−*1*) *0*, *rat5* (*1*/*2*) (*1*/*2*)]},

convex hull {*vector*[*rat5* (*1*/*2*) (*1*/*2*), *rat5 0 0*, *rat5 1 0*], *vector*[*rat5 1 0*, *rat5* (*1*/*2*) (*1*/*2*), *rat5 0 0*], *vector*[*rat5 0 0*, *rat5 1 0*, *rat5* (*1*/*2*) (*1*/*2*)]},

convex hull {*vector*[*rat5* (*1*/*2*) (*1*/*2*), *rat5 0 0*, *rat5 1 0*], *vector*[*rat5 0 0*, *rat5* (−*1*) *0*, *rat5* (*1*/*2*) (*1*/*2*)], *vector*[*rat5 0 0*, *rat5 1 0*, *rat5* (*1*/*2*) (*1*/*2*)]},

convex hull {*vector*[*rat5* (−*1*) *0*, *rat5* (−*1*/*2*) (−*1*/*2*), *rat5 0 0*], *vector*[*rat5 1 0*, *rat5* (−*1*/*2*) (−*1*/*2*), *rat5 0 0*], *vector*[*rat5 0 0*, *rat5* (−*1*) *0*, *rat5* (−*1*/*2*) (−*1*/*2*)]},

convex hull {*vector*[*rat5* (−*1*) *0*, *rat5* (−*1*/*2*) (−*1*/*2*), *rat5 0 0*], *vector*[*rat5 1 0*, *rat5* (−*1*/*2*) (−*1*/*2*), *rat5 0 0*], *vector*[*rat5 0 0*, *rat5* (−*1*) *0*, *rat5* (*1*/*2*) (*1*/*2*)]},

convex hull {*vector*[*rat5* (−*1*) *0*, *rat5* (*1*/*2*) (*1*/*2*), *rat5 0 0*], *vector*[*rat5 1 0*, *rat5* (*1*/*2*) (*1*/*2*), *rat5 0 0*], *vector*[*rat5 0 0*, *rat5 1 0*, *rat5* (−*1*/*2*) (−*1*/*2*)]},

convex hull {*vector*[*rat5* (−*1*) *0*, *rat5* (*1*/*2*) (*1*/*2*), *rat5 0 0*], *vector*[*rat5 1 0*, *rat5* (*1*/*2*) (*1*/*2*), *rat5 0 0*], *vector*[*rat5 0 0*, *rat5 1 0*, *rat5* (*1*/*2*) (*1*/*2*)]}}

have *f1* ∈ *?faces f2* ∈ *?faces*
using *assms icosahedron-facets*
by *auto*
moreover have ∀ *f*∈*?faces.*
convex hull {(*vector*[*rat5* (−*1*/*2*) (−*1*/*2*), *rat5 0 0*, *rat5* (−*1*) *0*]::*real^3*), *vector*[*rat5* (−*1*/*2*) (−*1*/*2*), *rat5 0 0*, *rat5 1 0*], *vector*[*rat5* (−*1*) *0*, *rat5* (−*1*/*2*) (−*1*/*2*), *rat5 0 0*]} *congruent f*
apply (*simp only*: *ball-simps*(*7*) *Set.ball-empty simp-thms*(*21*))
unfolding *conj-assoc*[*symmetric*]
by (*rule conjI*)+ *show-congruence*+
ultimately show *?thesis*
using *congruent-set* **by** *meson*
qed

lemma *icosahedron-facet1-equiangular*:
defines *v1* ≡ *vector*[*rat5* (−*1*/*2*) (−*1*/*2*), *rat5 0 0*, *rat5* (−*1*) *0*] :: *real^3*
defines *v2* ≡ *vector*[*rat5* (−*1*/*2*) (−*1*/*2*), *rat5 0 0*, *rat5 1 0*] :: *real^3*
defines *v3* ≡ *vector*[*rat5* (−*1*) *0*, *rat5* (−*1*/*2*) (−*1*/*2*), *rat5 0 0*] :: *real^3*
shows *equiangular* (*convex hull* {*v1*, *v2*, *v3*}) (*pi* / *3*)
unfolding *equiangular-def*
proof (*clarify*)
fix *e1 e2 v*
assume ∗: *e1 face-of convex hull* {*v1*, *v2*, *v3*} *aff-dim e1* = *1*
e2 face-of convex hull {*v1*, *v2*, *v3*} *aff-dim e2* = *1*
e1 ≠ *e2 v extreme-point-of e1 v extreme-point-of e2*
have *rat5 4 0* ≥ *0 rat5* (*1*/*2*) *0* = *1*/*2*
unfolding *rat5-def*
by (*simp-all add*: *of-rat-divide*)
have (*v* = *v1* ∧ ((*e1* = *convex hull* {*v1*, *v2*} ∧ *e2* = *convex hull* {*v1*, *v3*}) ∨
(*e1* = *convex hull* {*v1*, *v3*} ∧ *e2* = *convex hull* {*v1*, *v2*}))) ∨
(*v* = *v2* ∧ ((*e1* = *convex hull* {*v2*, *v1*} ∧ *e2* = *convex hull* {*v2*, *v3*}) ∨
(*e1* = *convex hull* {*v2*, *v3*} ∧ *e2* = *convex hull* {*v2*, *v1*}))) ∨
(*v* = *v3* ∧ ((*e1* = *convex hull* {*v3*, *v1*} ∧ *e2* = *convex hull* {*v3*, *v2*}) ∨
(*e1* = *convex hull* {*v3*, *v2*} ∧ *e2* = *convex hull* {*v3*, *v1*})))
using *conjI*[*OF* ∗(*1*,*2*)] *conjI*[*OF* ∗(*3*,*4*)] ∗(*5*−*7*) **unfolding** *assms icosahedron-facet1-edges*
by (*smt* (*verit*, *best*) *extreme-point-of-convex-hull-2 insert-commute vector3-eq*)
moreover have *vangle* (*v2* − *v1*) (*v3* − *v1*) = *pi* / *3*
vangle (*v1* − *v2*) (*v3* − *v2*) = *pi* / *3*
vangle (*v1* − *v3*) (*v2* − *v3*) = *pi* / *3*
unfolding *assms vector3-sub norm-eq-sqrt-inner vector3-dot vangle-def rat5-sub rat5-mul*
by (*simp-all add*: *vector3-eq-0 rat5-sub rat5-mul rat5-add rat5-eq-0 rat5-div* ‹*rat5 4 0* ≥ *0*› ‹*rat5* (*1*/*2*) *0* = *1*/*2*›)
ultimately show (∃ *a b. e1* = *convex hull* {*v*, *a*} ∧ *e2* = *convex hull* {*v*, *b*} ∧
vangle (*a* − *v*) (*b* − *v*) = *pi* / *3*)
using *vangle-commute*
by *metis*
qed

lemma *icosahedron-equiangular*:
  assumes *f face-of std-icosahedron* $\wedge$ *aff-dim f* = *2*
  shows *equiangular f* (*pi* / *3*)
proof –
  define *f1* **where** *f1* = *convex hull* {(*vector*[*rat5* (−*1*/*2*) (−*1*/*2*), *rat5 0 0*, *rat5* (−*1*) *0*]::*real*^*3*), *vector*[*rat5* (−*1*/*2*) (−*1*/*2*), *rat5 0 0*, *rat5 1 0*], *vector*[*rat5* (−*1*) *0*, *rat5* (−*1*/*2*) (−*1*/*2*), *rat5 0 0*]}
  have *f1 face-of std-icosahedron* $\wedge$ *aff-dim f1* = *2*
    using *f1-def icosahedron-facets* **by** *blast*
  then have *f1 congruent f*
    using *assms icosahedron-congruent-faces* **by** *blast*
  then show *?thesis*
    using *equiangular-congruent icosahedron-facet1-equiangular f1-def*
    by *blast*
qed

end

10 Main result

theory *Platonic-Solids*
  imports
    Euler-Polyhedron-Formula.Euler-Formula
    Polytope-More
    Tetrahedron
    Cube
    Octahedron
    Dodecahedron
    Icosahedron
begin

10.1 Possible Schläfli symbols

lemma *platonic-solid-limits*:
  fixes *p* :: (*real*^*3*) *set*
  assumes *is-polytope*: *polytope p*
    and *aff-dim-p*: *aff-dim p* = *3*
    and *edges-per-face*: $\forall$ *f*. *f face-of p* $\wedge$ *aff-dim f* = *2* $\longrightarrow$
        card {*e*. *e face-of p* $\wedge$ *aff-dim e* = *1* $\wedge$ *e* $\subseteq$ *f*} = *m*
    and *edges-per-vert*: $\forall$ *v*. *v face-of p* $\wedge$ *aff-dim v* = *0* $\longrightarrow$
        card {*e*. *e face-of p* $\wedge$ *aff-dim e* = *1* $\wedge$ *v* $\subseteq$ *e*} = *n*
  shows (*m* = *3* $\wedge$ *n* = *3*) $\vee$
      (*m* = *4* $\wedge$ *n* = *3*) $\vee$
      (*m* = *3* $\wedge$ *n* = *4*) $\vee$
      (*m* = *5* $\wedge$ *n* = *3*) $\vee$
      (*m* = *3* $\wedge$ *n* = *5*)
proof –
  let *?V* = *card* {*v*. *v face-of p* $\wedge$ *aff-dim v* = *0*}

let *?E = card {e. e face-of p ∧ aff-dim e = 1}*
let *?F = card {f. f face-of p ∧ aff-dim f = 2}*
have *?V + ?F − ?E = 2*
using *Euler-relation[OF is-polytope aff-dim-p]* **by** *fastforce*
then have *?V + ?F = ?E + 2*
by *linarith*
moreover have *?V ≥ 4*
using *polytope-vertex-lower-bound[OF is-polytope] aff-dim-p*
by *fastforce*
moreover have *?F ≥ 4*
proof −
have *f facet-of p ⟷ f face-of p ∧ aff-dim f = 2* **for** *f*
unfolding *facet-of-def aff-dim-p* **by** *auto*
then show *?thesis*
using *polytope-facet-lower-bound[OF is-polytope] aff-dim-p* **by** *presburger*
qed
ultimately have *?E ≥ 6*
by *linarith*
have *⋀f. f face-of p ⟹ aff-dim f = 2 ⟹ card {e. e face-of p ∧ aff-dim e = 1 ∧ e ⊆ f} ≥ 3*
proof −
fix *f*
assume *∗: f face-of p aff-dim f = 2*
then have *f-polytope: polytope f*
using *face-of-polytope-polytope is-polytope*
by *blast*
have *e facet-of f ⟷ e face-of p ∧ aff-dim e = 1 ∧ e ⊆ f* **for** *e*
using *face-of-face[OF ∗(1)]* **unfolding** *facet-of-def ∗(2)*
by *auto*
then show *card {e. e face-of p ∧ aff-dim e = 1 ∧ e ⊆ f} ≥ 3*
using *polytope-facet-lower-bound[OF f-polytope] ∗(2)*
by *presburger*
qed
then have *m ≥ 3*
using *edges-per-face*
by *(metis (mono-tags, lifting) Collect-empty-eq One-nat-def ‹?F ≥ 4› add-Suc-shift card.empty linorder-not-le numeral-Bit0 one-add-one plus-nat.add-0 zero-less-Suc)*

have *m ∗ ?F = 2 ∗ ?E*
using *edges-per-face facet-ridge-relation[OF is-polytope]* **unfolding** *aff-dim-p*
by *simp*
moreover have *n ∗ ?V = 2 ∗ ?E*
using *edge-vertex-relation[OF is-polytope edges-per-vert]* **.**
moreover have *m ∗ (n ∗ ?V) + n ∗ (m ∗ ?F) = m ∗ n ∗ (?E + 2)*
using *‹?V + ?F = ?E + 2›* **by** *algebra*
ultimately have *E-m-n-relation: 2 ∗ ?E ∗ (m + n) = m ∗ n ∗ (?E + 2)*
by *algebra*

```
  have n ≠ 0
    using ‹?E ≥ 6› E-m-n-relation ‹3 ≤ m› rel-simps(28)
    by fastforce
  moreover have n ≠ 1
    using ‹?F ≥ 4› ‹?V + ?F = ?E + 2› ‹n * ?V = 2 * ?E› by force
  moreover have n ≠ 2
    using E-m-n-relation ‹m * ?F = 2 * ?E› ‹?E ≥ 6› ‹?F ≥ 4› by force
  ultimately have n ≥ 3
    by auto

  have ?E * (m * n) < ?E * (2 * (m + n))
    using ‹3 ≤ m› ‹n ≠ 0› E-m-n-relation
    by (simp add: algebra-simps)
  then have m * n < 2 * (m + n)
    by auto
  then have int m * int n − 2 * int m − 2 * int n < 0
     by (smt (verit, ccfv-SIG) diff-add-cancel diff-less-0-iff-less group-cancel.add2
mult-2
        of-nat-add of-nat-less-iff of-nat-mult)
  then have int m * (int n − 2) − 2 * int n < 0
    by (simp add: right-diff-distrib')
  then have int m * (int n − 2) − 2 * (int n − 2) < 4
    by simp
  then have int-m-n-ineq: (int m − 2) * (int n − 2) < 4
    using int-distrib(3) by presburger
  have (m − 2) * (int n − 2) < 4
    using int-m-n-ineq ‹3 ≤ m› by simp
  then have int ((m − 2) * (n − 2)) < int 4
    by (simp add: of-nat-diff-if)
  then have m-n-ineq: (m − 2) * (n − 2) < 4
    by linarith

  have m − 2 ≥ 1 n − 2 ≥ 1
    using ‹3 ≤ m› ‹3 ≤ n› by auto

  have m ≤ 5
    proof (rule ccontr)
      assume ¬(m ≤ 5)
      then have *: m − 2 ≥ 4 by simp
      have (m − 2) * (n − 2) ≥ 4
        using mult-le-mono[OF * ‹n − 2 ≥ 1›] by auto
      then show False
        using m-n-ineq by linarith
    qed
  have n ≤ 5
  proof (rule ccontr)
    assume ¬(n ≤ 5)
    then have *: n − 2 ≥ 4 by simp
    have (m − 2) * (n − 2) ≥ 4
```

using *mult-le-mono*[*OF* ‹$m - 2 \geq 1$› $*$] **by** *auto*
then show *False*
using *m-n-ineq* **by** *linarith*
qed

have $m = 3 \lor m = 4 \lor m = 5$
using ‹$3 \leq m$› ‹$m \leq 5$› **by** *linarith*
moreover have $n = 3 \lor n = 4 \lor n = 5$
using ‹$3 \leq n$› ‹$n \leq 5$› **by** *linarith*
ultimately show *?thesis*
using *m-n-ineq* **by** *presburger*
qed

10.2 Exactly five Platonic solids

theorem *platonic-solids-full*:
shows ($\exists$ *p*::(*real*^*3*) *set. polytope p* $\land$ *aff-dim p* $= 3 \land$
($\forall$ *f. f face-of p* $\land$ *aff-dim f* $= 2 \longrightarrow$
card {*e. e face-of p* $\land$ *aff-dim e* $= 1 \land e \subseteq f$} $= m) \land$
($\forall$ *v. v face-of p* $\land$ *aff-dim v* $= 0 \longrightarrow$
card {*e. e face-of p* $\land$ *aff-dim e* $= 1 \land v \subseteq e$} $= n) \land$
($\forall$ *f1 f2. f1 face-of p* $\land$ *aff-dim f1* $= 2 \land$
f2 face-of p $\land$ *aff-dim f2* $= 2 \longrightarrow$ *f1 congruent f2*) $\land$
($\forall$ *e1 e2. e1 face-of p* $\land$ *aff-dim e1* $= 1 \land$
e2 face-of p $\land$ *aff-dim e2* $= 1 \longrightarrow$ *e1 congruent e2*) $\land$
($\exists$ *t*. $\forall$ *f. f face-of p* $\land$ *aff-dim f* $= 2 \longrightarrow$ *equiangular f t*)) $=$
$((m = 3 \land n = 3) \lor$
$(m = 4 \land n = 3) \lor$
$(m = 3 \land n = 4) \lor$
$(m = 5 \land n = 3) \lor$
$(m = 3 \land n = 5))$
(**is** *?lhs* $=$ *?rhs*)
proof
assume *?lhs*
then show *?rhs*
using *platonic-solid-limits* **by** *auto*
next
assume *?rhs*
then consider
(*m3-n3*) $m = 3 \land n = 3$ |
(*m3-n4*) $m = 3 \land n = 4$ |
(*m3-n5*) $m = 3 \land n = 5$ |
(*m4-n3*) $m = 4 \land n = 3$ |
(*m5-n3*) $m = 5 \land n = 3$
by *fast*
then show *?lhs*
proof (*cases*)
case *m3-n3*
moreover have *polytope std-tetrahedron*

by (*simp add*: *polytope-convex-hull std-tetrahedron-def*)
ultimately show *?thesis*
using *tetrahedron-fulldim tetrahedron-edges-per-face tetrahedron-edges-per-vertex tetrahedron-congruent-edges tetrahedron-congruent-faces tetrahedron-equiangular*
by *blast*
next
case *m3-n4*
moreover have *polytope std-octahedron*
by (*simp add*: *polytope-convex-hull std-octahedron-def*)
ultimately show *?thesis*
using *octahedron-fulldim octahedron-edges-per-face octahedron-edges-per-vertex octahedron-congruent-edges octahedron-congruent-faces octahedron-equiangular*
by *blast*
next
case *m3-n5*
moreover have *polytope std-icosahedron*
by (*simp add*: *polytope-convex-hull std-icosahedron-def*)
ultimately show *?thesis*
using *icosahedron-fulldim icosahedron-edges-per-face icosahedron-edges-per-vertex icosahedron-congruent-edges icosahedron-congruent-faces icosahedron-equiangular*
by *blast*
next
case *m4-n3*
moreover have *polytope std-cube*
by (*simp add*: *polytope-convex-hull std-cube-def*)
ultimately show *?thesis*
using *cube-fulldim cube-edges-per-face cube-edges-per-vertex cube-congruent-edges cube-congruent-faces cube-equiangular*
by *blast*
next
case *m5-n3*
moreover have *polytope std-dodecahedron*
by (*simp add*: *polytope-convex-hull std-dodecahedron-def*)
ultimately show *?thesis*
using *dodecahedron-fulldim dodecahedron-edges-per-face dodecahedron-edges-per-vertex dodecahedron-congruent-edges dodecahedron-congruent-faces dodecahedron-equiangular*
by *blast*
qed
qed

This is the original top-level statement of the HOL Light formalization, kept here for parity. It omits the regularity condition on *p*.

theorem *platonic-solids*:
shows $(\exists\, p{::}(real\hat{\ }3)\ set.\ polytope\ p \wedge aff\text{-}dim\ p = 3 \wedge$
$(\forall f.\ f\ face\text{-}of\ p \wedge aff\text{-}dim\ f = 2 \longrightarrow$
$card\ \{e.\ e\ face\text{-}of\ p \wedge aff\text{-}dim\ e = 1 \wedge e \subseteq f\} = m) \wedge$
$(\forall v.\ v\ face\text{-}of\ p \wedge aff\text{-}dim\ v = 0 \longrightarrow$
$card\ \{e.\ e\ face\text{-}of\ p \wedge aff\text{-}dim\ e = 1 \wedge v \subseteq e\} = n)) =$
$((m = 3 \wedge n = 3) \vee$

$(m = 4 \wedge n = 3) \vee$
$(m = 3 \wedge n = 4) \vee$
$(m = 5 \wedge n = 3) \vee$
$(m = 3 \wedge n = 5))$
(**is** *?lhs* = *?rhs*)
proof
assume *?lhs*
then show *?rhs*
using *platonic-solid-limits* **by** *auto*
next
assume *?rhs*
then show *?lhs*
unfolding *platonic-solids-full*[*symmetric*]
by *meson*
qed

end

References

[1] J. Harrison. Proof of the Platonic solids theorem, accessed 2025. HOL Light repository.