

Euler's Polyhedron Formula

Lawrence C. Paulson

February 6, 2026

Abstract

Euler stated in 1752 that every convex polyhedron satisfied the formula $V - E + F = 2$ where V , E and F are the numbers of its vertices, edges, and faces. For three dimensions, the well-known proof involves removing one face and then flattening the remainder to form a planar graph, which then is iteratively transformed to leave a single triangle. The history of that proof is extensively discussed and elaborated by Imre Lakatos [1], leaving one finally wondering whether the theorem even holds. The formal proof provided here has been ported from HOL Light, where it is credited to Lawrence [2]. The proof generalises Euler's observation from solid polyhedra to convex polytopes of arbitrary dimension.

Contents

1 Euler's Polyhedron Formula	3
1.1 Cells of a hyperplane arrangement	3
1.2 A cell complex is considered to be a union of such cells	7
1.3 Euler characteristic	10
1.4 Show that the characteristic is invariant w.r.t. hyperplane arrangement.	12
1.5 Euler-type relation for full-dimensional proper polyhedral cones	17
1.6 Euler-Poincare relation for special $(n - 1)$ -dimensional polytope	29
1.7 Now Euler-Poincare for a general full-dimensional polytope .	38

Acknowledgements The author was supported by the ERC Advanced Grant ALEXANDRIA (Project 742178) funded by the European Research Council.

1 Euler's Polyhedron Formula

One of the Famous 100 Theorems, ported from HOL Light

Cited source: Lawrence, J. (1997). A Short Proof of Euler's Relation for Convex Polytopes. *Canadian Mathematical Bulletin*, **40**(4), 471–474.

theory *Euler-Formula*

imports

HOL-Analysis.Analysis

begin

Interpret which "side" of a hyperplane a point is on.

definition *hyperplane-side*

where *hyperplane-side* $\equiv \lambda(a,b). \lambda x. \text{sgn } (a \cdot x - b)$

Equivalence relation imposed by a hyperplane arrangement.

definition *hyperplane-equiv*

where *hyperplane-equiv* $\equiv \lambda A x y. \forall h \in A. \text{hyperplane-side } h x = \text{hyperplane-side } h y$

lemma *hyperplane-equiv-refl* [*iff*]: *hyperplane-equiv* $A x x$

by (*simp add: hyperplane-equiv-def*)

lemma *hyperplane-equiv-sym*:

hyperplane-equiv $A x y \longleftrightarrow \text{hyperplane-equiv } A y x$

by (*auto simp: hyperplane-equiv-def*)

lemma *hyperplane-equiv-trans*:

$\llbracket \text{hyperplane-equiv } A x y; \text{hyperplane-equiv } A y z \rrbracket \Longrightarrow \text{hyperplane-equiv } A x z$

by (*auto simp: hyperplane-equiv-def*)

lemma *hyperplane-equiv-Un*:

hyperplane-equiv $(A \cup B) x y \longleftrightarrow \text{hyperplane-equiv } A x y \wedge \text{hyperplane-equiv } B x y$

by (*meson Un-iff hyperplane-equiv-def*)

1.1 Cells of a hyperplane arrangement

definition *hyperplane-cell* :: $(\text{'a}::\text{real-inner} \times \text{real}) \text{ set} \Rightarrow \text{'a set} \Rightarrow \text{bool}$

where *hyperplane-cell* $\equiv \lambda A C. \exists x. C = \text{Collect } (\text{hyperplane-equiv } A x)$

lemma *hyperplane-cell*: *hyperplane-cell* $A C \longleftrightarrow (\exists x. C = \{y. \text{hyperplane-equiv } A x y\})$

by (*simp add: hyperplane-cell-def*)

lemma *not-hyperplane-cell-empty* [*simp*]: $\neg \text{hyperplane-cell } A \{\}$

using *hyperplane-cell* **by** *auto*

lemma *nonempty-hyperplane-cell*: *hyperplane-cell* $A C \Longrightarrow (C \neq \{\})$

by auto

lemma *Union-hyperplane-cells*: $\bigcup \{C. \text{hyperplane-cell } A \ C\} = \text{UNIV}$
using *hyperplane-cell* by blast

lemma *disjoint-hyperplane-cells*:
[[*hyperplane-cell* $A \ C1$; *hyperplane-cell* $A \ C2$; $C1 \neq C2$]] $\implies \text{disjnt } C1 \ C2$
by (*force simp: hyperplane-cell-def disjnt-iff hyperplane-equiv-def*)

lemma *disjoint-hyperplane-cells-eq*:
[[*hyperplane-cell* $A \ C1$; *hyperplane-cell* $A \ C2$]] $\implies (\text{disjnt } C1 \ C2 \longleftrightarrow (C1 \neq C2))$
using *disjoint-hyperplane-cells* by auto

lemma *hyperplane-cell-empty* [*iff*]: *hyperplane-cell* $\{\} \ C \longleftrightarrow C = \text{UNIV}$
by (*simp add: hyperplane-cell hyperplane-equiv-def*)

lemma *hyperplane-cell-singleton-cases*:
assumes *hyperplane-cell* $\{(a,b)\} \ C$
shows $C = \{x. a \cdot x = b\} \vee C = \{x. a \cdot x < b\} \vee C = \{x. a \cdot x > b\}$
proof –
obtain x where $x: C = \{y. \text{hyperplane-side } (a, b) \ x = \text{hyperplane-side } (a, b) \ y\}$
using *assms* by (*auto simp: hyperplane-equiv-def hyperplane-cell*)
then show ?thesis
by (*auto simp: hyperplane-side-def sgn-if split: if-split-asm*)
qed

lemma *hyperplane-cell-singleton*:
hyperplane-cell $\{(a,b)\} \ C \longleftrightarrow$
 $(\text{if } a = 0 \text{ then } C = \text{UNIV} \text{ else } C = \{x. a \cdot x = b\} \vee C = \{x. a \cdot x < b\} \vee C = \{x. a \cdot x > b\})$
apply (*simp add: hyperplane-cell-def hyperplane-equiv-def hyperplane-side-def sgn-if split: if-split-asm*)
by (*smt (verit) Collect-cong gt-ex hyperplane-eq-Ex lt-ex*)

lemma *hyperplane-cell-Un*:
hyperplane-cell $(A \cup B) \ C \longleftrightarrow$
 $C \neq \{\} \wedge$
 $(\exists C1 \ C2. \text{hyperplane-cell } A \ C1 \wedge \text{hyperplane-cell } B \ C2 \wedge C = C1 \cap C2)$
by (*auto simp: hyperplane-cell hyperplane-equiv-def*)

lemma *finite-hyperplane-cells*:
finite $A \implies \text{finite } \{C. \text{hyperplane-cell } A \ C\}$
proof (*induction rule: finite-induct*)
case (*insert p A*)
obtain $a \ b$ where *peq*: $p = (a, b)$
by *fastforce*
have *Collect* (*hyperplane-cell* $\{p\}$) $\subseteq \{\{x. a \cdot x = b\}, \{x. a \cdot x < b\}, \{x. a \cdot x > b\}\}$

```

    using hyperplane-cell-singleton-cases
    by (auto simp: peq)
  then have *: finite (Collect (hyperplane-cell {p}))
    by (simp add: finite-subset)
  define C where C ≡ (⋃ C1 ∈ {C. hyperplane-cell A C}. ⋃ C2 ∈ {C. hyper-
plane-cell {p} C}. {C1 ∩ C2})
  have {a. hyperplane-cell (insert p A) a} ⊆ C
    using hyperplane-cell-Un [of {p} A] by (auto simp: C-def)
  moreover have finite C
    using * C-def insert.IH by blast
  ultimately show ?case
    using finite-subset by blast
qed auto

```

```

lemma finite-restrict-hyperplane-cells:
  finite A ⇒ finite {C. hyperplane-cell A C ∧ P C}
  by (simp add: finite-hyperplane-cells)

```

```

lemma finite-set-of-hyperplane-cells:
  [[finite A; ∧ C. C ∈ C ⇒ hyperplane-cell A C]] ⇒ finite C
  by (metis finite-hyperplane-cells finite-subset mem-Collect-eq subsetI)

```

```

lemma pairwise-disjoint-hyperplane-cells:
  (∧ C. C ∈ C ⇒ hyperplane-cell A C) ⇒ pairwise disjoint C
  by (metis disjoint-hyperplane-cells pairwiseI)

```

```

lemma hyperplane-cell-Int-open-affine:
  assumes finite A hyperplane-cell A C
  obtains S T where open S affine T C = S ∩ T
  using assms
proof (induction arbitrary: thesis C rule: finite-induct)
  case empty
  then show ?case
    by auto
next
  case (insert p A thesis C')
  obtain a b where peq: p = (a,b)
    by fastforce
  obtain C C1 where C1: hyperplane-cell {(a,b)} C1 and C: hyperplane-cell A
C
    and C' ≠ {} and C': C' = C1 ∩ C
  by (metis hyperplane-cell-Un insert.prem(2) insert-is-Un peq)
  then obtain S T where ST: open S affine T C = S ∩ T
    by (meson insert.IH)
  show ?case
proof (cases a=0)
  case True
  with insert.prem show ?thesis
    by (metis C1 Int-commute ST ⟨C' = C1 ∩ C⟩ hyperplane-cell-singleton

```

```

inf-top.right-neutral)
next
  case False
  then consider C1 = {x. a · x = b} | C1 = {x. a · x < b} | C1 = {x. b < a
· x}
  by (metis C1 hyperplane-cell-singleton)
  then show ?thesis
  proof cases
  case 1
  then show thesis
  by (metis C' ST affine-Int affine-hyperplane inf-left-commute insert.prem(1))
next
  case 2
  with ST show thesis
  by (metis Int-assoc C' insert.prem(1) open-Int open-halfspace-lt)
next
  case 3
  with ST show thesis
  by (metis Int-assoc C' insert.prem(1) open-Int open-halfspace-gt)
qed
qed
qed

```

```

lemma hyperplane-cell-relatively-open:
  assumes finite A hyperplane-cell A C
  shows openin (subtopology euclidean (affine hull C)) C
proof -
  obtain S T where open S affine T C = S ∩ T
  by (meson assms hyperplane-cell-Int-open-affine)
  show ?thesis
  proof (cases S ∩ T = {})
  case True
  then show ?thesis
  by (simp add: ⟨C = S ∩ T⟩)
next
  case False
  then have affine hull (S ∩ T) = T
  by (metis ⟨affine T⟩ ⟨open S⟩ affine-hull-affine-Int-open hull-same inf-commute)
  then show ?thesis
  using ⟨C = S ∩ T⟩ ⟨open S⟩ openin-subtopology by fastforce
qed
qed

```

```

lemma hyperplane-cell-relative-interior:
  [[finite A; hyperplane-cell A C]] ==> rel-interior C = C
  by (simp add: hyperplane-cell-relatively-open rel-interior-openin)

```

```

lemma hyperplane-cell-convex:
  assumes hyperplane-cell A C

```

shows *convex C*
proof –
obtain *c* **where** $c: C = \{y. \text{hyperplane-equiv } A \ c \ y\}$
by (*meson assms hyperplane-cell*)
have *convex* $(\bigcap h \in A. \{y. \text{hyperplane-side } h \ c = \text{hyperplane-side } h \ y\})$
proof (*rule convex-INT*)
fix $h :: 'a \times \text{real}$
assume $h \in A$
obtain $a \ b$ **where** $heq: h = (a, b)$
by *fastforce*
have [*simp*]: $\{y. \neg a \cdot c < a \cdot y \wedge a \cdot y = a \cdot c\} = \{y. a \cdot y = a \cdot c\}$
 $\{y. \neg b < a \cdot y \wedge a \cdot y \neq b\} = \{y. b > a \cdot y\}$
by *auto*
then show *convex* $\{y. \text{hyperplane-side } h \ c = \text{hyperplane-side } h \ y\}$
by (*fastforce simp: heq hyperplane-side-def sgn-if convex-halfspace-gt convex-halfspace-lt convex-hyperplane cong: conj-cong*)
qed
with *c* **show** *?thesis*
by (*simp add: hyperplane-equiv-def INTER-eq*)
qed

lemma *hyperplane-cell-Inter*:
assumes $\bigwedge C. C \in \mathcal{C} \implies \text{hyperplane-cell } A \ C$
and $C \neq \{\}$ **and** *INT*: $\bigcap C \neq \{\}$
shows *hyperplane-cell* $A \ (\bigcap C)$
proof –
have $\bigcap C = \{y. \text{hyperplane-equiv } A \ z \ y\}$
if $z \in \bigcap C$ **for** z
using *assms that* **by** (*force simp: hyperplane-cell hyperplane-equiv-def*)
with *INT hyperplane-cell* **show** *?thesis*
by *fastforce*
qed

lemma *hyperplane-cell-Int*:
 $\llbracket \text{hyperplane-cell } A \ S; \text{hyperplane-cell } A \ T; S \cap T \neq \{\} \rrbracket \implies \text{hyperplane-cell } A \ (S \cap T)$
by (*metis hyperplane-cell-Un sup.idem*)

1.2 A cell complex is considered to be a union of such cells

definition *hyperplane-cellcomplex*
where *hyperplane-cellcomplex* $A \ S \equiv$
 $\exists \mathcal{T}. (\forall C \in \mathcal{T}. \text{hyperplane-cell } A \ C) \wedge S = \bigcup \mathcal{T}$

lemma *hyperplane-cellcomplex-empty* [*simp*]: *hyperplane-cellcomplex* $A \ \{\}$
using *hyperplane-cellcomplex-def* **by** *auto*

lemma *hyperplane-cell-cellcomplex*:

hyperplane-cell $A C \implies \text{hyperplane-cellcomplex } A C$
by (*auto simp: hyperplane-cellcomplex-def*)

lemma *hyperplane-cellcomplex-Union:*

assumes $\bigwedge S. S \in \mathcal{C} \implies \text{hyperplane-cellcomplex } A S$
shows *hyperplane-cellcomplex* $A (\bigcup \mathcal{C})$

proof –

obtain \mathcal{F} **where** $\mathcal{F}: \bigwedge S. S \in \mathcal{C} \implies (\forall C \in \mathcal{F} S. \text{hyperplane-cell } A C) \wedge S = \bigcup (\mathcal{F} S)$

by (*metis assms hyperplane-cellcomplex-def*)

show *?thesis*

unfolding *hyperplane-cellcomplex-def*

using \mathcal{F} **by** (*fastforce intro: exI [where x= $\bigcup (\mathcal{F} \text{ ` } \mathcal{C})$]*)

qed

lemma *hyperplane-cellcomplex-Un:*

$\llbracket \text{hyperplane-cellcomplex } A S; \text{hyperplane-cellcomplex } A T \rrbracket$
 $\implies \text{hyperplane-cellcomplex } A (S \cup T)$

by (*smt (verit) Un-iff Union-Un-distrib hyperplane-cellcomplex-def*)

lemma *hyperplane-cellcomplex-UNIV [simp]: hyperplane-cellcomplex* $A UNIV$

by (*metis Union-hyperplane-cells hyperplane-cellcomplex-def mem-Collect-eq*)

lemma *hyperplane-cellcomplex-Inter:*

assumes $\bigwedge S. S \in \mathcal{C} \implies \text{hyperplane-cellcomplex } A S$
shows *hyperplane-cellcomplex* $A (\bigcap \mathcal{C})$

proof (*cases* $\mathcal{C} = \{\}$)

case *True*

then show *?thesis*

by *simp*

next

case *False*

obtain \mathcal{F} **where** $\mathcal{F}: \bigwedge S. S \in \mathcal{C} \implies (\forall C \in \mathcal{F} S. \text{hyperplane-cell } A C) \wedge S = \bigcup (\mathcal{F} S)$

by (*metis assms hyperplane-cellcomplex-def*)

have $*$: $\mathcal{C} = (\lambda S. \bigcup (\mathcal{F} S)) \text{ ` } \mathcal{C}$

using \mathcal{F} **by** *force*

define U **where** $U \equiv \bigcup \{T \in \{\bigcap (g \text{ ` } \mathcal{C}) \mid g. \forall S \in \mathcal{C}. g S \in \mathcal{F} S\}. T \neq \{\}\}$

have $\bigcap \mathcal{C} = \bigcup \{\bigcap (g \text{ ` } \mathcal{C}) \mid g. \forall S \in \mathcal{C}. g S \in \mathcal{F} S\}$

using *False* \mathcal{F} **unfolding** *Inter-over-Union [symmetric]*

by *blast*

also have $\dots = U$

unfolding *U-def*

by *blast*

finally have $\bigcap \mathcal{C} = U$.

have *hyperplane-cellcomplex* $A U$

using *False* \mathcal{F} **unfolding** *U-def*

apply (*intro hyperplane-cellcomplex-Union hyperplane-cell-cellcomplex*)

by (*auto intro!: hyperplane-cell-Inter*)

then show *?thesis*
by (*simp add: ⟨ $\bigcap C = U$ ⟩*)
qed

lemma *hyperplane-cellcomplex-Int:*
 $\llbracket \text{hyperplane-cellcomplex } A \ S; \text{ hyperplane-cellcomplex } A \ T \rrbracket$
 $\implies \text{hyperplane-cellcomplex } A \ (S \cap T)$
using *hyperplane-cellcomplex-Inter* [*of* $\{S, T\}$] **by force**

lemma *hyperplane-cellcomplex-Compl:*
assumes *hyperplane-cellcomplex* $A \ S$
shows *hyperplane-cellcomplex* $A \ (-S)$

proof –

obtain \mathcal{C} **where** $\mathcal{C}: \bigwedge C. C \in \mathcal{C} \implies \text{hyperplane-cell } A \ C$ **and** $S = \bigcup \mathcal{C}$

by (*meson assms hyperplane-cellcomplex-def*)

have *hyperplane-cellcomplex* $A \ (\bigcap T \in \mathcal{C}. -T)$

proof (*intro hyperplane-cellcomplex-Inter*)

fix $C0$

assume $C0 \in \text{uminus } C$

then obtain C **where** $C: C0 = -C \ C \in \mathcal{C}$

by *auto*

have $*$: $-C = \bigcup \{D. \text{hyperplane-cell } A \ D \wedge D \neq C\}$ (**is** $- = ?rhs$)

proof

show $-C \subseteq ?rhs$

using *hyperplane-cell* **by** *blast*

show $?rhs \subseteq -C$

by *clarify* (*meson ⟨ $C \in \mathcal{C} \rangle C \text{ disjnt-iff disjoint-hyperplane-cells}$ ⟩*)

qed

then show *hyperplane-cellcomplex* $A \ C0$

by (*metis* (*no-types, lifting*) $C(1)$ *hyperplane-cell-cellcomplex hyperplane-cellcomplex-Union*

mem-Collect-eq)

qed

then show *?thesis*

by (*simp add: ⟨ $S = \bigcup C \rangle \text{uminus-Sup}$ ⟩*)

qed

lemma *hyperplane-cellcomplex-diff:*
 $\llbracket \text{hyperplane-cellcomplex } A \ S; \text{ hyperplane-cellcomplex } A \ T \rrbracket$
 $\implies \text{hyperplane-cellcomplex } A \ (S - T)$
using *hyperplane-cellcomplex-Inter* [*of* $\{S, -T\}$]
by (*force simp: Diff-eq hyperplane-cellcomplex-Compl*)

lemma *hyperplane-cellcomplex-mono:*
assumes *hyperplane-cellcomplex* $A \ S \ A \subseteq B$
shows *hyperplane-cellcomplex* $B \ S$

proof –

obtain \mathcal{C} **where** $\mathcal{C}: \bigwedge C. C \in \mathcal{C} \implies \text{hyperplane-cell } A \ C$ **and** $eq: S = \bigcup \mathcal{C}$

by (*meson assms hyperplane-cellcomplex-def*)

show *?thesis*

```

unfolding eq
proof (intro hyperplane-cellcomplex-Union)
  fix C
  assume C ∈ C
  have  $\bigwedge x. x \in C \implies \exists D'. (\exists D. D' = D \cap C \wedge \text{hyperplane-cell } (B - A) D \wedge D \cap C \neq \{\}) \wedge x \in D'$ 
    unfolding hyperplane-cell-def by blast
  then
    have hyperplane-cellcomplex (A ∪ (B - A)) C
    unfolding hyperplane-cellcomplex-def hyperplane-cell-Un
    using C ⟨C ∈ C⟩ by (fastforce intro!: exI [where x = {D ∩ C | D. hyperplane-cell (B - A) D ∧ D ∩ C ≠ {} }])
    moreover have B = A ∪ (B - A)
    using ⟨A ⊆ B⟩ by auto
    ultimately show hyperplane-cellcomplex B C by simp
  qed
qed

```

```

lemma finite-hyperplane-cellcomplexes:
  assumes finite A
  shows finite {C. hyperplane-cellcomplex A C}
proof -
  have {C. hyperplane-cellcomplex A C} ⊆ image ∪ {T. T ⊆ {C. hyperplane-cell A C}}
    by (force simp: hyperplane-cellcomplex-def subset-eq)
  with finite-hyperplane-cells show ?thesis
    by (metis assms finite-Collect-subsets finite-surj)
qed

```

```

lemma finite-restrict-hyperplane-cellcomplexes:
  finite A  $\implies$  finite {C. hyperplane-cellcomplex A C ∧ P C}
  by (simp add: finite-hyperplane-cellcomplexes)

```

```

lemma finite-set-of-hyperplane-cellcomplex:
  assumes finite A ∧ C. C ∈ C  $\implies$  hyperplane-cellcomplex A C
  shows finite C
  by (metis assms finite-hyperplane-cellcomplexes mem-Collect-eq rev-finite-subset subsetI)

```

```

lemma cell-subset-cellcomplex:
   $\llbracket \text{hyperplane-cell } A C; \text{hyperplane-cellcomplex } A S \rrbracket \implies C \subseteq S \longleftrightarrow \sim \text{disjnt } C S$ 
  by (smt (verit) Union-iff disjnt-iff disjnt-subset1 disjoint-hyperplane-cells-eq hyperplane-cellcomplex-def subsetI)

```

1.3 Euler characteristic

```

definition Euler-characteristic :: ('a::euclidean-space × real) set  $\Rightarrow$  'a set  $\Rightarrow$  int
  where Euler-characteristic A S  $\equiv$ 
     $(\sum C \mid \text{hyperplane-cell } A C \wedge C \subseteq S. (-1) \wedge \text{nat } (\text{aff-dim } C))$ 

```

lemma *Euler-characteristic-empty* [simp]: *Euler-characteristic* $A \{\} = 0$
by (*simp add: sum.neutral Euler-characteristic-def*)

lemma *Euler-characteristic-cell-Union*:
assumes $\bigwedge C. C \in \mathcal{C} \implies \text{hyperplane-cell } A \ C$
shows *Euler-characteristic* $A (\bigcup \mathcal{C}) = (\sum C \in \mathcal{C}. (-1) \wedge \text{nat} (\text{aff-dim } C))$
proof –
have $\bigwedge x. [\text{hyperplane-cell } A \ x; x \subseteq \bigcup \mathcal{C}] \implies x \in \mathcal{C}$
by (*metis assms disjnt-Union1 disjnt-subset1 disjoint-hyperplane-cells-eq*)
then have $\{C. \text{hyperplane-cell } A \ C \wedge C \subseteq \bigcup \mathcal{C}\} = \mathcal{C}$
by (*auto simp: assms*)
then show *?thesis*
by (*auto simp: Euler-characteristic-def*)
qed

lemma *Euler-characteristic-cell*:
hyperplane-cell $A \ C \implies \text{Euler-characteristic } A \ C = (-1) \wedge (\text{nat}(\text{aff-dim } C))$
using *Euler-characteristic-cell-Union* [of $\{C\}$] **by** *force*

lemma *Euler-characteristic-cellcomplex-Un*:
assumes *finite* A *hyperplane-cellcomplex* $A \ S$
and AT : *hyperplane-cellcomplex* $A \ T$ **and** *disjnt* $S \ T$
shows *Euler-characteristic* $A (S \cup T) =$
Euler-characteristic $A \ S + \text{Euler-characteristic } A \ T$
proof –
have $*$: $\{C. \text{hyperplane-cell } A \ C \wedge C \subseteq S \cup T\} =$
 $\{C. \text{hyperplane-cell } A \ C \wedge C \subseteq S\} \cup \{C. \text{hyperplane-cell } A \ C \wedge C \subseteq T\}$
using *cell-subset-cellcomplex* [OF - AT] **by** (*auto simp: disjnt-iff*)
have $**$: $\{C. \text{hyperplane-cell } A \ C \wedge C \subseteq S\} \cap \{C. \text{hyperplane-cell } A \ C \wedge C \subseteq$
 $T\} = \{\}$
using *assms cell-subset-cellcomplex disjnt-subset1* **by** *fastforce*
show *?thesis*
unfolding *Euler-characteristic-def*
by (*simp add: finite-restrict-hyperplane-cells assms * ** flip: sum.union-disjoint*)
qed

lemma *Euler-characteristic-cellcomplex-Union*:
assumes *finite* A
and \mathcal{C} : $\bigwedge C. C \in \mathcal{C} \implies \text{hyperplane-cellcomplex } A \ C$ *pairwise disjnt* \mathcal{C}
shows *Euler-characteristic* $A (\bigcup \mathcal{C}) = \text{sum} (\text{Euler-characteristic } A) \ \mathcal{C}$
proof –
have *finite* \mathcal{C}
using *assms finite-set-of-hyperplane-cellcomplex* **by** *blast*
then show *?thesis*
using \mathcal{C}
proof (*induction rule: finite-induct*)
case *empty*
then show *?case*

```

    by auto
  next
  case (insert C C)
  then obtain disjoint C disjnt C (∪ C)
    by (metis disjnt-Union2 pairwise-insert)
  with insert show ?case
  by (simp add: Euler-characteristic-cellcomplex-Un hyperplane-cellcomplex-Union
    ⟨finite A⟩)
  qed
qed

```

lemma *Euler-characteristic:*

```

  fixes A :: ('n::euclidean-space * real) set
  assumes finite A
  shows Euler-characteristic A S =
    (∑ d = 0..DIM('n). (-1) ^ d * int (card {C. hyperplane-cell A C ∧ C ⊆
    S ∧ aff-dim C = int d}))
    (is - = ?rhs)

```

proof –

```

  have ∧ T. [hyperplane-cell A T; T ⊆ S] ⇒ aff-dim T ∈ {0..DIM('n)}
    by (metis atLeastAtMost-iff nle-le order.strict-iff-not aff-dim-negative-iff
      nonempty-hyperplane-cell aff-dim-le-DIM)
  then have *: aff-dim ‘ {C. hyperplane-cell A C ∧ C ⊆ S} ⊆ int ‘ {0..DIM('n)}
    by (auto simp: image-int-atLeastAtMost)
  have Euler-characteristic A S = (∑ y∈int ‘ {0..DIM('n)}.
    ∑ C∈{x. hyperplane-cell A x ∧ x ⊆ S ∧ aff-dim x = y}. (-1) ^ nat y)
    using sum.group [of {C. hyperplane-cell A C ∧ C ⊆ S} int ‘ {0..DIM('n)}
    aff-dim λC. (-1::int) ^ nat(aff-dim C), symmetric]
    by (simp add: asms Euler-characteristic-def finite-restrict-hyperplane-cells *)
  also have ... = ?rhs
    by (simp add: sum.reindex mult-of-nat-commute)
  finally show ?thesis .
qed

```

1.4 Show that the characteristic is invariant w.r.t. hyperplane arrangement.

lemma *hyperplane-cells-distinct-lemma:*

$$\begin{aligned}
 \{x. a \cdot x = b\} \cap \{x. a \cdot x < b\} &= \{\} \wedge \\
 \{x. a \cdot x = b\} \cap \{x. a \cdot x > b\} &= \{\} \wedge \\
 \{x. a \cdot x < b\} \cap \{x. a \cdot x = b\} &= \{\} \wedge \\
 \{x. a \cdot x < b\} \cap \{x. a \cdot x > b\} &= \{\} \wedge \\
 \{x. a \cdot x > b\} \cap \{x. a \cdot x = b\} &= \{\} \wedge \\
 \{x. a \cdot x > b\} \cap \{x. a \cdot x < b\} &= \{\}
 \end{aligned}$$

by *auto*

proposition *Euler-characteristic-lemma:*

```

  assumes finite A and hyperplane-cellcomplex A S
  shows Euler-characteristic (insert h A) S = Euler-characteristic A S

```

```

proof –
  obtain  $C$  where  $C: \bigwedge C. C \in \mathcal{C} \implies \text{hyperplane-cell } A \ C$  and  $S = \bigcup C$ 
    and pairwise disjoint  $C$ 
    by (meson assms hyperplane-cellcomplex-def pairwise-disjoint-hyperplane-cells)
  obtain  $a \ b$  where  $h = (a, b)$ 
    by fastforce
  have  $\bigwedge C. C \in \mathcal{C} \implies \text{hyperplane-cellcomplex } A \ C \wedge \text{hyperplane-cellcomplex}$ 
(insert  $(a, b) \ A$ )  $C$ 
    by (meson  $C$  hyperplane-cell-cellcomplex hyperplane-cellcomplex-mono subset-insertI)
  moreover
  have  $\text{sum} (\text{Euler-characteristic} (\text{insert} (a, b) \ A)) \ C = \text{sum} (\text{Euler-characteristic}$ 
 $A) \ C$ 
  proof (rule sum.cong [OF refl])
    fix  $C$ 
    assume  $C \in \mathcal{C}$ 
    have  $\text{Euler-characteristic} (\text{insert} (a, b) \ A) \ C = (-1) \wedge \text{nat}(\text{aff-dim } C)$ 
    proof (cases hyperplane-cell (insert  $(a, b) \ A$ )  $C$ )
      case True
      then show ?thesis
        using Euler-characteristic-cell by blast
      next
      case False
      with  $C[OF \langle C \in \mathcal{C} \rangle]$  have  $a \neq 0$ 
        by (smt (verit, ccfv-threshold) hyperplane-cell-Un hyperplane-cell-empty
hyperplane-cell-singleton insert-is-Un sup-bot-left)
        have convex  $C$ 
          using  $\langle \text{hyperplane-cell } A \ C \rangle$  hyperplane-cell-convex by blast
        define  $r$  where  $r \equiv (\sum D \in \{C' \cap C \mid C'. \text{hyperplane-cell } \{(a, b)\} \ C' \wedge C' \cap$ 
 $C \neq \{\}\}. (-1::\text{int}) \wedge \text{nat} (\text{aff-dim } D))$ 
        have  $\text{Euler-characteristic} (\text{insert} (a, b) \ A) \ C$ 
           $= (\sum D \mid (D \neq \{\}) \wedge$ 
             $(\exists C1 \ C2. \text{hyperplane-cell } \{(a, b)\} \ C1 \wedge \text{hyperplane-cell } A \ C2 \wedge$ 
 $D = C1 \cap C2)) \wedge D \subseteq C.$ 
             $(-1) \wedge \text{nat} (\text{aff-dim } D))$ 
        unfolding r-def Euler-characteristic-def insert-is-Un [of - A] hyperplane-cell-Un
        ..
        also have  $\dots = r$ 
          unfolding r-def
          apply (rule sum.cong [OF - refl])
          using  $\langle \text{hyperplane-cell } A \ C \rangle$  disjoint-hyperplane-cells disjnt-iff
          by (smt (verit, ccfv-SIG) Collect-cong Int-iff disjoint-iff subsetD subsetI)
        also have  $\dots = (-1) \wedge \text{nat}(\text{aff-dim } C)$ 
        proof –
          have  $C \neq \{\}$ 
            using  $\langle \text{hyperplane-cell } A \ C \rangle$  by auto
          show ?thesis
          proof (cases  $C \subseteq \{x. a \cdot x < b\} \vee C \subseteq \{x. a \cdot x > b\} \vee C \subseteq \{x. a \cdot x =$ 
 $b\}$ )

```

```

case Csub: True
with  $\langle C \neq \{\} \rangle$  have  $r = \text{sum } (\lambda c. (-1) \wedge \text{nat } (\text{aff-dim } c)) \{C\}$ 
  unfolding r-def
  apply (intro sum.cong [OF - refl])
  by (auto simp: \langle a \neq 0 \rangle hyperplane-cell-singleton)
also have  $\dots = (-1) \wedge \text{nat}(\text{aff-dim } C)$ 
  by simp
finally show ?thesis .
next
case False
then obtain  $u v$  where  $uv: u \in C \neg a \cdot u < b \ v \in C \neg a \cdot v > b$ 
  by blast
have CInt-ne:  $C \cap \{x. a \cdot x = b\} \neq \{\}$ 
proof (cases a \cdot u = b \vee a \cdot v = b)
  case True
    with  $uv$  show ?thesis
    by blast
  next
    case False
      have  $a \cdot v < a \cdot u$ 
      using False uv by auto
      define  $w$  where  $w \equiv v + ((b - a \cdot v) / (a \cdot u - a \cdot v)) *_R (u - v)$ 
      have **:  $v + a *_R (u - v) = (1 - a) *_R v + a *_R u$  for  $a$ 
      by (simp add: algebra-simps)
      have  $w \in C$ 
      unfolding w-def **
      proof (intro convexD-alt)
      qed (use \langle a \cdot v < a \cdot u \rangle \langle convex C \rangle uv in auto)
      moreover have  $w \in \{x. a \cdot x = b\}$ 
      using  $\langle a \cdot v < a \cdot u \rangle$  by (simp add: w-def inner-add-right inner-diff-right)
      ultimately show ?thesis
      by blast
    qed
  have Cab:  $C \cap \{x. a \cdot x < b\} \neq \{\} \wedge C \cap \{x. b < a \cdot x\} \neq \{\}$ 
proof -
  obtain  $u v$  where  $u \in C \ a \cdot u = b \ v \in C \ a \cdot v \neq b \ u \neq v$ 
  using False \langle C \cap \{x. a \cdot x = b\} \neq \{\} \rangle by blast
  have openin (subtopology euclidean (affine hull C))  $C$ 
  using  $\langle \text{hyperplane-cell } A \ C \rangle \langle \text{finite } A \rangle \text{hyperplane-cell-relatively-open}$ 
by blast
  then obtain  $\varepsilon$  where  $0 < \varepsilon$ 
    and  $\varepsilon: \bigwedge x'. \llbracket x' \in \text{affine hull } C; \text{dist } x' \ u < \varepsilon \rrbracket \implies x' \in C$ 
    by (meson \langle u \in C \rangle openin-euclidean-subtopology-iff)
  define  $\xi$  where  $\xi \equiv u - (\varepsilon / 2 / \text{norm } (v - u)) *_R (v - u)$ 
  have  $\xi \in C$ 
  proof (rule \varepsilon)
    show  $\xi \in \text{affine hull } C$ 
    by (simp add: \xi-def \langle u \in C \rangle \langle v \in C \rangle hull-inc mem-affine-3-minus2)
  qed (use \xi-def \langle 0 < \varepsilon \rangle in force)

```

```

consider  $a \cdot v < b \mid a \cdot v > b$ 
  using  $\langle a \cdot v \neq b \rangle$  by linarith
then show ?thesis
proof cases
  case 1
    moreover have  $\xi \in \{x. b < a \cdot x\}$ 
      using  $1 \langle 0 < \varepsilon \rangle \langle a \cdot u = b \rangle$  divide-less-cancel
      by (fastforce simp;  $\xi$ -def algebra-simps)
    ultimately show ?thesis
      using  $\langle v \in C \rangle \langle \xi \in C \rangle$  by blast
  next
    case 2
      moreover have  $\xi \in \{x. b > a \cdot x\}$ 
        using  $2 \langle 0 < \varepsilon \rangle \langle a \cdot u = b \rangle$  divide-less-cancel
        by (fastforce simp;  $\xi$ -def algebra-simps)
      ultimately show ?thesis
        using  $\langle v \in C \rangle \langle \xi \in C \rangle$  by blast
    qed
  qed
have  $r = (\sum C \in \{\{x. a \cdot x = b\} \cap C, \{x. b < a \cdot x\} \cap C, \{x. a \cdot x < b\}$ 
 $\cap C\}.$ 
     $(-1) \wedge \text{nat} (\text{aff-dim } C))$ 
  unfolding r-def
proof (intro sum.cong [OF - refl] equalityI)
  show  $\{\{x. a \cdot x = b\} \cap C, \{x. b < a \cdot x\} \cap C, \{x. a \cdot x < b\} \cap C\}$ 
     $\subseteq \{C' \cap C \mid C'. \text{hyperplane-cell } \{(a, b)\} C' \wedge C' \cap C \neq \{\}\}$ 
  apply clarsimp
    using Cab Int-commute  $\langle C \cap \{x. a \cdot x = b\} \neq \{\} \rangle$  hyper-
plane-cell-singleton  $\langle a \neq 0 \rangle$ 
    by metis
  qed (auto simp:  $\langle a \neq 0 \rangle$  hyperplane-cell-singleton)
  also have  $\dots = (-1) \wedge \text{nat} (\text{aff-dim } (C \cap \{x. a \cdot x = b\}))$ 
     $+ (-1) \wedge \text{nat} (\text{aff-dim } (C \cap \{x. b < a \cdot x\}))$ 
     $+ (-1) \wedge \text{nat} (\text{aff-dim } (C \cap \{x. a \cdot x < b\}))$ 
  using hyperplane-cells-distinct-lemma [of a b] Cab
  by (auto simp: sum.insert-if Int-commute Int-left-commute)
  also have  $\dots = (-1) \wedge \text{nat} (\text{aff-dim } C)$ 
proof  $-$ 
  have  $*$ :  $\text{aff-dim } (C \cap \{x. a \cdot x < b\}) = \text{aff-dim } C \wedge \text{aff-dim } (C \cap \{x. a$ 
 $\cdot x > b\}) = \text{aff-dim } C$ 
  by (metis Cab open-halfspace-lt open-halfspace-gt aff-dim-affine-hull
    affine-hull-convex-Int-open [OF  $\langle \text{convex } C \rangle$ ])
  obtain  $S T$  where open S affine T and Ceq: C = S  $\cap$  T
by (meson  $\langle \text{hyperplane-cell } A C \rangle \langle \text{finite } A \rangle$  hyperplane-cell-Int-open-affine)
  have affine hull C = affine hull T
  by (metis Ceq  $\langle C \neq \{\} \rangle \langle \text{affine } T \rangle \langle \text{open } S \rangle$  affine-hull-affine-Int-open
inf-commute)
  moreover
  have  $T \cap (\{x. a \cdot x = b\} \cap S) \neq \{\}$ 

```

using $Ceq \langle C \cap \{x. a \cdot x = b\} \neq \{\} \rangle$ **by** *blast*
then have $affine\ hull (C \cap \{x. a \cdot x = b\}) = affine\ hull (T \cap \{x. a \cdot x = b\})$
using $affine\ hull\ affine\ Int\ open[of\ T \cap \{x. a \cdot x = b\}\ S]$
by (*simp add: Ceq Int-ac* $\langle affine\ T \rangle$ *open S* $affine\ Int\ affine\ hyperplane$)
ultimately have $aff\ dim (affine\ hull\ C) = aff\ dim (affine\ hull (C \cap \{x. a \cdot x = b\})) + 1$
using $CInt\ ne\ False\ Ceq$
by (*auto simp: aff-dim-affine-Int-hyperplane* $\langle affine\ T \rangle$)
moreover have $0 \leq aff\ dim (C \cap \{x. a \cdot x = b\})$
by (*metis CInt-ne aff-dim-negative-iff linorder-not-le*)
ultimately show *?thesis*
by (*simp add: * nat-add-distrib*)
qed
finally show *?thesis* .
qed
qed
finally show $Euler\ characteristic (insert (a, b) A) C = (-1) \wedge nat(aff\ dim C)$.
qed
then show $Euler\ characteristic (insert (a, b) A) C = (Euler\ characteristic A C)$
by (*simp add: Euler-characteristic-cell C* $\langle C \in C \rangle$)
qed
ultimately show *?thesis*
by (*simp add: Euler-characteristic-cellcomplex-Union* $\langle S = \bigcup C \rangle$ *disjoint C* $\langle h = (a, b) \rangle$ *assms(1)*)
qed

lemma *Euler-characteristic-invariant-aux:*

assumes *finite B finite A hyperplane-cellcomplex A S*
shows $Euler\ characteristic (A \cup B) S = Euler\ characteristic A S$
using *assms*
by (*induction rule: finite-induct*) (*auto simp: Euler-characteristic-lemma hyperplane-cellcomplex-mono*)

lemma *Euler-characteristic-invariant:*

assumes *finite A finite B hyperplane-cellcomplex A S hyperplane-cellcomplex B S*
shows $Euler\ characteristic A S = Euler\ characteristic B S$
by (*metis Euler-characteristic-invariant-aux assms sup-commute*)

lemma *Euler-characteristic-inclusion-exclusion:*

assumes *finite A finite S* $\bigwedge K. K \in S \implies hyperplane\ cellcomplex\ A\ K$
shows $Euler\ characteristic A (\bigcup S) = (\sum \mathcal{T} \mid \mathcal{T} \subseteq S \wedge \mathcal{T} \neq \{\}. (-1) \wedge (card \mathcal{T} + 1) * Euler\ characteristic A (\bigcap \mathcal{T}))$

proof –

interpret *Incl-Excl hyperplane-cellcomplex A Euler-characteristic A*

```

proof
  show Euler-characteristic A (S ∪ T) = Euler-characteristic A S + Euler-characteristic
  A T
  if hyperplane-cellcomplex A S and hyperplane-cellcomplex A T and disjnt S T
for S T
  using that Euler-characteristic-cellcomplex-Un assms(1) by blast
  qed (use hyperplane-cellcomplex-Int hyperplane-cellcomplex-Un hyperplane-cellcomplex-diff
in auto)
  show ?thesis
  using restricted assms by blast
qed

```

1.5 Euler-type relation for full-dimensional proper polyhedral cones

```

lemma Euler-polyhedral-cone:
  fixes S :: 'n::euclidean-space set
  assumes polyhedron S conic S and intS: interior S ≠ {} and S ≠ UNIV
  shows (∑ d = 0..DIM('n). (- 1) ^ d * int (card {f. f face-of S ∧ aff-dim f =
  int d})) = 0 (is ?lhs = 0)
proof -
  have [simp]: affine hull S = UNIV
  by (simp add: affine-hull-nonempty-interior intS)
  with ⟨polyhedron S⟩
  obtain H where finite H
  and Seq: S = ⋂ H
  and Hex: ⋀ h. h ∈ H ⇒ ∃ a b. a ≠ 0 ∧ h = {x. a · x ≤ b}
  and Hsub: ⋀ G. G ⊂ H ⇒ S ⊂ ⋂ G
  by (fastforce simp: polyhedron-Int-affine-minimal)
  have 0 ∈ S
  using assms(2) conic-contains-0 intS interior-empty by blast
  have *: ∃ a. a ≠ 0 ∧ h = {x. a · x ≤ 0} if h ∈ H for h
proof -
  obtain a b where a ≠ 0 and ab: h = {x. a · x ≤ b}
  using Hex [OF ⟨h ∈ H⟩] by blast
  have 0 ∈ ⋂ H
  using Seq ⟨0 ∈ S⟩ by force
  then have 0 ∈ h
  using that by blast
  consider b=0 | b < 0 | b > 0
  by linarith
  then
  show ?thesis
proof cases
  case 1
  then show ?thesis
  using ⟨a ≠ 0⟩ ab by blast
next
  case 2

```

```

then show ?thesis
  using ⟨0 ∈ h⟩ ab by auto
next
case 3
have S ⊂ ∩(H - {h})
  using Hsub [of H - {h}] that by auto
then obtain x where x: x ∈ ∩(H - {h}) and x ∉ S
  by auto
define ε where ε ≡ min (1/2) (b / (a · x))
have b < a · x
  using ⟨x ∉ S⟩ ab x by (fastforce simp: ⟨S = ∩H⟩)
with 3 have 0 < a · x
  by auto
with 3 have 0 < ε
  by (simp add: ε-def)
have ε < 1
  using ε-def by linarith
have ε * (a · x) ≤ b
  unfolding ε-def using ⟨0 < a · x⟩ pos-le-divide-eq by fastforce
have x = inverse ε *R ε *R x
  using ⟨0 < ε⟩ by force
moreover
have ε *R x ∈ S
proof -
  have ε *R x ∈ h
    by (simp add: ⟨ε * (a · x) ≤ b⟩ ab)
  moreover have ε *R x ∈ ∩(H - {h})
proof -
  have ε *R x ∈ k if x ∈ k k ∈ H k ≠ h for k
proof -
  obtain a' b' where a' ≠ 0 k = {x. a' · x ≤ b'}
    using Hex ⟨k ∈ H⟩ by blast
  have (0 ≤ a' · x ⇒ a' · ε *R x ≤ a' · x)
    by (metis ⟨ε < 1⟩ inner-scaleR-right order-less-le pth-1 real-scaleR-def
scaleR-right-mono)
  moreover have (0 ≤ -(a' · x) ⇒ 0 ≤ -(a' · ε *R x))
    using ⟨0 < ε⟩ mult-le-0-iff order-less-imp-le by auto
  ultimately
  have a' · x ≤ b' ⇒ a' · ε *R x ≤ b'
  by (smt (verit) InterD ⟨0 ∈ ∩H⟩ ⟨k = {x. a' · x ≤ b'}⟩ inner-zero-right
mem-Collect-eq that(2))
  then show ?thesis
    using ⟨k = {x. a' · x ≤ b'}⟩ ⟨x ∈ k⟩ by fastforce
qed
with x show ?thesis
  by blast
qed
ultimately show ?thesis
  using Seq by blast

```

```

qed
with ⟨conic S⟩ have inverse  $\varepsilon *_{\mathbb{R}} \varepsilon *_{\mathbb{R}} x \in S$ 
by (meson ⟨ $0 < \varepsilon$ ⟩ conic-def inverse-nonnegative-iff-nonnegative order-less-le)
ultimately show ?thesis
using ⟨ $x \notin S$ ⟩ by presburger
qed
qed
then obtain fa where fa:  $\bigwedge h. h \in H \implies fa\ h \neq 0 \wedge h = \{x. fa\ h \cdot x \leq 0\}$ 
by metis
define fa-le-0 where fa-le-0  $\equiv \lambda h. \{x. fa\ h \cdot x \leq 0\}$ 
have fa':  $\bigwedge h. h \in H \implies fa-le-0\ h = h$ 
using fa fa-le-0-def by blast
define A where A  $\equiv (\lambda h. (fa\ h, 0::real)) \text{ ' } H$ 
have finite A
using ⟨finite H⟩ by (simp add: A-def)
then have ?lhs = Euler-characteristic A S
proof -
have [simp]:  $\text{card } \{f. f\ \text{face-of } S \wedge \text{aff-dim } f = \text{int } d\} = \text{card } \{C. \text{hyperplane-cell } A\ C \wedge C \subseteq S \wedge \text{aff-dim } C = \text{int } d\}$ 
if finite A and  $d \leq \text{card } (\text{Basis}::'n\ \text{set})$ 
for  $d :: \text{nat}$ 
proof (rule bij-betw-same-card)
have hyper1:  $\text{hyperplane-cell } A\ (\text{rel-interior } f) \wedge \text{rel-interior } f \subseteq S$ 
 $\wedge \text{aff-dim } (\text{rel-interior } f) = d \wedge \text{closure } (\text{rel-interior } f) = f$ 
if  $f\ \text{face-of } S\ \text{aff-dim } f = d$  for  $f$ 
proof -
have 1:  $\text{closure}(\text{rel-interior } f) = f$ 
proof -
have  $\text{closure}(\text{rel-interior } f) = \text{closure } f$ 
by (meson convex-closure-rel-interior face-of-imp-convex that(1))
also have  $\dots = f$ 
by (meson assms(1) closure-closed face-of-polyhedron-polyhedron polyhedron-imp-closed that(1))
finally show ?thesis .
qed
then have 2:  $\text{aff-dim } (\text{rel-interior } f) = d$ 
by (metis closure-aff-dim that(2))
have  $f \neq \{\}$ 
using aff-dim-negative-iff [of f] by (simp add: that(2))
obtain J0 where  $J0 \subseteq H$  and  $J0: f = \bigcap (fa-le-0 \text{ ' } H) \cap (\bigcap h \in J0. \{x. fa\ h \cdot x = 0\})$ 
proof (cases  $f = S$ )
case True
have  $S = \bigcap (fa-le-0 \text{ ' } H)$ 
using Seq fa by (auto simp: fa-le-0-def)
then show ?thesis
using True that by blast
next
case False

```

```

have fexp:  $f = \bigcap \{S \cap \{x. fa\ h \cdot x = 0\} \mid h. h \in H \wedge f \subseteq S \cap \{x. fa\ h \cdot x = 0\}\}$ 
proof (rule face-of-polyhedron-explicit)
  show  $S = \text{affine hull } S \cap \bigcap H$ 
  by (simp add: Seq hull-subset inf.absorb2)
  qed (auto simp: False <f ≠ { }> <f face-of S> <finite H> Hsub fa)
show ?thesis
proof
  have *:  $\bigwedge x\ h. \llbracket x \in f; h \in H \rrbracket \implies fa\ h \cdot x \leq 0$ 
  using Seq fa face-of-imp-subset <f face-of S> by fastforce
  show  $f = \bigcap (fa\ le\ 0 \ ' H) \cap (\bigcap h \in \{h \in H. f \subseteq S \cap \{x. fa\ h \cdot x = 0\}\}. \{x. fa\ h \cdot x = 0\})$ 
  (is f = ?I)
  proof
    show  $f \subseteq ?I$ 
    using <f face-of S> fa face-of-imp-subset by (force simp: * fa-le-0-def)
    show  $?I \subseteq f$ 
    apply (subst (2) fexp)
    apply (clarsimp simp: * fa-le-0-def)
    by (metis Inter-iff Seq fa mem-Collect-eq)
  qed
qed blast
qed
define  $H'$  where  $H' = (\lambda h. \{x. -(fa\ h) \cdot x \leq 0\}) \ ' H$ 
have  $\exists J. \text{finite } J \wedge J \subseteq H \cup H' \wedge f = \text{affine hull } f \cap \bigcap J$ 
proof (intro exI conjI)
  let  $?J = H \cup \text{image } (\lambda h. \{x. -(fa\ h) \cdot x \leq 0\}) \ J0$ 
  show finite (?J::'n set set)
  using <J0 ⊆ H> <finite H> finite-subset by fastforce
  show  $?J \subseteq H \cup H'$ 
  using <J0 ⊆ H> by (auto simp: H'-def)
  have  $f = \bigcap ?J$ 
  proof
    show  $f \subseteq \bigcap ?J$ 
    unfolding  $J0$  by (auto simp: fa^)
    have  $\bigwedge x\ j. \llbracket j \in J0; \forall h \in H. x \in h; \forall j \in J0. 0 \leq fa\ j \cdot x \rrbracket \implies fa\ j \cdot x = 0$ 
    by (metis <J0 ⊆ H> fa in-mono inf.absorb2 inf.orderE mem-Collect-eq)
    then show  $\bigcap ?J \subseteq f$ 
    unfolding  $J0$  by (auto simp: fa^)
  qed
  then show  $f = \text{affine hull } f \cap \bigcap ?J$ 
  by (simp add: Int-absorb1 hull-subset)
qed
then have *:  $\exists n\ J. \text{finite } J \wedge \text{card } J = n \wedge J \subseteq H \cup H' \wedge f = \text{affine hull } f \cap \bigcap J$ 
by blast
obtain  $J\ nJ$  where  $J: \text{finite } J \text{ card } J = nJ \ J \subseteq H \cup H'$  and fseq: f = affine hull f ∩ ∩ J
and minJ: ∏m J'. ∥finite J'; m < nJ; card J' = m; J' ⊆ H ∪ H'∥ ⇒ f

```

```

≠ affine hull f ∩ ⋂ J'
  using exists-least-iff [THEN iffD1, OF **] by metis
  have FF: f ⊆ (affine hull f ∩ ⋂ J') if J' ⊆ J for J'
  proof -
    have f ≠ affine hull f ∩ ⋂ J'
      using minJ
      by (metis J finite-subset psubset-card-mono psubset-imp-subset psub-
set-subset-trans that)
    then show ?thesis
      by (metis Int-subset-iff Inter-Un-distrib feq hull-subset inf-sup-ord(2)
psubsetI sup.absorb4 that)
  qed
  have ∃ a. {x. a · x ≤ 0} = h ∧ (h ∈ H ∧ a = fa h ∨ (∃ h'. h' ∈ H ∧ a =
-(fa h')))
    if h ∈ J for h
  proof -
    have h ∈ H ∪ H'
      using ⟨J ⊆ H ∪ H'⟩ that by blast
    then show ?thesis
      proof
        show ?thesis if h ∈ H
          using that fa by blast
        next
          assume h ∈ H'
          then obtain h' where h' ∈ H h = {x. 0 ≤ fa h' · x}
            by (auto simp: H'-def)
          then show ?thesis
            by (force simp: intro!: exI[where x=- (fa h')])
        qed
      qed
    then obtain ga
      where ga-h: ⋀ h. h ∈ J ⇒ h = {x. ga h · x ≤ 0}
      and ga-fa: ⋀ h. h ∈ J ⇒ h ∈ H ∧ ga h = fa h ∨ (∃ h'. h' ∈ H ∧ ga h
= -(fa h'))
      by metis
    have ∃: hyperplane-cell A (rel-interior f)
      proof -
        have D: rel-interior f = {x ∈ f. ∀ h ∈ J. ga h · x < 0}
          proof (rule rel-interior-polyhedron-explicit [OF ⟨finite J⟩ feq])
            show ga h ≠ 0 ∧ h = {x. ga h · x ≤ 0} if h ∈ J for h
              using that fa ga-fa ga-h by force
          qed (auto simp: FF)
        have H: h ∈ H ∧ ga h = fa h if h ∈ J for h
          proof -
            obtain z where z: z ∈ rel-interior f
              using 1 ⟨f ≠ {}⟩ by force
            then have z ∈ f ∧ z ∈ S
              using D ⟨f face-of S⟩ face-of-imp-subset by blast
            then show ?thesis
  
```

using *ga-fa* [*OF that*]
 by (*smt (verit, del-insts) D InterE Seq fa inner-minus-left mem-Collect-eq*
that z)
 qed
 then obtain *K* where $K \subseteq H$
 and $K: f = \bigcap (fa-le-0 \text{ ' } H) \cap (\bigcap h \in K. \{x. fa\ h \cdot x = 0\})$
 using *J0* $\langle J0 \subseteq H \rangle$ by *blast*
 have $E: rel\text{-interior } f = \{x. (\forall h \in H. fa\ h \cdot x \leq 0) \wedge (\forall h \in K. fa\ h \cdot x = 0) \wedge (\forall h \in J. ga\ h \cdot x < 0)\}$
 unfolding *D* by (*simp add: K fa-le-0-def*)
 have *relif*: $rel\text{-interior } f \neq \{\}$
 using *1* $\langle f \neq \{\} \rangle$ by *force*
 with *E* have *disjnt J K*
 using *H disjnt-iff* by *fastforce*
 define *IFJK* where $IFJK \equiv \lambda h. \text{if } h \in J \text{ then } \{x. fa\ h \cdot x < 0\}$
 else if $h \in K$ then $\{x. fa\ h \cdot x = 0\}$
 else if $rel\text{-interior } f \subseteq \{x. fa\ h \cdot x = 0\}$
 then $\{x. fa\ h \cdot x = 0\}$
 else $\{x. fa\ h \cdot x < 0\}$
 have *relint-f*: $rel\text{-interior } f = \bigcap (IFJK \text{ ' } H)$
 proof
 have *A*: *False*
 if $x: x \in rel\text{-interior } f$ and $y: y \in rel\text{-interior } f$ and *less0*: $fa\ h \cdot y < 0$
 and *fa0*: $fa\ h \cdot x = 0$ and $h \in H$ $h \notin J$ $h \notin K$ for $x\ h\ y$
 proof –
 obtain ε where $x \in f$ $\varepsilon > 0$
 and $\varepsilon: \bigwedge t. \llbracket dist\ x\ t \leq \varepsilon; t \in affine\ hull\ f \rrbracket \implies t \in f$
 using *x* by (*force simp: mem-rel-interior-cball*)
 then have $y \neq x$
 using *fa0 less0* by *force*
 define x' where $x' \equiv x + (\varepsilon / norm(y - x)) *_R (x - y)$
 have $x \in affine\ hull\ f \wedge y \in affine\ hull\ f$
 by (*metis* $\langle x \in f \rangle hull\text{-inc mem-rel-interior-cball } y$)
 moreover have $dist\ x\ x' \leq \varepsilon$
 using $\langle 0 < \varepsilon \rangle \langle y \neq x \rangle$ by (*simp add: x'-def divide-simps dist-norm*
norm-minus-commute)
 ultimately have $x' \in f$
 by (*simp add: \varepsilon mem-affine-3-minus x'-def*)
 have $x' \in S$
 using $\langle f\ \text{face-of } S \rangle \langle x' \in f \rangle$ *face-of-imp-subset* by *auto*
 then have $x' \in h$
 using *Seq that(5)* by *blast*
 then have $x' \in \{x. fa\ h \cdot x \leq 0\}$
 using *fa that(5)* by *blast*
 moreover have $\varepsilon / norm(y - x) * -(fa\ h \cdot y) > 0$
 using $\langle 0 < \varepsilon \rangle \langle y \neq x \rangle$ *less0* by (*simp add: field-split-simps*)
 ultimately show *?thesis*
 by (*simp add: x'-def fa0 inner-diff-right inner-right-distrib*)
 qed

show $rel\text{-interior } f \subseteq \bigcap (IFJK \text{ ' } H)$
unfolding $IFJK\text{-def}$ **by** (*smt (verit, ccfv-SIG) A E H INT-I in-mono mem-Collect-eq subsetI*)
show $\bigcap (IFJK \text{ ' } H) \subseteq rel\text{-interior } f$
using $\langle K \subseteq H \rangle \langle disjnt J K \rangle$
apply (*clarsimp simp add: ball-Un E H disjnt-iff IFJK-def*)
apply (*smt (verit, del-insts) IntI Int-Collect subsetD*)
done
qed
obtain z **where** $zrelf: z \in rel\text{-interior } f$
using *relif* **by** *blast*
moreover
have $H: z \in IFJK h \implies (x \in IFJK h) = (hyperplane\text{-side } (fa h, 0) z = hyperplane\text{-side } (fa h, 0) x)$ **for** $h x$
using $zrelf$ **by** (*auto simp: IFJK-def hyperplane-side-def sgn-if split: if-split-asm*)
then have $z \in \bigcap (IFJK \text{ ' } H) \implies (x \in \bigcap (IFJK \text{ ' } H)) = hyperplane\text{-equiv } A z x$ **for** x
unfolding $A\text{-def Inter-iff hyperplane-equiv-def ball-simps}$ **using** H **by** *blast*
then have $x \in rel\text{-interior } f \iff hyperplane\text{-equiv } A z x$ **for** x
using *relint-f zrelf* **by** *presburger*
ultimately show *?thesis*
by (*metis equalityI hyperplane-cell mem-Collect-eq subset-iff*)
qed
have $4: rel\text{-interior } f \subseteq S$
by (*meson face-of-imp-subset order-trans rel-interior-subset that(1)*)
show *?thesis*
using $1\ 2\ 3\ 4$ **by** *blast*
qed
have $hyper2: (closure\ c\ face\text{-of } S \wedge aff\text{-dim } (closure\ c) = d) \wedge rel\text{-interior } (closure\ c) = c$
if $c: hyperplane\text{-cell } A\ c$ **and** $c \subseteq S$ $aff\text{-dim } c = d$ **for** c
proof (*intro conjI*)
obtain J **where** $J \subseteq H$ **and** $J: c = (\bigcap h \in J. \{x. (fa\ h) \cdot x < 0\}) \cap (\bigcap h \in (H - J). \{x. (fa\ h) \cdot x = 0\})$
proof –
obtain z **where** $z: c = \{y. \forall x \in H. sgn (fa\ x \cdot y) = sgn (fa\ x \cdot z)\}$
using c **by** (*force simp: hyperplane-cell A-def hyperplane-equiv-def hyperplane-side-def*)
show *thesis*
proof
let $?J = \{h \in H. sgn (fa\ h \cdot z) = -1\}$
have $1: fa\ h \cdot x < 0$
if $\forall h \in H. sgn (fa\ h \cdot x) = sgn (fa\ h \cdot z)$ **and** $h \in H$ **and** $sgn (fa\ h \cdot z) = -1$ **for** $x h$
using *that* **by** (*metis sgn-1-neg*)
have $2: sgn (fa\ h \cdot z) = -1$
if $\forall h \in H. sgn (fa\ h \cdot x) = sgn (fa\ h \cdot z)$ **and** $h \in H$ **and** $fa\ h \cdot x \neq 0$

```

for  $x h$ 
  proof –
    have  $\llbracket 0 < fa\ h \cdot x; 0 < fa\ h \cdot z \rrbracket \implies False$ 
      using that fa by (smt (verit, del-insts) Inter-iff Seq  $\langle c \subseteq S \rangle$  mem-Collect-eq subset-iff z)
    then show ?thesis
      by (metis that sgn-if sgn-zero-iff)
    qed
    have  $\exists: sgn\ (fa\ h \cdot x) = sgn\ (fa\ h \cdot z)$ 
      if  $h \in H$  and  $\forall h. h \in H \wedge sgn\ (fa\ h \cdot z) = -1 \implies fa\ h \cdot x < 0$ 
      and  $\forall h \in H - \{h \in H. sgn\ (fa\ h \cdot z) = -1\}. fa\ h \cdot x = 0$ 
    for  $x h$ 
      using that 2 by (metis (mono-tags, lifting) Diff-iff mem-Collect-eq sgn-neg)
    show  $c = (\bigcap h \in ?J. \{x. fa\ h \cdot x < 0\}) \cap (\bigcap h \in H - ?J. \{x. fa\ h \cdot x = 0\})$ 
      unfolding  $z$  by (auto intro: 1 2 3)
    qed auto
  qed
  have finite J
    using  $\langle J \subseteq H \rangle \langle finite\ H \rangle$  finite-subset by blast
  show closure c face-of S
  proof –
    have  $cc: closure\ c = closure\ (\bigcap h \in J. \{x. fa\ h \cdot x < 0\}) \cap closure\ (\bigcap h \in H - J. \{x. fa\ h \cdot x = 0\})$ 
      unfolding  $J$ 
    proof (rule closure-Int-convex)
      show convex  $(\bigcap h \in J. \{x. fa\ h \cdot x < 0\})$ 
        by (simp add: convex-INT convex-halfspace-lt)
      show convex  $(\bigcap h \in H - J. \{x. fa\ h \cdot x = 0\})$ 
        by (simp add: convex-INT convex-hyperplane)
      have  $o1: open\ (\bigcap h \in J. \{x. fa\ h \cdot x < 0\})$ 
        by (metis open-INT[OF  $\langle finite\ J \rangle$ ] open-halfspace-lt)
      have  $o2: openin\ (top-of-set\ (affine\ hull\ (\bigcap h \in H - J. \{x. fa\ h \cdot x = 0\})))\ (\bigcap h \in H - J. \{x. fa\ h \cdot x = 0\})$ 
        proof –
          have affine  $(\bigcap h \in H - J. \{n. fa\ h \cdot n = 0\})$ 
            using affine-hyperplane by auto
          then show ?thesis
            by (metis (no-types) affine-hull-eq openin-subtopology-self)
        qed
      show rel-interior  $(\bigcap h \in J. \{x. fa\ h \cdot x < 0\}) \cap rel-interior\ (\bigcap h \in H - J. \{x. fa\ h \cdot x = 0\}) \neq \{\}$ 
        by (metis nonempty-hyperplane-cell c rel-interior-open o1 rel-interior-openin o2 J)
      qed
    have  $clo-im-J: closure\ '(\lambda h. \{x. fa\ h \cdot x < 0\})\ 'J = (\lambda h. \{x. fa\ h \cdot x \leq 0\})\ 'J$ 
      using  $\langle J \subseteq H \rangle$  by (force simp: image-comp fa)

```

```

have cleq: closure ( $\bigcap_{h \in H} - J. \{x. \text{fa } h \cdot x = 0\}$ ) = ( $\bigcap_{h \in H} - J. \{x. \text{fa } h \cdot x = 0\}$ )
by (intro closure-closed) (blast intro: closed-hyperplane)
have **: ( $\bigcap_{h \in J. \{x. \text{fa } h \cdot x \leq 0\}$ )  $\cap$  ( $\bigcap_{h \in H} - J. \{x. \text{fa } h \cdot x = 0\}$ )
face-of S
if ( $\bigcap_{h \in J. \{x. \text{fa } h \cdot x < 0\}$ )  $\neq \{\}$ 
proof (cases J=H)
case True
have [simp]: ( $\bigcap_{x \in H. \{x. \text{fa } x \cdot x \leq 0\}$ ) =  $\bigcap H$ 
using fa by auto
show ?thesis
using  $\langle \text{polyhedron } S \rangle$  by (simp add: Seq True polyhedron-imp-convex)
face-of-refl)
next
case False
have **: ( $\bigcap_{h \in J. \{n. \text{fa } h \cdot n \leq 0\}$ )  $\cap$  ( $\bigcap_{h \in H} - J. \{x. \text{fa } h \cdot x = 0\}$ )
=
( $\bigcap_{h \in H} - J. S \cap \{x. \text{fa } h \cdot x = 0\}$ ) (is ?L = ?R)
proof
show ?L  $\subseteq$  ?R
by clarsimp (smt (verit) DiffI InterI Seq fa mem-Collect-eq)
show ?R  $\subseteq$  ?L
using False Seq  $\langle J \subseteq H \rangle$  fa by blast
qed
show ?thesis
unfolding **
proof (rule face-of-Inter)
show ( $\lambda h. S \cap \{x. \text{fa } h \cdot x = 0\}$ ) ' $(H - J) \neq \{\}$ '
using False  $\langle J \subseteq H \rangle$  by blast
show T face-of S
if T:  $T \in (\lambda h. S \cap \{x. \text{fa } h \cdot x = 0\})$  ' $(H - J)$  for T
proof -
obtain h where h:  $T = S \cap \{x. \text{fa } h \cdot x = 0\}$  and  $h \in H$   $h \notin J$ 
using T by auto
have  $S \cap \{x. \text{fa } h \cdot x = 0\}$  face-of S
proof (rule face-of-Int-supporting-hyperplane-le)
show convex S
by (simp add: assms(1) polyhedron-imp-convex)
show  $\text{fa } h \cdot x \leq 0$  if  $x \in S$  for x
using that Seq fa  $\langle h \in H \rangle$  by auto
qed
then show ?thesis
using h by blast
qed
qed
qed
have *:  $\bigwedge S. S \in (\lambda h. \{x. \text{fa } h \cdot x < 0\})$  ' $J \implies \text{convex } S \wedge \text{open } S$ '
using convex-halfspace-lt open-halfspace-lt by fastforce
show ?thesis

```

```

    unfolding cc
    apply (simp add: * closure-Inter-convex-open)
    by (metis ** cleq clo-im-J image-image)
  qed
  show aff-dim (closure c) = int d
    by (simp add: that)
  show rel-interior (closure c) = c
    by (metis ⟨finite A⟩ c convex-rel-interior-closure hyperplane-cell-convex
hyperplane-cell-relative-interior)
  qed
  have rel-interior ‘ {f. f face-of S ∧ aff-dim f = int d}
    = {C. hyperplane-cell A C ∧ C ⊆ S ∧ aff-dim C = int d}
    using hyper1 hyper2 by fastforce
  then show bij-betw (rel-interior) {f. f face-of S ∧ aff-dim f = int d} {C.
hyperplane-cell A C ∧ C ⊆ S ∧ aff-dim C = int d}
    unfolding bij-betw-def inj-on-def by (metis (mono-tags) hyper1 mem-Collect-eq)

  qed
  show ?thesis
    by (simp add: Euler-characteristic ⟨finite A⟩)
  qed
  also have ... = 0
  proof -
    have A: hyperplane-cellcomplex A (- h) if h ∈ H for h
    proof (rule hyperplane-cellcomplex-mono [OF hyperplane-cell-cellcomplex])
      have - h = {x. fa h · x = 0} ∨ - h = {x. fa h · x < 0} ∨ - h = {x. 0 <
fa h · x}
      by (smt (verit, ccfv-SIG) Collect-cong Collect-neg-eq fa that)
    then show hyperplane-cell {(fa h, 0)} (- h)
      by (simp add: hyperplane-cell-singleton fa that)
    show {(fa h, 0)} ⊆ A
      by (simp add: A-def that)
    qed
  then have ∧h. h ∈ H ⇒ hyperplane-cellcomplex A h
    using hyperplane-cellcomplex-Compl by fastforce
  then have hyperplane-cellcomplex A S
    by (simp add: Seq hyperplane-cellcomplex-Inter)
  then have D: Euler-characteristic A (UNIV::'n set) =
    Euler-characteristic A (∩ H) + Euler-characteristic A (- ∩ H)
    using Euler-characteristic-cellcomplex-Un
    by (metis Compl-partition Diff-cancel Diff-eq Seq ⟨finite A⟩ disjnt-def hyper-
plane-cellcomplex-Compl)
  have Euler-characteristic A UNIV = Euler-characteristic {} (UNIV::'n set)
    by (simp add: Euler-characteristic-invariant ⟨finite A⟩)
  then have E: Euler-characteristic A UNIV = (-1) ^ (DIM('n))
    by (simp add: Euler-characteristic-cell)
  have DD: Euler-characteristic A (∩ (uminus ‘ J)) = (- 1) ^ DIM('n)
    if J ≠ {} J ⊆ H for J
  proof -

```

```

define  $B$  where  $B \equiv (\lambda h. (fa\ h, 0::real)) \text{ ' } J$ 
then have  $B \subseteq A$ 
  by (simp add: A-def image-mono that)
have  $\exists x. y = -x$  if  $y \in \bigcap (uminus \text{ ' } H)$  for  $y::'n$  — Weirdly, the assumption
is not used
  by (metis add.inverse-inverse)
moreover have  $-x \in \bigcap (uminus \text{ ' } H) \iff x \in interior\ S$  for  $x$ 
proof —
  have  $1: interior\ S = \{x \in S. \forall h \in H. fa\ h \cdot x < 0\}$ 
    using rel-interior-polyhedron-explicit [OF <finite H> - fa]
    by (metis (no-types, lifting) inf-top-left Hsub Seq <affine hull S = UNIV>)
rel-interior-interior
  have  $2: \bigwedge x y. \llbracket y \in H; \forall h \in H. fa\ h \cdot x < 0; -x \in y \rrbracket \implies False$ 
    by (smt (verit, best) fa inner-minus-right mem-Collect-eq)
  show ?thesis
    apply (simp add: 1)
    by (smt (verit) 2 * fa Inter-iff Seq inner-minus-right mem-Collect-eq)
qed
ultimately have INT-Compl-H:  $\bigcap (uminus \text{ ' } H) = uminus \text{ ' } interior\ S$ 
  by blast
obtain  $z$  where  $z: z \in \bigcap (uminus \text{ ' } J)$ 
  using  $\langle J \subseteq H \rangle \langle \bigcap (uminus \text{ ' } H) = uminus \text{ ' } interior\ S \rangle intS$  by fastforce
have  $\bigcap (uminus \text{ ' } J) = Collect\ (hyperplane-equiv\ B\ z)$  (is  $?L = ?R$ )
proof
  show  $?L \subseteq ?R$ 
    using  $\langle J \subseteq H \rangle z$ 
    by (fastforce simp: hyperplane-equiv-def hyperplane-side-def B-def set-eq-iff)
)
  show  $?R \subseteq ?L$ 
    using  $z \langle J \subseteq H \rangle$  apply (clarsimp simp add: hyperplane-equiv-def hyper-
plane-side-def B-def)
    by (metis fa in-mono mem-Collect-eq sgn-le-0-iff)
qed
then have hyper-B: hyperplane-cell  $B (\bigcap (uminus \text{ ' } J))$ 
  by (metis hyperplane-cell)
have Euler-characteristic A  $(\bigcap (uminus \text{ ' } J)) = Euler-characteristic\ B (\bigcap$ 
(uminus ' J)))
proof (rule Euler-characteristic-invariant [OF <finite A>])
  show finite B
    using  $\langle B \subseteq A \rangle \langle finite\ A \rangle finite-subset$  by blast
  show hyperplane-cellcomplex A  $(\bigcap (uminus \text{ ' } J))$ 
  by (meson <B ⊆ A> hyper-B hyperplane-cell-cellcomplex hyperplane-cellcomplex-mono)
  show hyperplane-cellcomplex B  $(\bigcap (uminus \text{ ' } J))$ 
    by (simp add: hyper-B hyperplane-cell-cellcomplex)
qed
also have  $\dots = (-1)^{\wedge nat\ (aff-dim\ (\bigcap (uminus \text{ ' } J)))}$ 
  using Euler-characteristic-cell hyper-B by blast
also have  $\dots = (-1)^{\wedge DIM('n)}$ 
proof —

```

have *affine hull* \cap (*uminus* ' *H*) = *UNIV*
by (*simp add: INT-Compl-H affine-hull-nonempty-interior intS interior-negations*)
then have *affine hull* \cap (*uminus* ' *J*) = *UNIV*
by (*metis Inf-superset-mono hull-mono subset-UNIV subset-antisym subset-image-iff that(2)*)
with *aff-dim-eq-full* **show** *?thesis*
by (*metis nat-int*)
qed
finally show *?thesis* .
qed
have *EE*: $(\sum \mathcal{T} \mid \mathcal{T} \subseteq \text{uminus } ' H \wedge \mathcal{T} \neq \{\}) \cdot (-1)^{\wedge} (\text{card } \mathcal{T} + 1) * \text{Euler-characteristic } A (\cap \mathcal{T})$
 $= (\sum \mathcal{T} \mid \mathcal{T} \subseteq \text{uminus } ' H \wedge \mathcal{T} \neq \{\}) \cdot (-1)^{\wedge} (\text{card } \mathcal{T} + 1) * (-1)^{\wedge} \text{DIM}('n)$
by (*intro sum.cong [OF refl]*) (*fastforce simp: subset-image-iff intro!: DD*)
also have ... = $(-1)^{\wedge} \text{DIM}('n)$
proof -
have *A*: $(\sum y = 1.. \text{card } H. \sum t \in \{x \in \{\mathcal{T}. \mathcal{T} \subseteq \text{uminus } ' H \wedge \mathcal{T} \neq \{\}\}. \text{card } x = y\}. (-1)^{\wedge} (\text{card } t + 1))$
 $= (\sum \mathcal{T} \in \{\mathcal{T}. \mathcal{T} \subseteq \text{uminus } ' H \wedge \mathcal{T} \neq \{\}\}. (-1)^{\wedge} (\text{card } \mathcal{T} + 1))$
proof (*rule sum.group*)
have $\bigwedge C. \llbracket C \subseteq \text{uminus } ' H; C \neq \{\} \rrbracket \implies \text{Suc } 0 \leq \text{card } C \wedge \text{card } C \leq \text{card } H$
by (*meson* $\langle \text{finite } H \rangle$ *card-eq-0-iff finite-surj le-zero-eq not-less-eq-eq surj-card-le*)
then show $\text{card } \{\mathcal{T}. \mathcal{T} \subseteq \text{uminus } ' H \wedge \mathcal{T} \neq \{\}\} \subseteq \{1.. \text{card } H\}$
by *force*
qed (*auto simp:* $\langle \text{finite } H \rangle$)

have $(\sum n = \text{Suc } 0.. \text{card } H. - (\text{int } (\text{card } \{x. x \subseteq \text{uminus } ' H \wedge x \neq \{\} \wedge \text{card } x = n\}) * (-1)^{\wedge} n))$
 $= (\sum n = \text{Suc } 0.. \text{card } H. (-1)^{\wedge} (\text{Suc } n) * (\text{card } H \text{ choose } n))$
proof (*rule sum.cong [OF refl]*)
fix *n*
assume $n \in \{\text{Suc } 0.. \text{card } H\}$
then have $\{\mathcal{T}. \mathcal{T} \subseteq \text{uminus } ' H \wedge \mathcal{T} \neq \{\} \wedge \text{card } \mathcal{T} = n\} = \{\mathcal{T}. \mathcal{T} \subseteq \text{uminus } ' H \wedge \text{card } \mathcal{T} = n\}$
by *auto*
then have $\text{card} \{\mathcal{T}. \mathcal{T} \subseteq \text{uminus } ' H \wedge \mathcal{T} \neq \{\} \wedge \text{card } \mathcal{T} = n\} = \text{card } (\text{uminus } ' H) \text{ choose } n$
by (*simp add:* $\langle \text{finite } H \rangle$ *n-subsets*)
also have ... = $\text{card } H \text{ choose } n$
by (*metis card-image double-complement inj-on-inverseI*)
finally
show - $(\text{int } (\text{card } \{\mathcal{T}. \mathcal{T} \subseteq \text{uminus } ' H \wedge \mathcal{T} \neq \{\} \wedge \text{card } \mathcal{T} = n\}) * (-1)^{\wedge} n) = (-1)^{\wedge} \text{Suc } n * \text{int } (\text{card } H \text{ choose } n)$
by *simp*
qed

```

also have ... = - (∑ k = Suc 0..card H. (-1) ^ k * (card H choose k))
  by (simp add: sum-negf)
also have ... = 1 - (∑ k=0..card H. (-1) ^ k * (card H choose k))
  using atLeastSucAtMost-greaterThanAtMost by (simp add: sum.head [of 0])
also have ... = 1 - 0 ^ card H
  using binomial-ring [of -1 1::int card H] by (simp add: mult.commute
atLeast0AtMost)
also have ... = 1
  using Seq ⟨finite H⟩ ⟨S ≠ UNIV⟩ card-0-eq by auto
finally have C: (∑ n = Suc 0..card H. - (int (card {x. x ⊆ uminus ' H ∧
x ≠ {} ∧ card x = n}) * (-1) ^ n)) = (1::int) .

  have (∑ T | T ⊆ uminus ' H ∧ T ≠ {}. (-1) ^ (card T + 1)) = (1::int)
  unfolding A [symmetric] by (simp add: C)
  then show ?thesis
  by (simp flip: sum-distrib-right power-Suc)
qed
  finally have (∑ T | T ⊆ uminus ' H ∧ T ≠ {}. (-1) ^ (card T + 1) *
Euler-characteristic A (∩ T))
    = (-1) ^ DIM('n) .
  then have Euler-characteristic A (∪ (uminus ' H)) = (-1) ^ (DIM('n))
  using Euler-characteristic-inclusion-exclusion [OF ⟨finite A⟩]
  by (smt (verit) A Collect-cong ⟨finite H⟩ finite-imageI image-iff sum.cong)
  then show ?thesis
  using D E by (simp add: uminus-Inf Seq)
qed
finally show ?thesis .
qed

```

1.6 Euler-Poincare relation for special $(n - 1)$ -dimensional polytope

lemma *Euler-Poincare-lemma*:

```

fixes p :: 'n::euclidean-space set
assumes DIM('n) ≥ 2 polytope p i ∈ Basis and affp: affine hull p = {x. x · i
= 1}
shows (∑ d = 0..DIM('n) - 1. (-1) ^ d * int (card {f. f face-of p ∧ aff-dim f
= int d})) = 1
proof -
  have aff-dim p = aff-dim {x. i · x = 1}
  by (metis (no-types, lifting) Collect-cong aff-dim-affine-hull affp inner-commute)
  also have ... = int (DIM('n) - 1)
  using aff-dim-hyperplane [of i 1] ⟨i ∈ Basis⟩ by fastforce
  finally have AP: aff-dim p = int (DIM('n) - 1) .
  show ?thesis
  proof (cases p = {})
  case True
  with AP show ?thesis by simp
next

```

```

case False
define S where S ≡ conic hull p
have 1: (conic hull f) ∩ {x. x · i = 1} = f if f ⊆ {x. x · i = 1} for f
  using that
  by (smt (verit, ccfv-threshold) affp conic-hull-Int-affine-hull hull-hull inner-zero-left mem-Collect-eq)
obtain K where finite K and K: p = convex hull K
  by (meson assms(2) polytope-def)
then have convex-cone hull K = conic hull (convex hull K)
  using False convex-cone-hull-separate-nonempty by auto
then have polyhedron S
  using polyhedron-convex-cone-hull
  by (simp add: S-def ⟨polytope p⟩ polyhedron-conic-hull-polytope)
then have convex S
  by (simp add: polyhedron-imp-convex)
then have conic S
  by (simp add: S-def conic-conic-hull)
then have 0 ∈ S
  by (simp add: False S-def)
have S ≠ UNIV
proof
  assume S = UNIV
  then have conic hull p ∩ {x. x · i = 1} = p
    by (metis 1 affp hull-subset)
  then have bounded {x. x · i = 1}
    using S-def ⟨S = UNIV⟩ assms(2) polytope-imp-bounded by auto
  then obtain B where B > 0 and B: ∧x. x ∈ {x. x · i = 1} ⇒ norm x ≤ B
    using bounded-normE by blast
  define x where x ≡ (∑ b ∈ Basis. (if b = i then 1 else B + 1) *R b)
  obtain j where j: j ∈ Basis j ≠ i
    using ⟨DIM('n) ≥ 2⟩
    by (metis DIM-complex DIM-ge-Suc0 card-2-iff' card-le-Suc0-iff-eq euclidean-space-class.finite-Basis le-antisym)
  have B + 1 ≤ |x · j|
    using j by (simp add: x-def)
  also have ... ≤ norm x
    using Basis-le-norm j by blast
  finally have norm x > B
    by simp
  moreover have x · i = 1
    by (simp add: x-def ⟨i ∈ Basis⟩)
  ultimately show False
    using B by force
qed
have S ≠ {}
  by (metis False S-def empty-subsetI equalityI hull-subset)
have ∧c x. [0 < c; x ∈ p; x ≠ 0] ⇒ 0 < (c *R x) · i
  by (metis (mono-tags) Int-Collect Int-iff affp hull-inc inner-commute inner-scaleR-right mult.right-neutral)

```

then have *doti-gt0*: $0 < x \cdot i$ **if** $S: x \in S$ **and** $x \neq 0$ **for** x
using *that by* (*auto simp: S-def conic-hull-explicit*)
have $\bigwedge a. \{a\}$ *face-of* $S \implies a = 0$
using \langle *conic S* \rangle *conic-contains-0 face-of-conic* **by** *blast*
moreover have $\{0\}$ *face-of* S
proof –
have $\bigwedge a b u. \llbracket a \in S; b \in S; a \neq b; u < 1; 0 < u; (1 - u) *_R a + u *_R b = 0 \rrbracket \implies False$
using *conic-def euclidean-all-zero-iff inner-left-distrib scaleR-eq-0-iff*
by (*smt (verit, del-insts) doti-gt0 <conic S> <i ∈ Basis>*)
then show *?thesis*
by (*auto simp: in-segment face-of-singleton extreme-point-of-def <0 ∈ S>*)
qed
ultimately have *face-0*: $\{f. f \text{ face-of } S \wedge (\exists a. f = \{a\})\} = \{\{0\}\}$
by *auto*
have *interior* $S \neq \{\}$
proof
assume *interior* $S = \{\}$
then obtain $a b$ **where** $a \neq 0$ **and** $ab: S \subseteq \{x. a \cdot x = b\}$
by (*metis <convex S> empty-interior-subset-hyperplane*)
have $\{x. x \cdot i = 1\} \subseteq \{x. a \cdot x = b\}$
by (*metis S-def ab affine-hyperplane affp hull-inc subset-eq subset-hull*)
moreover have $\neg \{x. x \cdot i = 1\} \subset \{x. a \cdot x = b\}$
using *aff-dim-hyperplane [of a b]*
by (*metis AP <a ≠ 0> aff-dim-eq-full-gen affine-hyperplane affp hull-subset less-le-not-le subset-hull*)
ultimately have $S \subseteq \{x. x \cdot i = 1\}$
using ab **by** *auto*
with $\langle S \neq \{\} \rangle$ **show** *False*
using \langle *conic S* \rangle *conic-contains-0* **by** *fastforce*
qed
then have $(\sum d = 0..DIM('n). (-1) ^ d * \text{int} (\text{card} \{f. f \text{ face-of } S \wedge \text{aff-dim } f = \text{int } d\})) = 0$
using *Euler-polyhedral-cone <S ≠ UNIV> <conic S> <polyhedron S>* **by** *blast*
then have $1 + (\sum d = 1..DIM('n). (-1) ^ d * (\text{card} \{f. f \text{ face-of } S \wedge \text{aff-dim } f = d\})) = 0$
by (*simp add: sum.atLeast-Suc-atMost aff-dim-eq-0 face-0*)
moreover have $(\sum d = 1..DIM('n). (-1) ^ d * (\text{card} \{f. f \text{ face-of } S \wedge \text{aff-dim } f = d\}))$
 $= - (\sum d = 0..DIM('n) - 1. (-1) ^ d * \text{int} (\text{card} \{f. f \text{ face-of } p \wedge \text{aff-dim } f = \text{int } d\}))$
proof –
have $(\sum d = 1..DIM('n). (-1) ^ d * (\text{card} \{f. f \text{ face-of } S \wedge \text{aff-dim } f = d\}))$
 $= (\sum d = \text{Suc } 0.. \text{Suc } (DIM('n) - 1). (-1) ^ d * (\text{card} \{f. f \text{ face-of } S \wedge \text{aff-dim } f = d\}))$
by *auto*
also have $\dots = - (\sum d = 0..DIM('n) - 1. (-1) ^ d * \text{card} \{f. f \text{ face-of } S \wedge \text{aff-dim } f = 1 + \text{int } d\})$
unfolding *sum.atLeast-Suc-atMost-Suc-shift* **by** (*simp add: sum-negf*)

also have ... = - ($\sum d = 0..DIM('n) - 1. (-1)^d * card \{f. f \text{ face-of } p \wedge \text{aff-dim } f = \text{int } d\}$)
proof -
 { **fix** d
 assume $d \leq DIM('n) - Suc\ 0$
 have *conic-face-p*: (*conic hull f*) *face-of S* **if** $f \text{ face-of } p$ **for** f
 proof (*cases f={}*)
 case *False*
 have $\{c *_R x \mid c \cdot x. 0 \leq c \wedge x \in f\} \subseteq \{c *_R x \mid c \cdot x. 0 \leq c \wedge x \in p\}$
 using *face-of-imp-subset* **that** **by** *blast*
 moreover
 have *convex* $\{c *_R x \mid c \cdot x. 0 \leq c \wedge x \in f\}$
 by (*metis* (*no-types*) *cone-hull-expl convex-cone-hull face-of-imp-convex*
that)
 moreover
 have $(\exists c \cdot x. ca *_R a = c *_R x \wedge 0 \leq c \wedge x \in f) \wedge (\exists c \cdot x. cb *_R b = c *_R x \wedge 0 \leq c \wedge x \in f)$
 if $\forall a \in p. \forall b \in p. (\exists x \in f. x \in \text{open-segment } a\ b) \longrightarrow a \in f \wedge b \in f$
 and $0 \leq ca \wedge a \in p \wedge 0 \leq cb \wedge b \in p$
 and $0 \leq cx \wedge x \in f$ **and** *oseg*: $cx *_R x \in \text{open-segment } (ca *_R a) (cb *_R b)$
 for $ca\ a\ cb\ b\ cx\ x$
 proof -
 have $ai: a \cdot i = 1$ **and** $bi: b \cdot i = 1$
 using *affp hull-inc* *that(3,5)* **by** *fastforce+*
 have $xi: x \cdot i = 1$
 using *affp* *that* $\langle f \text{ face-of } p \rangle$ *face-of-imp-subset hull-subset* **by** *fastforce*
 show *?thesis*
 proof (*cases cx *_R x = 0*)
 case *True*
 then show *?thesis*
 using $\langle \{0\} \text{ face-of } S \rangle$ *face-ofD* $\langle \text{conic } S \rangle$ *that*
 by (*smt* (*verit*, *best*) *S-def conic-def hull-subset insertCI singletonD subsetD*)
 next
 case *False*
 then have $cx \neq 0 \wedge x \neq 0$
 by *auto*
 obtain u **where** $0 < u \wedge u < 1$ **and** $u: cx *_R x = (1 - u) *_R (ca *_R a) + u *_R (cb *_R b)$
 using *oseg in-segment(2)* **by** *metis*
 show *?thesis*
 proof (*cases x = a*)
 case *True*
 then have $ua: (cx - (1 - u) *_R ca) *_R a = (u *_R cb) *_R b$
 using u **by** (*simp add: algebra-simps*)
 then have $(cx - (1 - u) *_R ca) * 1 = u *_R cb * 1$
 by (*metis ai bi inner-scaleR-left*)
 then have $a=b \vee cb = 0$

```

    using ua <0 < u> by force
  then show ?thesis
    by (metis True scaleR-zero-left that(2) that(4) that(7))
next
case False
show ?thesis
proof (cases x = b)
  case True
  then have ub: (cx - (u * cb)) *R b = ((1 - u) * ca) *R a
    using u by (simp add: algebra-simps)
  then have (cx - (u * cb)) * 1 = ((1 - u) * ca) * 1
    by (metis ai bi inner-scaleR-left)
  then have a=b ∨ ca = 0
    using <u < 1> ub by auto
  then show ?thesis
    using False True that(4) that(7) by auto
next
case False
have cx > 0
  using <cx ≠ 0> <0 ≤ cx> by linarith
have False if ca = 0
proof -
  have cx = u * cb
  by (metis add-0 bi inner-real-def inner-scaleR-left real-inner-1-right
scale-eq-0-iff that u xi)
  then show False
    using <x ≠ b> <cx ≠ 0> that u by force
qed
with <0 ≤ ca> have ca > 0
  by force
have aff: x ∈ affine hull p ∧ a ∈ affine hull p ∧ b ∈ affine hull p
  using affp xi ai bi by blast
show ?thesis
proof (cases cb=0)
  case True
  have u': cx *R x = ((1 - u) * ca) *R a
    using u by (simp add: True)
  then have cx = ((1 - u) * ca)
    by (metis ai inner-scaleR-left mult.right-neutral xi)
  then show ?thesis
    using True u' <cx ≠ 0> <ca ≥ 0> <x ∈ f> by auto
next
case False
with <cb ≥ 0> have cb > 0
  by linarith
{ have False if a=b
proof -
  have *: cx *R x = ((1 - u) * ca + u * cb) *R b
    using u that by (simp add: algebra-simps)

```

```

    then have  $cx = ((1 - u) * ca + u * cb)$ 
      by (metis xi bi inner-scaleR-left mult.right-neutral)
    with  $\langle x \neq b \rangle \langle cx \neq 0 \rangle$  * show False
      by force
    qed
  }
  moreover
    have  $cx *_{\mathbb{R}} x /_{\mathbb{R}} cx = (((1 - u) * ca) *_{\mathbb{R}} a + (cb * u) *_{\mathbb{R}} b)$ 
      / $\mathbb{R}$   $cx$ 
      using u by simp
    then have  $x_{eq}: x = ((1 - u) * ca / cx) *_{\mathbb{R}} a + (cb * u / cx) *_{\mathbb{R}} b$ 
      by (simp add:  $\langle cx \neq 0 \rangle$  divide-inverse-commute scaleR-right-distrib)
    then have  $proj: 1 = ((1 - u) * ca / cx) + (cb * u / cx)$ 
      using ai bi xi by (simp add: inner-left-distrib)
    then have  $eq: cx + ca * u = ca + cb * u$ 
      using  $\langle cx > 0 \rangle$  by (simp add: field-simps)
    have  $\exists u > 0. u < 1 \wedge x = (1 - u) *_{\mathbb{R}} a + u *_{\mathbb{R}} b$ 
      proof (intro exI conjI)
        show  $0 < \text{inverse } cx * u * cb$ 
          by (simp add:  $\langle 0 < cb \rangle \langle 0 < cx \rangle \langle 0 < u \rangle$ )
        show  $\text{inverse } cx * u * cb < 1$ 
          using  $proj \langle 0 < ca \rangle \langle 0 < cx \rangle \langle u < 1 \rangle$  by (simp add:
          divide-simps)
        show  $x = (1 - \text{inverse } cx * u * cb) *_{\mathbb{R}} a + (\text{inverse } cx * u *
          cb) *_{\mathbb{R}} b$ 
          using eq  $\langle cx \neq 0 \rangle$  by (simp add: xeq field-simps)
      qed
    ultimately show ?thesis
      using that by (metis in-segment(2))
    qed
  qed
  qed
  qed
  qed
  ultimately show ?thesis
    using that by (auto simp: S-def conic-hull-explicit face-of-def)
  qed auto
  moreover
    have  $\text{conic-hyperplane-eq}: \text{conic hull } (f \cap \{x. x \cdot i = 1\}) = f$ 
      if  $f$  face-of  $S$   $0 < \text{aff-dim } f$  for  $f$ 
    proof
      show  $\text{conic hull } (f \cap \{x. x \cdot i = 1\}) \subseteq f$ 
        by (metis  $\langle \text{conic } S \rangle$  face-of-conic inf-le1 subset-hull that(1))
      have  $\exists c x'. x = c *_{\mathbb{R}} x' \wedge 0 \leq c \wedge x' \in f \wedge x' \cdot i = 1$  if  $x \in f$  for  $x$ 
        proof (cases  $x=0$ )
          case True
            obtain  $y$  where  $y \in f$   $y \neq 0$ 
              by (metis  $\langle 0 < \text{aff-dim } f \rangle$  aff-dim-sing aff-dim-subset insertCI
              linorder-not-le subset-iff)

```

then have $y \cdot i > 0$
using $\langle f \text{ face-of } S \rangle \text{ doti-gt0 face-of-imp-subset}$ **by** *blast*
then have $y /_R (y \cdot i) \in f \wedge (y /_R (y \cdot i)) \cdot i = 1$
using $\langle \text{conic } S \rangle \langle f \text{ face-of } S \rangle \langle y \in f \rangle \text{ conic-def face-of-conic}$ **by** *fastforce*
then show *?thesis*
using *True* **by** *fastforce*
next
case *False*
then have $x \cdot i > 0$
using $\langle f \text{ face-of } S \rangle \text{ doti-gt0 face-of-imp-subset}$ **that** **by** *blast*
then have $x /_R (x \cdot i) \in f \wedge (x /_R (x \cdot i)) \cdot i = 1$
using $\langle \text{conic } S \rangle \langle f \text{ face-of } S \rangle \langle x \in f \rangle \text{ conic-def face-of-conic}$ **by** *fastforce*
then show *?thesis*
by $(\text{metis } \langle 0 < x \cdot i \rangle \text{ divideR-right eucl-less-le-not-le})$
qed
then show $f \subseteq \text{conic hull } (f \cap \{x. x \cdot i = 1\})$
by $(\text{auto simp: conic-hull-explicit})$
qed

have *conic-face-S: conic hull f face-of S*
if *f face-of S* **for** *f*
by $(\text{metis } \langle \text{conic } S \rangle \text{ face-of-conic hull-same that})$

have *aff-1d: aff-dim (conic hull f) = aff-dim f + 1 (is ?lhs = ?rhs)*
if *f face-of p* **and** $f \neq \{\}$ **for** *f*
proof $(\text{rule order-antisym})$
have $?lhs \leq \text{aff-dim}(\text{affine hull } (\text{insert } 0 (\text{affine hull } f)))$
proof $(\text{intro aff-dim-subset hull-minimal})$
show $f \subseteq \text{affine hull } \text{insert } 0 (\text{affine hull } f)$
by $(\text{metis hull-insert hull-subset insert-subset})$
show *conic (affine hull insert 0 (affine hull f))*
by $(\text{metis affine-hull-span-0 conic-span hull-inc insertI1})$
qed
also have $\dots \leq ?rhs$
by $(\text{simp add: aff-dim-insert})$
finally show $?lhs \leq ?rhs$.
have $\text{aff-dim } f < \text{aff-dim } (\text{conic hull } f)$
proof $(\text{intro aff-dim-psubset psubsetI})$
show $\text{affine hull } f \subseteq \text{affine hull } (\text{conic hull } f)$
by $(\text{simp add: hull-mono hull-subset})$
have $0 \notin \text{affine hull } f$
using *affp face-of-imp-subset hull-mono that(1)* **by** *fastforce*
moreover have $0 \in \text{affine hull } (\text{conic hull } f)$
by $(\text{simp add: } \langle f \neq \{\} \rangle \text{ hull-inc})$
ultimately show $\text{affine hull } f \neq \text{affine hull } (\text{conic hull } f)$
by *auto*
qed
then show $?rhs \leq ?lhs$
by *simp*

qed

have face-S-imp-face-p: $\bigwedge f. f \text{ face-of } S \implies f \cap \{x. x \cdot i = 1\} \text{ face-of } p$
by (metis 1 S-def affp convex-affine-hull face-of-slice hull-subset)

have conic-eq-f: $\text{conic hull } f \cap \{x. x \cdot i = 1\} = f$
if $f \text{ face-of } p$ for f
by (metis 1 affp face-of-imp-subset hull-subset le-inf-iff that)

have dim-f-hyperplane: $\text{aff-dim } (f \cap \{x. x \cdot i = 1\}) = \text{int } d$
if $f \text{ face-of } S$ $\text{aff-dim } f = 1 + \text{int } d$ for f
proof –
have conic f
using $\langle \text{conic } S \rangle \text{ face-of-conic that}(1)$ by blast
then have $0 \in f$
using conic-contains-0 that by force
moreover have $\neg f \subseteq \{0\}$
using subset-singletonD that(2) by fastforce
ultimately obtain y where $y: y \in f \ y \neq 0$
by blast
then have $y \cdot i > 0$
using doti-gt0 face-of-imp-subset that(1) by blast
have $\text{aff-dim } (\text{conic hull } (f \cap \{x. x \cdot i = 1\})) = \text{aff-dim } (f \cap \{x. x \cdot i$
 $= 1\}) + 1$
proof (rule aff-1d)
show $f \cap \{x. x \cdot i = 1\} \text{ face-of } p$
by (simp add: face-S-imp-face-p that(1))
have $\text{inverse}(y \cdot i) *_{\mathbb{R}} y \in f$
using $\langle 0 < y \cdot i \rangle \langle \text{conic } S \rangle \text{ conic-mul face-of-conic that}(1) \ y(1)$ by
fastforce
moreover have $\text{inverse}(y \cdot i) *_{\mathbb{R}} y \in \{x. x \cdot i = 1\}$
using $\langle y \cdot i > 0 \rangle$ by (simp add: field-simps)
ultimately show $f \cap \{x. x \cdot i = 1\} \neq \{ \}$
by blast
qed
then show ?thesis
by (simp add: conic-hyperplane-eq that)
qed
have $\text{card } \{f. f \text{ face-of } S \wedge \text{aff-dim } f = 1 + \text{int } d\}$
 $= \text{card } \{f. f \text{ face-of } p \wedge \text{aff-dim } f = \text{int } d\}$
proof (intro bij-betw-same-card bij-betw-imageI)
show $\text{inj-on } (\lambda f. f \cap \{x. x \cdot i = 1\}) \{f. f \text{ face-of } S \wedge \text{aff-dim } f = 1 +$
 $\text{int } d\}$
by (smt (verit) conic-hyperplane-eq inj-on-def mem-Collect-eq of-nat-less-0-iff)
show $(\lambda f. f \cap \{x. x \cdot i = 1\}) ' \{f. f \text{ face-of } S \wedge \text{aff-dim } f = 1 + \text{int } d\}$
 $= \{f. f \text{ face-of } p \wedge \text{aff-dim } f = \text{int } d\}$
using aff-1d conic-eq-f conic-face-p
by (fastforce simp: image-iff face-S-imp-face-p dim-f-hyperplane)

```

      qed
    }
  then show ?thesis
    by force
  qed
  finally show ?thesis .
  qed
  ultimately show ?thesis
    by auto
  qed
  qed
  qed

```

corollary *Euler-poincare-special:*

```

  fixes p :: 'n::euclidean-space set
  assumes 2 ≤ DIM('n) polytope p i ∈ Basis and affp: affine hull p = {x. x · i
= 0}
  shows (∑ d = 0..DIM('n) - 1. (-1) ^ d * card {f. f face-of p ∧ aff-dim f =
d}) = 1
  proof -
    { fix d
      have eq: image((+) i) ' {f. f face-of p} ∩ image((+) i) ' {f. aff-dim f = int d}
        = image((+) i) ' {f. f face-of p} ∩ {f. aff-dim f = int d}
      by (auto simp: aff-dim-translation-eq)
      have card {f. f face-of p ∧ aff-dim f = int d} = card (image((+) i) ' {f. f
face-of p ∧ aff-dim f = int d})
      by (simp add: inj-on-image card-image)
      also have ... = card (image((+) i) ' {f. f face-of p} ∩ {f. aff-dim f = int d})
      by (simp add: Collect-conj-eq image-Int inj-on-image eq)
      also have ... = card {f. f face-of (+) i ' p ∧ aff-dim f = int d}
      by (simp add: Collect-conj-eq faces-of-translation)
      finally have card {f. f face-of p ∧ aff-dim f = int d} = card {f. f face-of (+)
i ' p ∧ aff-dim f = int d} .
    }
  then
  have (∑ d = 0..DIM('n) - 1. (-1) ^ d * card {f. f face-of p ∧ aff-dim f = d})
    = (∑ d = 0..DIM('n) - 1. (-1) ^ d * card {f. f face-of (+) i ' p ∧ aff-dim
f = int d})
  by simp
  also have ... = 1
  proof (rule Euler-Poincare-lemma)
    have ∧x. [i ∈ Basis; x · i = 1] ⇒ ∃y. y · i = 0 ∧ x = y + i
    by (metis add-cancel-left-left eq-diff-eq inner-diff-left inner-same-Basis)
    then show affine hull (+) i ' p = {x. x · i = 1}
    using ⟨i ∈ Basis⟩ unfolding affine-hull-translation affp by (auto simp:
algebra-simps)
  qed (use assms polytope-translation-eq in auto)
  finally show ?thesis .
  qed

```

1.7 Now Euler-Poincare for a general full-dimensional polytope

theorem *Euler-Poincare-full*:

fixes $p :: 'n::euclidean-space\ set$

assumes $polytope\ p\ \text{aff-dim}\ p = DIM('n)$

shows $(\sum d = 0..DIM('n). (-1) ^ d * (card\ \{f. f\ \text{face-of}\ p \wedge \text{aff-dim}\ f = d\})) = 1$

proof –

define $augm :: 'n \Rightarrow 'n \times real$ **where** $augm \equiv \lambda x. (x, 0)$

define S **where** $S \equiv augm\ 'p$

obtain $i :: 'n$ **where** $i: i \in Basis$

by (*meson SOME-Basis*)

have *bounded-linear augm*

by (*auto simp: augm-def bounded-linearI'*)

then have *polytope S*

unfolding *S-def using polytope-linear-image <polytope p> bounded-linear.linear*

by *blast*

have *face-pS*: $\bigwedge F. F\ \text{face-of}\ p \iff augm\ 'F\ \text{face-of}\ S$

using *S-def <bounded-linear augm> augm-def bounded-linear.linear face-of-linear-image*

inj-on-def **by** *blast*

have *aff-dim-eq[simp]*: $\text{aff-dim}\ (augm\ 'F) = \text{aff-dim}\ F$ **for** F

using *<bounded-linear augm> aff-dim-injective-linear-image bounded-linear.linear*

unfolding *augm-def inj-on-def* **by** *blast*

have $*$: $\{F. F\ \text{face-of}\ S \wedge \text{aff-dim}\ F = \text{int}\ d\} = (\text{image}\ augm)\ ' \{F. F\ \text{face-of}\ p \wedge \text{aff-dim}\ F = \text{int}\ d\}$

(**is** *?lhs = ?rhs*) **for** d

proof

have $\bigwedge G. \llbracket G\ \text{face-of}\ S; \text{aff-dim}\ G = \text{int}\ d \rrbracket$

$\implies \exists F. F\ \text{face-of}\ p \wedge \text{aff-dim}\ F = \text{int}\ d \wedge G = augm\ 'F$

by (*metis face-pS S-def aff-dim-eq face-of-imp-subset subset-imageE*)

then show *?lhs \subseteq ?rhs*

by (*auto simp: image-iff*)

qed (*auto simp: image-iff face-pS*)

have *ceqc*: $card\ \{f. f\ \text{face-of}\ S \wedge \text{aff-dim}\ f = \text{int}\ d\} = card\ \{f. f\ \text{face-of}\ p \wedge \text{aff-dim}\ f = \text{int}\ d\}$ **for** d

unfolding $*$

by (*rule card-image*) (*auto simp: inj-on-def augm-def*)

have $(\sum d = 0..DIM('n \times real) - 1. (-1) ^ d * \text{int}\ (card\ \{f. f\ \text{face-of}\ S \wedge \text{aff-dim}\ f = \text{int}\ d\})) = 1$

proof (*rule Euler-poincare-special*)

show $2 \leq DIM('n \times real)$

by *auto*

have *snd0*: $(a, b) \in \text{affine hull}\ S \implies b = 0$ **for** $a\ b$

using *S-def <bounded-linear augm> affine-hull-linear-image augm-def* **by** *blast*

moreover have $\bigwedge a. (a, 0) \in \text{affine hull}\ S$

using *S-def <bounded-linear augm> aff-dim-eq-full affine-hull-linear-image assms(2) augm-def* **by** *blast*

ultimately show $\text{affine hull}\ S = \{x. x \cdot (0::'n, 1::real) = 0\}$

```

    by auto
  qed (auto simp: ⟨polytope S⟩ Basis-prod-def)
  then show ?thesis
    by (simp add: ceqc)
  qed

```

In particular, the Euler relation in 3 dimensions

corollary *Euler-relation:*

```

  fixes p :: 'n::euclidean-space set
  assumes polytope p aff-dim p = 3 DIM('n) = 3
  shows (card {v. v face-of p ∧ aff-dim v = 0} + card {f. f face-of p ∧ aff-dim f
= 2}) - card {e. e face-of p ∧ aff-dim e = 1} = 2
proof -
  have ∧x. [x face-of p; aff-dim x = 3] ⇒ x = p
    using assms by (metis face-of-aff-dim-lt less-irrefl polytope-imp-convex)
  then have 3: {f. f face-of p ∧ aff-dim f = 3} = {p}
    using assms by (auto simp: face-of-refl polytope-imp-convex)
  have (∑ d = 0..3. (-1) ^ d * int (card {f. f face-of p ∧ aff-dim f = int d})) =
  1
    using Euler-Poincare-full [of p] assms by simp
  then show ?thesis
    by (simp add: sum.atLeast0-atMost-Suc-shift numeral-3-eq-3 3)
  qed

```

end

References

- [1] I. Lakatos. *Proofs and Refutations: The Logic of Mathematical Discovery*. 1976.
- [2] J. Lawrence. A short proof of Euler's relation for convex polytopes. *Canadian Mathematical Bulletin*, 40(4):471–474, 1997.