Formalizing Polyhedral Geometry

(in Lean 4)

What are polyhedra?

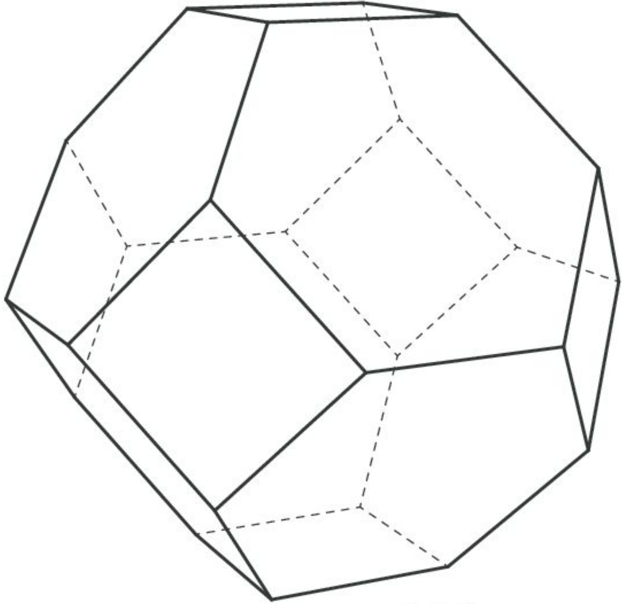
- A <u>halfspace</u> is a set of vectors on one side of a hyperplane

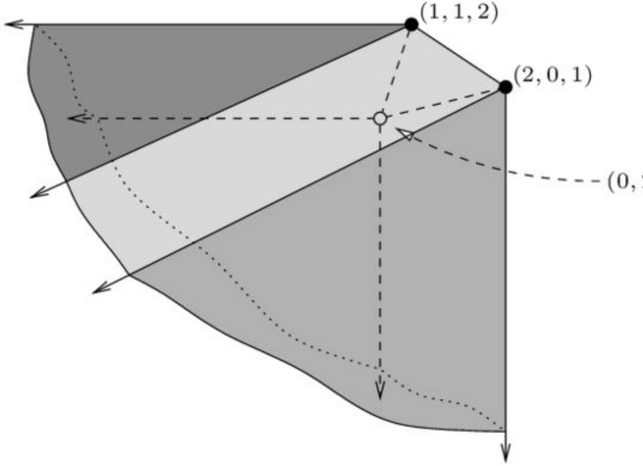
$$H = \{x \in \mathbb{R}^n \mid a^T x \leq b\}$$

- A <u>polyhedron</u> is an intersection of finitely many halfspaces.

$$P = \bigcap_{i=1}^{k} H_i$$

- A <u>polytope</u> is a bounded polyhedron.





Our Project

- Focused on
  - Establishing basic definitions
    - Halfspaces
    - Polyhedra
    - Cones
    - Conical and convex hulls
  - Formalizing introductory theorems
    - Convexity of halfspaces and polyhedra
    - Carathéodory’s theorem

https://github.com/uw-math-ai/lean-polyhedral-geometry

Formalizing is Hard

- Written proofs often leave out “intuitive” details
  - Implicit induction
    - “Minimal element” proofs
    - Induction upon non-trivial types like cardinalities
  - “Letting things go to infinity”
  - Implicit inclusion of non trivial facts:
    - Boundness requires a “bornology”
    - Hopping between a particular basis and being basis agnostic
  - Notion of Finite
    - `Finite` vs. `Finset`
    - Finsets have a cardinality of type natural number, while sets have a special cardinality type
      - Need coercion to apply theorems

Formalizing is Hard (cont'd)

- Mathlib theorems are often not immediately compatible with our situation and frequently require "massaging" or rephrasing of our definitions to apply nicely
- Difficult to build the "most general" definition of our math objects
  - `Type` vs. `Type*`
  - Indexing via natural numbers vs. an arbitrary finite set
- Difficult to choose between structures and predicates for our math objects

A Few Definitions

```
def Halfspace (s : Set V) : Prop :=
  ∃ (f : V →ₗ[ℝ] ℝ) (c : ℝ), s = { x | f x ≤ c }
def Polyhedron (s : Set V) : Prop :=
  ∃ (I : Type) (H : I → Set V), Finite I ∧ (∀ i : I, Halfspace
  (H i)) ∧ s = ⋂ (i : I), H i
def conicalHull (s : Set V) : Set V :=
  { x | ∃ (t : Finset V) (a : V → ℝ),
  (∀ v ∈ t, 0 ≤ a v) ∧ ↑t ⊆ s ∧ x = ∑ v ∈ t, a v • v }
```

Carathéodory’s Theorem

```
theorem caratheordory (s : Set V) (x : V) (h : x ∈ conicalHull s) :
  ∃ (t : Finset V), ↑t ⊆ s ∧ t.card ≤ Module.finrank ℝ V ∧ x ∈ conicalHull
t
```

Proof Outline:

- Since `x ∈ conicalHull s`, there exists a conical combination `x = ∑ v ∈ t, a v • v`
- If `t.card` less than dimension of the vector space, we are done
- Otherwise, pick a minimal set of vectors `t` which form a conical combination of `x`
- Since `t.card` is greater than the dimension of the vector space, we can remove one of the vectors from the conical combination using linear dependence
  - This is tricky, since we must always have the scalar multiples be positive. (Harder than showing you can removing a vector from span).
- This is a contradiction since we picked the smallest set of vectors

```
theorem caratheordory (s : Set V) (x : V) (h : x ∈ conicalHull s) :
  ∃ (t : Finset V), ↑t ⊆ s ∧ t.card ≤ Module.finrank ℝ V ∧ x ∈ conicalHull t := by
  rcases min_elt (conicalCombo_cards s x) (conicalCombo_cards_nonempty s x h) with ⟨n, h', h_minimality⟩
  rcases h' with ⟨t, ⟨a, h_a_nonneg, h_t_subset, h_x_combo⟩, rfl⟩
  rcases le_or_gt t.card (Module.finrank ℝ V) with h_t_card | h_t_card
  . use t, h_t_subset, h_t_card, t, a
  apply False.elim
  have := reduce_by_one t a x h_a_nonneg h_x_combo h_t_card
  rcases this with ⟨t',a',t'_le_t,t'_sub_t,a'_nonneg,t'_x_conic_combo⟩
  have t'_subset_s : ↑t' ⊆ s := by
      have : (↑t' : Set V) ⊆ (t : Set V) := by
      exact t'_sub_t
      apply subset_trans this h_t_subset
  have t'_is_in_conicalCombos : t'.card ∈ conicalCombo_cards s x := by
      use t'
      use ⟨a',a'_nonneg,t'_subset_s,t'_x_conic_combo⟩
  have := h_minimality t'.card t'_le_t
  show False
  exact this t'_is_in_conicalCombos
```

Sources

- Polyhedral Combinatorics by Gaku Liu, https://drive.google.com/file/d/1TRg7iQ0RpIRteF2IUiKIe3E-YagVnkH0/view

- Lean 4 Index, https://leanprover-community.github.io/mathlib4_docs/

- Theorem Proving in Lean, https://leanprover.github.io/theorem_proving_in_lean4/