Formalization and Finite Algebra

Clea Bergsman, Katherine Buesing, Sahan Wijetunga

2025-07-10

Outline

Recall: What is a Bilinear Form?

Definition

A **bilinear form** is a map $\beta : V \times W \to K$, where V and W are K-vector spaces and K is a field, when

- $\beta(v_1 + v_2, w) = \beta(v_1, w) + \beta(v_2, w)$
- $\beta(v, w_1 + w_2) = \beta(v, w_1) + \beta(v, w_2)$
- $\beta(\lambda v, w) = \beta(v, \lambda w) = \lambda\beta(v, w)$

hold for all $v \in V$, $w \in W$, and $\lambda \in K$.

Recall: Symmetric, Alternating, and Skew Bilinear Forms

Symmetric

$\beta(v, w) = \beta(w, v)\ \forall\ v, w$

Alternating

$\beta(v, v) = 0,\ \forall\ v$

Skew

$\beta(v, w) = -\beta(w, v),\ \forall\ v, w$

Note: Bilinear forms that are **anti-symmetric** are both alternating and skew-symmetric.

Reflexive Bilinear Forms

Definition

A bilinear form β is **reflexive** if $\beta(v, w) = 0 \iff \beta(w, v) = 0 \; \forall \; v, w \in V$

```
lemma alt_is_reflexive (β:BilinForm k V) (h:Alt β) : IsRefl β := by
  intro v w l
  have hv : β v v = 0 := by apply h
  have hw : β w w = 0 := by apply h
  have h1 : β (v+w) (v+w) = (β v) v + (β w) v + (β v) w + (β w) w :=
    calc
    (β (v+w)) (v+w) = (β v) (v+w) + (β w) (v+w) := by
        rw [LinearMap.BilinForm.add_left]
    _ = (β v) v + (β w) v + (β v) w + (β w) w := by
      rw [LinearMap.BilinForm.add_right v v w, LinearMap.BilinForm.add_right w v w,
        ← add_assoc]; ring
  have hvw : β (v+w) (v+w) = 0 := by apply h
  rw [hv, hw, hvw, zero_add, add_zero, add_comm] at h1
  have h2: 0 + -(β w) v = (β v) w + (β w) v + -(β w) v := by
    apply (@add_right_cancel_iff _ _ _ (-(β w) v) 0 ((β v) w + (β w) v)).mpr h1
  rw [l, zero_add] at h1
  symm at h1
  exact h1
```

```
lemma symm_is_reflexive (β:BilinForm k V) (h:Symm β) :
    IsRefl β := by
  intro v w l
  have h1: (β v) w = (β w) v := by apply h
  rw [l] at h1
  symm at h1
  exact h1
```

Bilinear Forms as Matrices

- Given a basis $(v_1, \dots, v_n)$ for V we can form a matrix A corresponding to the bilinear form $B : V \times V \to k$ by defining $A_{ij} := B(v_i, v_j)$. Thus,

$$B(x, y) = [x]^T A[y]$$

  for $x, y \in V$.
- The dot product on $\mathbf{R}^n$ has matrix

$$I_n = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{bmatrix}$$

Bilinear Forms as Matrices

- Symmetric forms have matrix satisfying $A^T = A$.

$$\begin{bmatrix} 1 & 2 \\ 2 & 3 \end{bmatrix}$$

- Alternating forms have matrix satisfying $A^T = -A$.

$$\begin{bmatrix} 0 & 5 \\ -5 & 0 \end{bmatrix}$$

- Nondegenerate forms have matrix with $\det(A) \neq 0$.

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \quad (\det = 4 \cdot 1 - 3 \cdot 2 \neq 0)$$

Nondegenerate Definition

Theorem

Let β be a bilinear form on V, $M = [\beta(v_i, v_j)]$, and $v_1, ..., v_n$ a basis of V The following are equivalent:

- $det(M) \neq 0$
- $\forall w \in V \ \beta(v, w) = 0 \implies v = 0$
- $\forall v \in V \ \beta(v, w) = 0 \implies w = 0$

```
theorem nondeg_rank (β : BilinForm k V) [FiniteDimensional k V]
(n : ℕ ) (h: Module.rank k V = n) (b : Basis (Fin n) k V):
  (LinearMap.BilinForm.Nondegenerate β) ↔
  Matrix.rank (BilinForm.toMatrix b β ) = n := by
  constructor
  -- Nondegenerate β → rank M = n
  intro hn
  have nondeg : β.Nondegenerate := by apply hn
  have nul_zero : LinearMap.ker β = ⊥ := by
    apply LinearMap.BilinForm.nondegenerate_iff_ker_eq_bot.mp;
    exact nondeg
  let M : Matrix (Fin n) (Fin n) k := BilinForm.toMatrix b β
  have h1 : Module.rank k ↥(range β ) + (Module.rank k (ker β))
  = Module.rank k V  :=
      by apply LinearMap.rank_range_add_rank_ker
  have zero : (Module.rank k ↥(ker β)) = 0 := by
  rw [nul_zero]; simp
  rw [zero, add_zero] at h1
  rw [h] at h1
  simp at h1
  exact h1
```

Theorems

```
--  rank M = n → Nondegenerate β
intro hn
have h1 : Module.rank k ↥(range β ) + (Module.rank k (ker β))
= Module.rank k V  :=
    by apply LinearMap.rank_range_add_rank_ker
simp at h1
rw [hn] at h1
have nul_zero : Module.rank k ↥(ker β) = 0 := by
  apply (add_eq_left _ _ (↑n) (Module.rank k ↥(ker β))).mp
  at h1
simp at nul_zero
apply LinearMap.BilinForm.nondegenerate_iff_ker_eq_bot.mpr
exact nul_zero
```

Creating a Basis using Disjoint Bases

Theorem

Let V be a vector space over a field k, and let W_1 and W_2 be subspaces of V such that $W_1 + W_2 = V$ and $W_1 \cap W_2 = \{0\}$. Let B_1 and B_2 be bases of W_1 and W_2, respectively.

- *We can create a basis for V using the union of B_1 and B_2*

Additional Theorems

Theorem

Let V be a vector space over a field k, and let W_1 and W_2 be subspaces of V such that $W_1 \cap W_2 = \{0\}$. Let e_1 and e_2 be linearly independent sets of W_1 and W_2, respectively.

- *The disjoint union of e_1 and e_2 is linearly independent.*

Theorem

Let V be a vector space over a field k, and let W_1 and W_2 be two subspaces of a larger vector space V, where the direct sum of W_1 and W_2 is equal to V. Let s_1 and s_2 be sets of vectors that span W_1 and W_2, respectively.

- *The union of s_1 and s_2 span V.*

Additional Theorems in Lean

- Theorem: Linear independence by transverse subspaces

```
theorem lin_indep_by_transverse_subspaces
    (k V : Type) [Field k] [AddCommGroup V] [Module k V]
    (I₁ I₂ : Type) [Fintype I₁] [Fintype I₂]
    (b₁ : I₁ → V) (b₂ : I₂ → V)
    (b1_indep : LinearIndependent k b₁)
    (b2_indep : LinearIndependent k b₂)
    (W₁ W₂ : Submodule k V) (h_int : W₁ ⊓ W₂ = ⊥)
    (hbw1 : ∀ i, b₁ i ∈ W₁) (hbw2 : ∀ i, b₂ i ∈ W₂)
    [DecidableEq I₁] [DecidableEq I₂]
    : LinearIndependent k (Sum.elim b₁ b₂)
```

Additional Theorems in Lean Continued

- Theorem: Span of union of sets

```
lemma union_span' (W₁ W₂ : Submodule k V) (s₁ s₂ : Set V)
  (hs₁: W₁ = Submodule.span k s₁)
  (hs₂: W₂ = Submodule.span k s₂)
  (hw: ⊤ = W₁ ⊔ W₂)
  : ⊤ = Submodule.span k (s₁ ∪ s₂)
```

Pen and Paper Proof

- In order to show that the disjoint union of bases B_1 and B_2 is a basis for V, we want to show that this disjoint union is both linearly independent and spans the entirety of V.
- Since $W_1 \cap W_2 = \{0\}$, and B_1 and B_2 are both individually linearly independent, we can conclude that their union is linearly independent.
- Now we want to show that their union spans all of V. Since $W_1 + W_2 = V$, and B_1 and B_2 span W_1 and W_2, respectively, we can conclude that their union spans V.

Lean Proof

```
def basis_of_direct_sum
        (W₁ W₂ : Submodule k V)
        (ι₁ ι₂ : Type) [Fintype ι₁]
        [Fintype ι₂]
        (B₁ : Basis ι₁ k W₁)
        (B₂ : Basis ι₂ k W₂)
        (hspan : W₁ ⊔ W₂ = (⊤: Submodule k V))
        (hindep : W₁ ⊓ W₂ = (⊥:Submodule k V))
        [DecidableEq ι₁] [DecidableEq ι₂]
        [FiniteDimensional k V]:
        Basis (ι₁ ⊕ ι₂) k V := by
```

Lean Proof

```
have hli: LinearIndependent k (Sum.elim
(W₁.subtype ∘ B₁) (W₂.subtype ∘ B₂)) := by
    apply lin_indep_by_transverse_subspaces
    · apply LinearIndependent.map' B₁.linearIndependent W₁.subtype
    (by simp)
    · apply LinearIndependent.map' B₂.linearIndependent W₂.subtype
    (by simp)
    · have k₀ : Disjoint W₁ W₂ := by
        rw[disjoint_iff]
        exact hindep
        rw[Disjoint.eq_bot k₀]
    · simp
    · simp
```

Lean Proof

```
have hsp: ⊤ ≤ Submodule.span k (Set.range (Sum.elim
(W₁.subtype ∘ B₁) (W₂.subtype ∘ B₂))) := by
   simp
   rw[union_span']
   exact W₁
   exact W₂
   · exact span_range (Basis.span_eq B₁)
   · exact span_range (Basis.span_eq B₂)
   · rw[hspan]
   exact Basis.mk hli hsp
```

Hyperbolic Forms

- A 2-dimensional vector space V is called **Hyperbolic** if there exists a basis (e, f) such that the form has matrix

$$H_2 = \begin{pmatrix} 0 & 1 \\ * & 0 \end{pmatrix}$$

- Note if V is symmetric as well, it has matrix $\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$, and if alternating then matrix $\begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}$.
- A generic vector space V is *Hyperbolic* if it has a basis making its form equal to $\begin{bmatrix} H_2 & & & \\ & H_2 & & \\ & & \ddots & \\ & & & H_2 \end{bmatrix}$

Alternating is Hyperbolic

Theorem

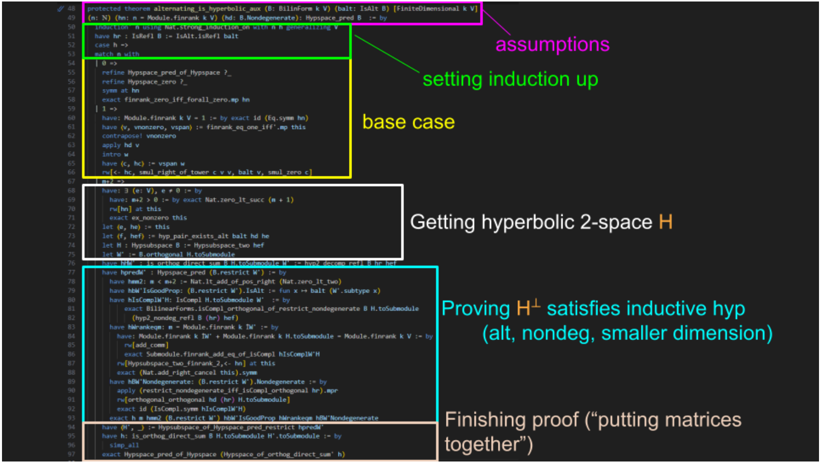
Every nondegenerate (finite dimensional) alternating bilinear form B on V is Hyperbolic.

Pen and Paper Proof (That nondegenerate (finite dimensional) alternating bilinear forms are Hyperbolic)

- If $V \neq 0$, pick $e \neq 0$ in V.
  - From nondegeneracy, pick v with $B(e, v) \neq 0$.
  - From scaling, pick $f = \lambda v$ with $B(e, f) = 1$.
- Then $H = \text{Span}(e, f)$ has matrix $\begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}$ and is Hyperbolic.
- Fact: $V = H \oplus H^{\perp}$.
- Check:
  - $H^{\perp}$ Alternating
  - $H^{\perp}$ Nondegenerate
  - $\dim(H^{\perp}) < \dim(V)$
- From induction, $H^{\perp}$ is Hyperbolic. Thus,

$$A_V = \begin{pmatrix} A_H & 0 \\ 0 & A_{H^{\perp}} \end{pmatrix} = \begin{pmatrix} \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix} & \\ & \begin{pmatrix} H_2 & & \\ & \ddots & \\ & & H_2 \end{pmatrix} \end{pmatrix} = \begin{pmatrix} H_2 & & & \\ & H_2 & & \\ & & \ddots & \\ & & & H_2 \end{pmatrix}$$

Lean Proof (That nondegenerate (finite dimensional) alternating bilinear forms are Hyperbolic)



```
protected theorem alternating_is_hyperbolic_aux (B: BilinForm k V) (balt: IsAlt B) [FiniteDimensional k V]
(n: ℕ) (hn: n = Module.finrank k V) (hd: B.Nondegenerate): Hypspace_pred B  := by
  induction' n using Nat.strong_induction_on with n h generalizing V
  have hr : IsRefl B := IsAlt.isRefl balt
  case h =>
  match n with
  | 0 =>
    refine Hypspace_pred_of_Hypspace ?_
    refine Hypspace_zero ?_
    symm at hn
    exact finrank_zero_iff_forall_zero.mp hn
  | 1 =>
    have: Module.finrank k V = 1 := by exact id (Eq.symm hn)
    have ⟨v, vnonzero, vspan⟩ := finrank_eq_one_iff'.mp this
    contrapose! vnonzero
    apply hd v
    intro w
    have ⟨c, hc⟩ := vspan w
    rw[<- hc, smul_right_of_tower c v v, balt v, smul_zero c]
  | m+2 =>
    have: ∃ (e: V), e ≠ 0 := by
      have: m+2 > 0 := by exact Nat.zero_lt_succ (m + 1)
      rw[hn] at this
      exact ex_nonzero this
    let ⟨e, he⟩ := this
    let ⟨f, hef⟩ := hyp_pair_exists_alt balt hd he
    let H : Hypsubspace B := Hypsubspace_two hef
    let W' := B.orthogonal H.toSubmodule
    have hHW' : is_orthog_direct_sum B H.toSubmodule W' := hyp2_decomp_refl B hr hef
    have hpredW' : Hypspace_pred (B.restrict W') := by
      have hmm2: m < m+2 := Nat.lt_add_of_pos_right (Nat.zero_lt_two)
      have hbW'IsGoodProp: (B.restrict W').IsAlt := fun x ↦ balt (W'.subtype x)
      have hIsComplW'H: IsCompl H.toSubmodule W'  := by
          exact BilinearForms.isCompl_orthogonal_of_restrict_nondegenerate B H.toSubmodule
            (hyp2_nondeg_refl B (hr) hef)
      have hWrankeqm: m = Module.finrank k ↥W' := by
        have: Module.finrank k ↥W' + Module.finrank k H.toSubmodule = Module.finrank k V := by
          rw[add_comm]
          exact Submodule.finrank_add_eq_of_isCompl hIsComplW'H
        rw[Hypsubspace_two_finrank_2,<- hn] at this
        exact (Nat.add_right_cancel this).symm
      have hBW'Nondegenerate: (B.restrict W').Nondegenerate := by
        apply (restrict_nondegenerate_iff_isCompl_orthogonal hr).mpr
        rw[orthogonal_orthogonal hd (hr) H.toSubmodule]
        exact id (IsCompl.symm hIsComplW'H)
      exact h m hmm2 (B.restrict W') hbW'IsGoodProp hWrankeqm hBW'Nondegenerate
    have ⟨H', _⟩ := Hypsubspace_of_Hypspace_pred_restrict hpredW'
    have h: is_orthog_direct_sum B H.toSubmodule H'.toSubmodule := by
      simp_all
    exact Hypspace_pred_of_Hypspace (Hypspace_of_orthog_direct_sum' h)
```

Lean Proof v.s. Paper Proof

The lean proof was

- Quick
- Efficient
- Similar in length and style to the paper one
- Relied heavily on the power of Mathlib and a >1000 line auxiliary file setting up the theory of Hyperbolic spaces

Symmetric Forms

Theorem

Every nondegenerate (finite dimensional, field characteristic not 2) symmetric bilinear form is the direct sum of a Hyperbolic form and a definite (anisotropic) form.

- The paper proof is similar in length to the alternating one.
- The lean proof was 10x longer
  - Involves subspaces and “direct sum”

Lean Hyperbolic Space Definitions

```
@[ext]
structure Hypspace (B: BilinForm k V) where
  I: Type
  basis : Basis (I ⊕ I) k V
  pred: Hypspace_fun_pred B basis

...

@[ext]
structure Hypsubspace (B: BilinForm k V) where
  I: Type
  coe : I ⊕ I → V
  pred: Hypspace_fun_pred B coe
```

References

1. Avigad, J. Buzzard, K. Lewis R. Y. Massot, P. (2020). *Mathematics in Lean*.
2. Liesen, J. Mehrmann, V. (2015). *Linear Algebra*.
3. Reich, E. (2005, February 28). Bilinear Forms. Retrieved July 10, 2005, from https://math.mit.edu/ dav/bilinearforms.pdf

Thank you!

Special thanks to Dr. George McNinch and the REU