IMPERIAL

Formalizing Group Action in Lean

Project 1: Group Actions in Lean 4

Frankie Feng-Cheng WANG
Department of Mathematics, Imperial College London
MATH70040 Formalising Mathematics

Outline

What is a Group Action?

Motivation:

- Groups act on sets, formalizing symmetry
- Connects abstract algebra with concrete transformations
- Foundation for orbit-stabilizer theorems, Burnside's lemma, etc.

Project Goals:

- ✓ Core Definitions: `GroupAction` typeclass, Faithful, Transitive
- ✓ Concrete Examples: Symmetric group, D_4, conjugation, etc.
- ✓ Key Theorems: Permutation representation (Thm 16.3), Stabilizer subgroup (Thm 16.12)

Main Results

Theorem 16.3: Permutation Representation

Every group action induces a group homomorphism $\phi : G \to \mathrm{Sym}(X)$

Theorem 16.12: Stabilizer Subgroup

For any $x \in X$, the stabilizer $G_x = \{g \in G \mid g \cdot x = x\}$ is a subgroup of G

Both theorems are fully formalized in Lean 4 with explicit proofs.

Group Action Definition

Mathematical Foundation

Mathematics

A group G **acts** on a set X via a function

$$\cdot : G \times X \to X$$

Axioms:

1. **Associativity**: $(g_1 g_2) \cdot x = g_1 \cdot (g_2 \cdot x)$
2. **Identity**: $1 \cdot x = x$

for all $g_1, g_2 \in G, x \in X$

Lean 4 Code

```
16  class GroupAction (G : Type*) [Monoid G] (X : Type*)
    ↪  where
17
      ↪  /-- The action function that applies a group element to
18    act : G → X → X
19
      ↪  /-- The multiplication axiom: `(g₁ * g₂) • x = g₁ • (g
20    ga_mul : ∀ g₁ g₂ x, act (g₁ * g₂) x = act g₁ (act g₂ x)
21    /-- The identity axiom: `1 • x = x`. -/
22    ga_one : ∀ x, act 1 x = x
```

Orbits and Stabilizers

Faithful Actions

Mathematics

An action is **faithful** if distinct group elements act differently.

Formally:

$$\forall g_1, g_2 \in G, (\forall x, g_1 \cdot x = g_2 \cdot x) \Rightarrow g_1 = g_2$$

Intuition: The action "faithfully represents" the group structure

Lean 4 Code

```
28 def GroupAction.faithful {G : Type*} [Group G] {X :
↪  Type*} [GroupAction G X] : Prop :=
29   ∀ g₁ g₂ : G, (∀ x : X, GroupAction.act g₁ x =
  ↪  GroupAction.act g₂ x) → g₁ = g₂
```

Source: `/Defs.lean:28-29`

Transitive Actions

Mathematics

An action is **transitive** if any element can be moved to any other.

Formally:

$$\forall x_1, x_2 \in X, \exists g \in G,\ g \cdot x_1 = x_2$$

Intuition: The group "acts transitively" on the entire set

Lean 4 Code

```
36  def GroupAction.transitive {G : Type*} [Group G] {X :
    ↪  Type*} [GroupAction G X] : Prop :=
37    ∀ x₁ x₂ : X, ∃ g : G, GroupAction.act g x₁ = x₂
```

Source: `/Defs.lean:36-37`

Example 1: Symmetric Group on X

$\mathrm{Sym}(X)$ acts on X by applying permutations.

```
instance permGroupAction (X : Type*) : GroupAction (Equiv.Perm X) X :=
  { act := fun g x => g x
    ga_mul := by
      intro g1 g2 x
      rfl
    ga_one := by
      intro x
      rfl }
```

One of the most fundamental actions.

Source: /Examples.lean:21-28

Example 1: Symmetric Group Properties

Faithful & Transitive

Faithful

```
theorem permGroupAction_faithful (X : Type*) :                    37
  GroupAction.faithful (G := Equiv.Perm X) (X := X) :=            38
by                                                                39
  intro g₁ g₂ h                                                   40
  ext x                                                           41
  exact h x                                                       42
                                                                  43
```

Transitive

```
theorem permGroupAction_transitive
  (X : Type*) [DecidableEq X] :
  GroupAction.transitive (G := Equiv.Perm X) (X := X) :=
by
  intro x₁ x₂
  refine ⟨Equiv.swap x₁ x₂, ?_⟩
  simp [GroupAction.act]
```

Source: /Examples.lean:30-43

Example 2: Left Multiplication

G acts on itself by left multiplication: $g_1 \cdot g_2 = g_1 * g_2$

```
instance groupAsGSet (G : Type*) [Group G] : GroupAction G G :=
  { act := fun g1 g2 => g1 * g2
    ga_mul := by
      intro g1 g2 g3
      rw [mul_assoc]
    ga_one := by
      intro g
      rw [one_mul] }
```

This action is **transitive** but **not faithful**.

Source: `/Examples.lean:47-54`

Example 3: Subgroup Action

A subgroup $H \leq G$ acts on G by left multiplication.

```
instance subgroupAsGSet {G : Type*} [Group G] (H : Subgroup G) : GroupAction H G :=
  { act := fun h g => (h : G) * g
    ga_mul := by
      intros
      simp [mul_assoc]
    ga_one := by
      intros
      simp }
```

Coercion from H to G handled implicitly.

Source: `/Examples.lean:59-66`

Example 4: Conjugation

Conjugation: $h \cdot g = hgh^{-1}$

```
instance subgroupAsGSetConjugation {G : Type*} [Group G] (H : Subgroup G) : GroupAction H H :=
  { act := fun h g => h * g * h⁻¹
    ga_mul := by
      intro g₁ g₂ g₃
      simp
      rw [← mul_assoc g₁]
      rw [mul_assoc g₁ g₂]
      rw [← mul_assoc]
    ga_one := by
      intros
      simp }
```

Orbits are **conjugacy classes**; stabilizers are **centralizers**.

Source: `/Examples.lean:69-79`

Example 5: Scalar Action on $\mathbb{C}^n$

$\mathbb{C}^\times$ acts on $\mathbb{C}^n$ by componentwise multiplication.

```
instance vectorSpaceAsCStarSet (n : ℕ) : GroupAction (ℂˣ) (Fin n → ℂ) :=
  { act := fun r v => fun i => (r : ℂ) * v i
    ga_mul := by
      intros r1 r2 v
      ext i
      simp [mul_assoc]
    ga_one := by
      intro v
      ext i
      simp }
```

Uses `Fin n → ℂ` to represent $\mathbb{C}^n$ in Lean.

Source: `/Examples.lean:84-93`

Example 6: Dihedral Group D_4

D_4 (symmetries of a square) acts on $\mathbb{Z}/4$ (vertices).

```
/-- The dihedral group D₄ of order 8, symmetry group of a square. -/
abbrev D4 := DihedralGroup 4

/-- Type alias for vertices of a square, indexed by ZMod 4. -/
abbrev Vertex := ZMod 4

/-- Type alias for sides of a square, indexed by ZMod 4. -/
abbrev Side := ZMod 4

/-- Type alias for midpoints of sides of a square, indexed by ZMod 4. -/
abbrev Midpoint := ZMod 4

/-- The action of D₄ on ZMod 4 representing vertices/sides/midpoints of a square. -/
def d4Act (g : D4) (x : ZMod 4) : ZMod 4 :=
  match g with
  -- Rotation r_i: x ↦ x - i
  | DihedralGroup.r i => x - i
  -- Reflection s_i: x ↦ i - x
  | DihedralGroup.sr i => i - x

/-- The dihedral group D₄ acts on ZMod 4 (vertices of a square). -/
instance d4ActionZMod4 : GroupAction D4 (ZMod 4) :=
  { act := d4Act
    ga_mul := by
      intro g1 g2 x
      simp [d4Act, sub_eq_add_neg] <;> ring
    ga_one := by
```

Orbit-Stabilizer Theorem Setup

Relating |G|, |Orb(x)|, |Stab(x)|

Burnside's Lemma Application

Counting Fixed Points

Theorem 16.3: Permutation Representation

Theorem 16.3

Every group action induces a group homomorphism $\phi : G \to \mathrm{Sym}(X)$ such that:

$$\phi(g)(x) = g \cdot x \quad \text{for all } g \in G, x \in X$$

This theorem connects group actions with permutation representations.

Proof Strategy: 5 Steps

1. **Define** σ_g: For each $g \in G$, construct $\sigma_g : X \to X$
2. **Prove Bijection**: Show σ_g is bijective (left/right inverse)
3. **Construct** ϕ: Package σ_g as $\phi(g) \in \mathrm{Sym}(X)$
4. **Verify Homomorphism**: Prove $\phi(g_1 g_2) = \phi(g_1) \circ \phi(g_2)$
5. **Package Theorem**: Combine all pieces into final statement

Step 1-2: Define σ_g & Prove Bijection

Define σ_g

```
/-- Defines the action map `σ_g : X → X` by `x ↦ g • x`.
def sigma (g : G) : X → X :=
  fun x => GroupAction.act g x
```

Prove Bijective

```
def sigmaPerm (g : G) : Equiv.Perm X :=
  { toFun := sigma g
    invFun := sigma g⁻¹
    left_inv := by
      intro x
      -- Step 1: combine `g⁻¹` and `g` using the action
      ↪ axiom.
      calc
        GroupAction.act g⁻¹ (GroupAction.act g x) =
            GroupAction.act (g⁻¹ * g) x := by
          simpa using (GroupAction.ga_mul g⁻¹ g x).symm
        -- Step 2: simplify `g⁻¹ * g` to `1`.
        _ = GroupAction.act (1 : G) x := by
          -- Alternative: use `congrArg` with
          ↪ `inv_mul_cancel g`.
          -- congrArg (fun t => GroupAction.act t x)
          ↪ (inv_mul_cancel g)
          simp
        -- Step 3: the identity acts as the identity on
        ↪ `X`.
        _ = x := GroupAction.ga_one x
    right_inv := by
      intro x
      -- Step 1: combine `g` and `g⁻¹` using the action
      ↪ axiom.
      calc
        GroupAction.act g (GroupAction.act g⁻¹ x) =
            GroupAction.act (g * g⁻¹) x := by
          simpa using (GroupAction.ga_mul g g⁻¹ x).symm
```

Step 3-4: Construct ϕ & Verify Homomorphism

Define $\phi : G \to \mathrm{Sym}(X)$

```
def phi (g : G) : Equiv.Perm X :=
  sigmaPerm g
```

Prove $\phi(g_1 g_2) = \phi(g_1) \circ \phi(g_2)$

```
lemma phi_mul (g₁ g₂ : G) :
  phi (X := X) (g₁ * g₂) = phi (X := X) g₁ * phi (X := X) g₂ := by
  -- Extensionality reduces equality of permutations to pointwise equality.
  apply Equiv.ext
  intro (x : X)
  calc
    -- Step 1: expand `phi` and the action definition.
    phi (g₁ * g₂) x = GroupAction.act (g₁ * g₂) x := rfl
    -- Step 2: use the action law to separate `g₁` and `g₂`.
    _ = GroupAction.act g₁ (GroupAction.act g₂ x) := GroupAction.ga_mul g₁ g₂ x
```

Step 5: Package the Theorem

Final Lean Theorem

```
/-- The action yields a permutation representation. -/
theorem group_action_to_perm_representation :
  ∃ (ψ : G → Equiv.Perm X),
    (∀ g x, ψ g x = GroupAction.act g x) ∧
    (∀ g₁ g₂, ψ (g₁ * g₂) = ψ g₁ * ψ g₂) ∧
    (ψ 1 = 1) := by
  exact ⟨phi, ⟨phi_apply, ⟨phi_mul, phi_one⟩⟩⟩
```

Proof by construction: We exhibit ϕ and verify all properties.

- Action property: $\phi(\mathsf{g})(\mathsf{x}) = \mathsf{g} \cdot \mathsf{x}$
- Homomorphism property: $\phi(\mathsf{g}_1\mathsf{g}_2) = \phi(\mathsf{g}_1) \circ \phi(\mathsf{g}_2)$
- Identity property: $\phi(1) = \mathsf{id}$

Source: `/Permutation.lean:102-108`

Theorem 16.12: Stabilizer Subgroup

Mathematics

Definition: The **stabilizer** of $x \in X$ is:

$$G_x = \{g \in G \mid g \cdot x = x\}$$

Theorem 16.12: G_x is a subgroup of G

Proof requires:

1. $1 \in G_x$ (identity)
2. $g_1, g_2 \in G_x \Rightarrow g_1 g_2 \in G_x$ (closure)
3. $g \in G_x \Rightarrow g^{-1} \in G_x$ (inverses)

Lean Structure

```
25  def stabilizerSet (x : X) : Set G :=
26    { g : G | GroupAction.act g x = x }
27
28  /-- The stabilizer `G_x` as a subgroup of `G`. -/
29  def stabilizer (x : X) : Subgroup G := by
30    exact
31      { carrier := stabilizerSet (G := G) (X := X) x
32        one_mem' := by
33          -- The identity fixes every x.
```

Lean requires explicit proofs of all three subgroup axioms.

Source: /Stabilizer.lean:25-33

Stabilizer in Lean: Definition & Proof

Part 1: Define Stabilizer Set (Stabilizer.lean:25-26)

```
def stabilizerSet (x : X) : Set G :=
  { g : G | GroupAction.act g x = x }
```

Part 2: Construct Subgroup (Stabilizer.lean:29-53, key excerpts)

```
def stabilizer (x : X) : Subgroup G := by
  exact
    { carrier := stabilizerSet (G := G) (X := X) x
      one_mem' := by
        -- The identity fixes every x.
        simp [stabilizerSet, GroupAction.ga_one x]
      mul_mem' := by
        intro g₁ g₂ hg₁ hg₂
        -- Closure under multiplication via the action axiom.
        GroupAction.act (g₁ * g₂) x = GroupAction.act g₁ (GroupAction.act g₂ x) := by
          simpa using (GroupAction.ga_mul g₁ g₂ x)
```

What Lean Guarantees

- ✓ **Type Correctness**: All functions type-check, no runtime type errors
- ✓ **No Hidden Assumptions**: Every axiom explicitly declared
- ✓ **Axiom Matching**: Our proofs use only standard mathlib axioms (no choice beyond mathlib)
- ✓ **Subgroup Verification**: `Stabilizer` is proven to satisfy all subgroup axioms
- ✓ **Homomorphism Properties**: ϕ proven to preserve group structure

Lean's proof assistant guarantees mathematical correctness—no gaps, no handwaving.

Challenges & Lessons Learned

Technical Challenges:

- **Typeclass Resolution**: Manual instance declarations for `GroupAction`
- **Equiv Mechanism**: Understanding `Equiv.Perm` vs raw bijections
- **Coercions**: Handling coercion from `Subgroup H` to `H` in examples
- **Function Extensionality**: Using `Equiv.ext` to prove permutation equality

Lessons:

- Read Mathlib source code for patterns
- Use `calc` mode for clarity
- Lean enforces rigor: every step must be justified

Conclusion & Future Work

What We Achieved:

- Formalized core group action theory in Lean 4
- 7 concrete examples from abstract algebra
- 2 fundamental theorems with complete proofs

Future Extensions:

- **Orbit-Stabilizer Theorem**: $|G| = |\text{Orb}(x)| \cdot |G_x|$
- **Burnside's Lemma**: Counting orbits under symmetry
- **Applications**: Cayley's Theorem, Sylow Theorems
- **Category Theory**: Generalizations to functors and natural transformations

Thank You!

Questions?

GitHub Repository

`https://github.com/FrankieeW/GroupAction`

Version: `v1.2.1-lean-only` (stable)

Contact

Frankie Feng-Cheng WANG
Department of Mathematics, Imperial College London
MATH70040 Formalising Mathematics