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

1. **Introduction**
   Motivation & Main Results

2. **Core Definitions**
   GroupAction, Faithful, Transitive

3. **Examples**
   Symmetric Group, D_4, and more

4. **Key Theorems**
   Theorem 16.3 (Permutation Representation) & Theorem 16.12 (Stabilizer Subgroup)

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

```
class GroupAction (G : Type*)
    [Monoid G] (X : Type*) where
  act : G → X → X
  ga_mul : ∀ g₁ g₂ x,
    act (g₁ * g₂) x =
    act g₁ (act g₂ x)
  ga_one : ∀ x, act 1 x = x
```

Source: `/Defs.lean:15-22`

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

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

Source: `/Defs.lean:26-28`

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

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

Source: `/Defs.lean:33-35`

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:20-27`

Example 1: Symmetric Group Properties

Faithful & Transitive

Faithful

```
theorem permFaithful (X : Type*) [Nonempty X] :
    faithful (Equiv.Perm X) X := by
  intro g1 g2 h
  ext x
  have := h x
  exact this
```

Transitive

```
theorem permTransitive (X : Type*) [Nonempty X] :
    transitive (Equiv.Perm X) X := by
  intro x y
  obtain ⟨z⟩ := ‹Nonempty X›
  use Equiv.Perm.swap z x |>.trans
       (Equiv.Perm.swap z y)
  simp
```

Source: `/Examples.lean:29-44`

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:46-54`

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
    ga_mul := by intro ⟨h1,_⟩ ⟨h2,_⟩ g; simp; rw [mul_assoc]
    ga_one := by intro g; simp; rw [one_mul] }
```

Coercion from H to G handled implicitly.

Source: `/Examples.lean:58-65`

Example 4: Conjugation

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

```
instance conjugationAction (G : Type*) [Group G] : GroupAction G G :=
  { act := fun g h => g * h * g⁻¹
    ga_mul := fun g1 g2 g => by
      calc g1 * g2 * g * (g1 * g2)⁻¹
           = g1 * g2 * g * g2⁻¹ * g1⁻¹ := by rw [mul_inv_rev]
           _ = g1 * (g2 * g * g2⁻¹) * g1⁻¹ := by rw [mul_assoc, mul_assoc]
    ga_one := fun g => by simp }
```

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

Source: `/Examples.lean:67-78`

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

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

```
instance complexScalarAction (n : ℕ) :
    GroupAction ℂ* (Fin n → ℂ) :=
  { act := fun ⟨c, _⟩ v => fun i => c * v i
    ga_mul := by intro ⟨c1,_⟩ ⟨c2,_⟩ v; ext i; ring
    ga_one := by intro v; ext i; simp }
```

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

Source: `/Examples.lean:82-92`

Example 6: Dihedral Group D_4

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

```
abbrev D4 := DihedralGroup 4

def d4Act : D4 → (Fin 4) → (Fin 4)
  | .r k, v => ((v.val + k) % 4 : Fin 4)
  | .sr k, v => ((k - v.val) % 4 : Fin 4)

instance d4ActionZMod4 : GroupAction D4 (Fin 4) :=
  { act := d4Act
    ga_mul := by intro g1 g2 v;
      cases g1 <;> cases g2 <;> simp [d4Act]; ring_nf
    ga_one := by intro v; simp [d4Act] }
```

Geometric interpretation: Rotations and reflections of square vertices.

Source: `/Examples.lean:108-156`

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

```
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
      calc GroupAction.act g⁻¹
              (GroupAction.act g x)
        = GroupAction.act
            (g⁻¹ * g) x := by
            rw [←ga_mul]
        _ = x := by simp
    right_inv := ... }
```

Key insight: g^{-1} provides the inverse map.

Source: `/Permutation.lean:24-67`

Step 3-4: Construct ϕ & Verify Homomorphism

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

```
def phi : G → Equiv.Perm X :=
  fun g => sigmaPerm g
```

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

```
lemma phi_mul (g₁ g₂ : G) :
    phi (g₁ * g₂) = phi g₁ * phi g₂ := by
  apply Equiv.ext
  intro x
  calc phi (g₁ * g₂) x
      = GroupAction.act (g₁ * g₂) x := rfl
    _ = GroupAction.act g₁
        (GroupAction.act g₂ x) :=
        GroupAction.ga_mul g₁ g₂ x
```


Highlight: Uses `GroupAction.ga_mul` axiom for associativity.

Step 5: Package the Theorem

Final Lean Theorem

```
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: $\phi(g)(x) = g \cdot x$
- Homomorphism property: $\phi(g_1 g_2) = \phi(g_1) \circ \phi(g_2)$
- Identity property: $\phi(1) = \mathrm{id}$

Source: `/Permutation.lean:102-114`

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

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

def stabilizer (x : X) :
    Subgroup G :=
  { carrier := stabilizerSet x
    one_mem' := ...
    mul_mem' := ...
    inv_mem' := ... }
```

Lean requires explicit proofs of all three subgroup axioms.

Source: `/Stabilizer.lean:22-56`

Stabilizer in Lean: Definition & Proof

Part 1: Define Stabilizer Set (Stabilizer.lean:22-23)

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

Part 2: Construct Subgroup (Stabilizer.lean:26-54, key excerpts)

```
def stabilizer (x : X) : Subgroup G := by
  exact
    { carrier := stabilizerSet x
      one_mem' := by simp [stabilizerSet, GroupAction.ga_one x]
      mul_mem' := by
        intro g₁ g₂ hg₁ hg₂
        calc GroupAction.act (g₁ * g₂) x
            = GroupAction.act g₁ (GroupAction.act g₂ x) := by
              simp using (GroupAction.ga_mul g₁ g₂ x)
          _ = GroupAction.act g₁ x := by rw [hg₂]
          _ = x := hg₁
      inv_mem' := by
        intro g hg
        calc GroupAction.act g⁻¹ x
            = GroupAction.act g⁻¹ (GroupAction.act g x) := by rw [hg]
            = x := by simp [GroupAction.ga_mul, GroupAction.ga_one] }
```

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**: $|G| = |\mathrm{Orb}(x)| \cdot |G_x|$
- **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