MAG P≠NP: A Group-Theoretic Approach

Lean 4 Formalization Blueprint

Version 1.0

Masaru Numagaki

January 2026

Abstract

This blueprint documents the Lean 4 formalization of P≠NP in the MAG (Minimum Asymmetry Gap) model. The MAG framework translates computational complexity into group theory: **polytime** corresponds to solvable groups, **NP-hard** corresponds to non-solvable groups (with A_5 as the minimal non-solvable core), and **solves** corresponds to group isomorphism.

Main Results (all sorry-free):

1. **Theorem 3.1**: MAG internal separation (unconditional)
2. **Theorem 4.6**: A_5 Universal Barrier (unconditional)
3. **Theorem 4.7**: Solvable vanishing depth (unconditional)
4. **Theorem 5.4**: Bridge theorem (conditional on TranslationInterface)
5. **Theorem 5.7**: Toy model end-to-end verification (unconditional)

Proof Status: Core/ directory: sorry = 0, axiom = 0

Contents

Chapter 1

Introduction

This blueprint presents a formal verification of P≠NP in the MAG (Minimum Asymmetry Gap) model, implemented in Lean 4 [dMU21] with Mathlib [The20]. The MAG framework translates computational complexity into group theory:

- **Polytime** corresponds to *solvable groups*
- **NP-hard** corresponds to *non-solvable groups* (with A_5 as the minimal non-solvable core)
- **Solves** corresponds to *group isomorphism*

1.1 Main Results

The main formalized results are:

1. **MAG internal separation** (Theorem 3.1): Unconditional, fully formalized
2. A_5 **Universal Barrier** (Theorem 4.6): Unconditional, fully formalized
3. **Solvable vanishing depth** (Theorem 4.7): Unconditional, fully formalized
4. **Bridge theorem** (Theorem 5.4): Conditional on TranslationInterface, fully formalized
5. **Toy model verification** (Theorem 5.7): End-to-end, unconditional, fully formalized

1.2 Scope and Non-claims

Remark 1.2.1 (Scope). This formalization does NOT unconditionally claim to solve the Clay P vs NP problem. What we prove:

- **Unconditional:** MAG-internal P≠NP, where definitions are group-theoretic
- **Conditional:** If TranslationInterface holds for model M, then P≠NP in M

The TranslationInterface separates the formalized core from unformalized translation assumptions.

1.3 Historical Foundation

The MAG framework can be understood as the **computational analogue of Klein's Erlangen Program**. Just as Klein (1872) proposed classifying geometries by their transformation groups, MAG classifies computational problems by their symmetry groups and algorithms by their operation groups.

The MAG framework builds on several foundational results:

- **Galois Theory** [Gal46]: The non-solvability of A_5 and the connection between solvability and radical solutions
- **Klein's Erlangen Program** [Kle93]: The methodology of translating mathematical objects into group-theoretic objects for classification
- **Krohn-Rhodes Decomposition** [KR65, RS09]: Every finite automaton decomposes into simple groups and aperiodic monoids
- **Barrington's Theorem** [Bar86]: NC^1 = width-5 branching programs over S_5
- **Cook-Levin Theorem** [Coo71]: SAT is NP-complete, establishing the existence of NP-hard problems
- **Classification of Finite Simple Groups** [GLS18]: A_5 is the smallest non-solvable simple group
- **Feit-Thompson Theorem** [FT63]: All groups of odd order are solvable

For a comprehensive treatment of computational complexity, see Arora and Barak [AB09]. For group theory foundations, see Rotman [Rot12].

1.4 File Structure

Paper Section	Lean File	Status
Section 2 (Definitions)	`Core/MAG_P_neq_NP.lean`	✓ sorry-free
Section 3 (Main Theorem)	`Core/MAG_P_neq_NP.lean`	✓ sorry-free
Section 4.1–4.5	`Core/A5_Barrier.lean`	✓ sorry-free
Section 4.6	`Core/A5_Barrier.lean`	✓ sorry-free
Section 4.7	`Core/BarringtonBarrier.lean`	✓ sorry-free
Section 5.1–5.3	`Core/Bridge/Interface.lean`	✓ sorry-free
Section 5.4	`Core/Bridge/Theorem.lean`	✓ sorry-free
Section 5.5	`Core/Bridge/ToyInstantiation.lean`	✓ sorry-free
Appendix A	`Support/Barrington/*.lean`	✓ sorry-free
Appendix A	`Support/Circuits/KrohnRhodes.lean`	✓ sorry-free
Appendix A	`Support/Syntactic/Theory.lean`	✓ sorry-free
Appendix B	`Blueprint/*`	Design docs

Chapter 2

MAG Model Definitions

This chapter introduces the core definitions of the MAG framework. *(Paper: Section 2)*

2.1 Problem and Algorithm Structures

Definition 2.1.1 (ProblemInstance)**.** `MAG.Core.ProblemInstance` ✓ Lean*(Paper: Definition 2.1)*

A *problem instance* is represented by its symmetry group G.

```
structure ProblemInstance where
  G : Type
  [group_G : Group G]
  [fintype_G : Fintype G]
```

Definition 2.1.2 (Algorithm)**.** `MAG.Core.Algorithm` ✓ Lean*(Paper: Definition 2.2)*

An *algorithm* is represented by the group of operations it can generate.

```
structure Algorithm where
  G_alg : Type
  [group_alg : Group G_alg]
  [fintype_alg : Fintype G_alg]
```

2.2 MAG Complexity Classes

Definition 2.2.1 (IsMAGPolyTime)**.** `MAG.Core.IsMAGPolyTime` ✓ Lean *(Paper: Definition 2.3)*

An algorithm is *MAG-polytime* if its operation group is solvable. This definition is justified by the Krohn-Rhodes decomposition theorem [KR65]: standard Boolean gates decompose into solvable components.

$$a \in \mathrm{P}_{\mathrm{MAG}} \quad :\Longleftrightarrow \quad A(a) \text{ is solvable}$$

```
class IsMAGPolyTime (alg : Algorithm) : Prop where
  solvable_structure : IsSolvable alg.G_alg
```

Definition 2.2.2 (IsMAGNPHard)**.** `MAG.Core.IsMAGNPHard` ✓ Lean *(Paper: Definition 2.4)*

A problem is *MAG-NP-hard* if its symmetry group is non-solvable.

$$p \text{ is MAG-NP-hard} \quad :\Longleftrightarrow \quad \neg\,\mathrm{IsSolvable}(G(p))$$

```
class IsMAGNPHard (prob : ProblemInstance) : Prop where
  non_solvable : ň IsSolvable prob.G
```

Definition 2.2.3 (Solves)**.** `MAG.Core.Solves` ✓ Lean *(Paper: Definition 2.5)*

An algorithm *solves* a problem if their groups are isomorphic.

$$\text{Solves}(a, p) \quad :\Longleftrightarrow \quad A(a) \simeq G(p) \quad \text{(group isomorphism)}$$

```
def Solves (alg : Algorithm) (prob : ProblemInstance) : Prop :=
  Nonempty (alg.G_alg * prob.G)
```

Definition 2.2.4 (MAG_P_equals_NP)**.** `MAG.Core.MAG_P_equals_NP` ✓ Lean *(Paper: Definition 2.6)*

The MAG version of P=NP states that every MAG-NP-hard problem can be solved by some MAG-polytime algorithm.

```
def MAG_P_equals_NP : Prop :=
   (prob : ProblemInstance) [IsMAGNPHard prob],
     (alg : Algorithm), IsMAGPolyTime alg  Solves alg prob
```

2.3 The A_5 Instance

Definition 2.3.1 (A5_Instance)**.** `MAG.Core.A5_Instance` ✓ Lean

The alternating group A_5 as a problem instance.

```
def A5_Instance : ProblemInstance where
  G := alternatingGroup (Fin 5)
```

Chapter 3

Core Algebraic Facts

3.1 Properties of A_5

Theorem 3.1.1 (A_5 is non-solvable)**.** `MAG.Core.A5_not_solvable` ✓ *Lean*

The alternating group A_5 is not solvable:

$$\neg\,\mathrm{IsSolvable}(A_5)$$

This is the fundamental algebraic fact underlying MAG separation, first established by Galois [Gal46]. The proof uses the simplicity of A_5 and explicit non-commuting witnesses.

Theorem 3.1.2 (A_5 is simple)**.** `MAG.Core.A5_is_simple` ✓ *Lean*

A_5 is a simple group (from Mathlib: `alternatingGroup.isSimpleGroup_five`).

Theorem 3.1.3 ($|A_5| = 60$)**.** `MAG.Core.A5_card` ✓ *Lean*

The cardinality of A_5 is exactly 60 (proven via `native_decide`). By the classification of finite simple groups [GLS18], A_5 is the smallest non-solvable simple group. The Feit-Thompson theorem [FT63] further establishes that non-solvability requires even order.

Corollary 3.1.4 (A_5 is MAG-NP-hard)**.** `MAG.Core.A5_is_NPHard` ✓ *Lean*

Since A_5 is non-solvable, the A_5 instance is MAG-NP-hard.

3.2 Solvability Preservation

Lemma 3.2.1 (Isomorphism preserves solvability)**.** `MAG.Core.solvable_of_equiv` ✓ *Lean (Paper: Lemma 2.7)*

If $G \simeq H$ (group isomorphism) and G is solvable, then H is solvable.

$$G \simeq H \wedge \mathrm{IsSolvable}(G) \implies \mathrm{IsSolvable}(H)$$

Chapter 4

MAG Separation Theorem

(Paper: Section 3)

4.1 Main Theorem

Theorem 4.1.1 (MAG P≠NP — Theorem 3.1)**.** `MAG.Core.MAG_P_neq_NP` ✓ *Lean (Paper: Theorem 3.1)*

$P \neq NP$ in the MAG model:

$$\neg \text{MAG\_P\_equals\_NP}$$

Proof sketch:

1. *Assume MAG $P = NP$ for contradiction.*
2. *Take A_5 as an NP-hard instance (non-solvable).*
3. *By assumption, there exists a polytime algorithm a solving A_5.*
4. *a is polytime $\Rightarrow$ $A(a)$ is solvable.*
5. *a solves $A_5 \Rightarrow A(a) \simeq A_5$.*
6. *By Lemma 3.2.1, A_5 would be solvable.*
7. *Contradiction with Theorem 3.1.1.*

Status: *sorry = 0, axiom = 0*

Chapter 5

A_5 Universal Barrier

(Paper: Section 4)

5.1 Syntactic Systems and Deep Structures

Definition 5.1.1 (SyntacticSystem). `MAG.Core.SyntacticSystem` ✓ Lean*(Paper: Definition 4.1)*

A *syntactic system* represents a formal derivation system with solvable group structure.

```
structure SyntacticSystem where
  Derivations : Type*
  [group : Group Derivations]
  [fintype : Fintype Derivations]
  is_solvable : IsSolvable Derivations
```

Definition 5.1.2 (DeepStructure). `MAG.Core.DeepStructure` ✓ Lean*(Paper: Definition 4.2)*

A *deep structure* contains an A_5 embedding via injective homomorphism.

```
structure DeepStructure where
  SymmetryGroup : Type*
  [group : Group SymmetryGroup]
  [fintype : Fintype SymmetryGroup]
  embedding : alternatingGroup (Fin 5) * SymmetryGroup
  embedding_injective : Function.Injective embedding
```

Definition 5.1.3 (CanDescribe). `MAG.Core.CanDescribe` ✓ Lean *(Paper: Definition 4.3)*

A syntactic system *can describe* a deep structure if there exists a group isomorphism.

```
def CanDescribe (sys : SyntacticSystem) (deep : DeepStructure) : Prop :=
  Nonempty (sys.Derivations * deep.SymmetryGroup)
```

5.2 Kernel Propagation

Lemma 5.2.1 (Non-solvable kernel propagates). *`MAG.Core.nonsolvable_of_injective_from_nonsolvable`* ✓ *Lean(Paper: Lemma 4.4)*

If H is non-solvable and $f : H \hookrightarrow G$ is injective, then G is non-solvable.

$$\neg\text{IsSolvable}(H) \wedge (H \hookrightarrow G) \implies \neg\text{IsSolvable}(G)$$

Corollary 5.2.2 (A_5 embedding implies non-solvability). *`MAG.Core.A5_embedding_implies_nonsolvable`* ✓ *Lean (Paper: Lemma 4.5)*

Any deep structure (with A_5 embedding) has a non-solvable symmetry group.

5.3 Universal Barrier Theorem

Theorem 5.3.1 (A_5 Universal Barrier — Theorem 4.6)**.** `MAG.Core.A5_Universal_Barrier`
✓ *Lean (Paper: Theorem 4.6)*

No solvable syntactic system can describe a deep structure containing an A_5 embedding:

$$\forall S\ D,\ \neg\,\mathrm{CanDescribe}(S, D)$$

This theorem reflects the Krohn-Rhodes complexity hierarchy [KR65, RS09]: A_5 is the minimal Level 2 group, unreachable from Level 0–1 structures.

Status: *sorry = 0, axiom = 0*

Chapter 6

Commutator Tree Vanishing Depth

(Paper: Section 4.7)

6.1 Commutator Trees

Definition 6.1.1 (CommTerm). `MAG.Core.BarringtonBarrier.CommTerm` ✓ Lean

A balanced commutator tree of depth n:

- Depth 0: a variable
- Depth $n+1$: commutator $[t, u]$ of two depth-n terms

```
inductive CommTerm ( : Type u) :   Type u
  | var :   CommTerm  0
  | comm {n : } : CommTerm  n  CommTerm  n  CommTerm  (n+1)
```

Theorem 6.1.2 (Evaluation lands in derived series). *MAG.Core.BarringtonBarrier.CommTerm.eval_mem_de*
✓ *Lean*

For any commutator term t of depth n, $\mathrm{eval}(t) \in \mathrm{derivedSeries}(G, n)$.

6.2 Vanishing Depth

Theorem 6.2.1 (Solvable vanishing depth — Theorem 4.7). *MAG.Core.BarringtonBarrier.CommTerm.solva*
✓ *Lean (Paper: Theorem 4.7)*

For any solvable group G, there exists k such that all commutator trees of depth $n \geq k$ evaluate to 1:

$$\exists k,\ \forall n \geq k,\ \forall t : \mathrm{CommTerm}_n,\ \mathrm{eval}(t) = 1$$

Status: *sorry = 0, axiom = 0*

Theorem 6.2.2 (A_5 has no vanishing depth). *MAG.Core.BarringtonBarrier.A5_no_vanishing_depth*
✓ *Lean*

A_5 has no finite vanishing depth (its derived series is constantly $\top$).

Chapter 7

Translation Interface and Bridge

(Paper: Section 5)

7.1 Abstract Computational Model

Definition 7.1.1 (ComputationalModel)**.** `MAG.Core.Bridge.ComputationalModel` ✓ Lean *(Paper: Definition 5.1)*

An abstract interface for any computational model:

```
class ComputationalModel (M : Type*) where
  Problem : Type*
  Algorithm : Type*
  problemSymmetry : Problem  Type*
  algorithmOperations : Algorithm  Type*
  solves : Algorithm  Problem  Prop
  isPolyTime : Algorithm  Prop
  isNPHard : Problem  Prop
```

7.2 Translation Interface

Definition 7.2.1 (TranslationInterface)**.** `MAG.Core.Bridge.TranslationInterface` ✓ Lean *(Paper: Definition 5.2)*

The interface connecting a computational model to MAG semantics:

1. **(A) polyTimeIsSolvable:** Polytime algorithms have solvable operation groups (justified by Krohn-Rhodes [KR65])
2. **(B) npHardHasA5:** NP-hard problems have A_5 embedding in symmetry (justified by Barrington [Bar86] + syntactic theory)
3. **(C) solvesImpliesGroupCorrespondence:** Solving implies group isomorphism

7.3 Standard P=NP Definition

Definition 7.3.1 (Standard_P_equals_NP)**.** `MAG.Core.Bridge.Standard_P_equals_NP` ✓ Lean *(Paper: Definition 5.3)*

P = NP on model M: every NP-hard problem can be solved by some polytime algorithm.

```
def Standard_P_equals_NP (M : Type*) [ComputationalModel M] : Prop :=
   (p : Problem), isNPHard p
     a : Algorithm, isPolyTime a  solves a p
```

7.4 Bridge Theorem

Theorem 7.4.1 (Bridge P≠NP — Theorem 5.4)**.** *MAG.Core.Bridge.Bridge_P_neq_NP* ✓ *Lean*
(Paper: Theorem 5.4)

If TranslationInterface holds for model M and an NP-hard problem exists, then $P \neq NP$:

$$\text{TranslationInterface}(M) \wedge (\exists p,\ \text{isNPHard}(p)) \implies \neg\,\text{Standard\_P\_equals\_NP}(M)$$

Status: *sorry = 0, axiom = 0*

Corollary 7.4.2 (Bridge P≠NP (with existence))**.** *MAG.Core.Bridge.Bridge_P_neq_NP'* ✓ *Lean*

Variant that takes existence of NP-hard problem as explicit hypothesis.

Chapter 8

Toy Model Verification

(Paper: Section 5.5)

8.1 BarringtonToy Model

Definition 8.1.1 (BarringtonToyModel). `MAG.Core.Bridge.BarringtonToyModel` ✓ Lean *(Paper: Construction 5.5)*

A toy computational model where:

- Problems ARE their symmetry groups
- Algorithms ARE their operation groups
- Conditions are satisfied BY DEFINITION

Theorem 8.1.2 (Toy satisfies TranslationInterface). *`MAG.Core.Bridge.instTranslationInterfaceBarring`*
✓ *Lean*

The BarringtonToy model satisfies all three TranslationInterface conditions.

8.2 NP-Hard Witness

Definition 8.2.1 (S_5 Problem). `MAG.Core.Bridge.S5_Problem` ✓ Lean *(Paper: Lemma 5.6)*

A toy problem whose symmetry group is $S_5 = \mathrm{Perm}(\mathrm{Fin}\ 5)$.

Theorem 8.2.2 (S_5 is NP-hard in toy model). *`MAG.Core.Bridge.S5_is_nphard`* ✓ *Lean*

The S_5 problem is NP-hard in the toy sense (contains A_5 via subgroup inclusion).

Theorem 8.2.3 (NP-hard problems exist in toy model). *`MAG.Core.Bridge.toy_exists_nphard`*
✓ *Lean*

There exists an NP-hard problem in the BarringtonToy model.

8.3 End-to-End Result

Theorem 8.3.1 (BarringtonToy P≠NP — Theorem 5.7). *`MAG.Core.Bridge.BarringtonToy_P_neq_NP`*
✓ *Lean*

$P \neq NP$ holds in the BarringtonToy model:

$$\neg\,\mathrm{Standard\_P\_equals\_NP}(\mathrm{BarringtonToyModel})$$

This theorem demonstrates the bridge plumbing works end-to-end without sorry.

Status: *sorry = 0, axiom = 0*

Chapter 9

Barrington's Theorem Components

(Paper: Appendix A)

This chapter formalizes the core algebraic content of Barrington's theorem [Bar86].

9.1 Non-Commuting Elements

Definition 9.1.1 (commutator). `MAG.Support.Barrington.Complete.commutator` ✓ Lean

Group commutator: $[a, b] = a^{-1}b^{-1}ab$.

Theorem 9.1.2 (σ and τ do not commute). *`MAG.Support.Barrington.Complete.__not_commute`*
✓ *Lean*

$[\sigma, \tau] \neq 1$, *proven via* `decide`.

Theorem 9.1.3 (AND via commutator). *`MAG.Support.Barrington.Complete.commutator_and_table`*
✓ *Lean*

The commutator encodes AND truth table:

$$[\sigma, \tau] \neq 1, \quad [\sigma, 1] = 1, \quad [1, \tau] = 1, \quad [1, 1] = 1$$

9.2 Branching Programs and Length Analysis

Definition 9.2.1 (BranchingProgram). `MAG.Support.Barrington.Complete.BranchingProgram`
✓ Lean

A branching program as a list of instructions.

Theorem 9.2.2 (Commutator program length). *`MAG.Support.Barrington.Complete.length_commutatorPr`*
✓ *Lean*

$|[P_1, P_2]| = 2|P_1| + 2|P_2|$.

Theorem 9.2.3 (Translation length bound). *`MAG.Support.Barrington.Complete.translate_length_bound`*
✓ *Lean*

$|\text{translate}(C)| \leq 4^{\text{depth}(C)}$.

Theorem 9.2.4 (NC^1 has poly BP). *`MAG.Support.Barrington.Arithmetic.NC1_has_poly_BP`*
✓ *Lean*

NC^1 functions have polynomial-size width-5 branching programs.

Chapter 10

Krohn-Rhodes Circuit Solvability

(Paper: Appendix A)

This chapter proves that standard Boolean circuits have solvable operation groups, based on the Krohn-Rhodes decomposition theorem [KR65, RS09].

Definition 10.0.1 (GateType)**.** `MAG.Support.Circuits.KrohnRhodes.GateType` ✓ Lean

Standard Boolean gate types: AND, OR, NOT, INPUT.

Definition 10.0.2 (GateGroup)**.** `MAG.Support.Circuits.KrohnRhodes.GateGroup` ✓ Lean

The group component of each gate's syntactic monoid.

Theorem 10.0.3 (Gate groups are solvable)**.** *`MAG.Support.Circuits.KrohnRhodes.gate_group_solvable`* ✓ *Lean*

Every basic gate has a solvable group.

Theorem 10.0.4 (Circuit Composition Principle)**.** *`MAG.Support.Circuits.KrohnRhodes.Circuit_Composit`* ✓ *Lean*

Standard Boolean circuits have solvable operation groups:

$$\forall C,\ \mathrm{IsSolvable}(\mathrm{CircuitOpGroup}(C))$$

Chapter 11

Syntactic Theory

(Paper: Appendix A)

This chapter proves the fundamental theorem of syntactic monoids.

Definition 11.0.1 (SyntacticMonoid)**.** `MAG.Support.Syntactic.Theory.SyntacticMonoid` ✓ Lean

The quotient $\Sigma^*/\sim_L$.

Definition 11.0.2 (WordProblem)**.** `MAG.Support.Syntactic.Theory.WordProblem` ✓ Lean

The word problem of a group G: $w \in L \Leftrightarrow w.\text{prod} = 1$.

Theorem 11.0.3 (Word problem syntactic monoid isomorphism)**.** *`MAG.Support.Syntactic.Theory.wordPro`* ✓ *Lean*

The syntactic monoid of the word problem of G is isomorphic to G:

$$\text{SyntacticMonoid}(\text{WordProblem}_G) \simeq G$$

Appendix: Lean Artifact Map

Core Theorems

Theorem/Definition	File	Status
MAG_P_neq_NP (Thm 3.1)	Core/MAG_P_neq_NP.lean	✓ sorry-free
A5_Universal_Barrier (Thm 4.6)	Core/A5_Barrier.lean	✓ sorry-free
solvable_vanishing_depth (Thm 4.7)	Core/BarringtonBarrier.lean	✓ sorry-free
Bridge_P_neq_NP (Thm 5.4)	Core/Bridge/Theorem.lean	✓ sorry-free
BarringtonToy_P_neq_NP (Thm 5.7)	Core/Bridge/ToyInstantiation.lean	✓ sorry-free

Support Modules

Module	File	Status
Barrington algebraic content	Support/Barrington/Complete.lean	✓ sorry-free
Barrington arithmetic	Support/Barrington/Arithmetic.lean	✓ sorry-free
Krohn-Rhodes solvability	Support/Circuits/KrohnRhodes.lean	✓ sorry-free
Syntactic monoid theory	Support/Syntactic/Theory.lean	✓ sorry-free

Verification Command

Entry point: `Main.lean`

To verify: `lake build`

Bibliography

[AB09] Sanjeev Arora and Boaz Barak. *Computational Complexity: A Modern Approach*. Cambridge University Press, 2009. Standard reference for computational complexity theory.

[Bar86] David A. Barrington. Bounded-width polynomial-size branching programs recognize exactly those languages in NC^1. In *Proceedings of the 18th Annual ACM Symposium on Theory of Computing (STOC)*, pages 1–5, 1986. Establishes NC^1 = width-5 branching programs over S_5.

[Coo71] Stephen A. Cook. The complexity of theorem-proving procedures. In *Proceedings of the 3rd Annual ACM Symposium on Theory of Computing (STOC)*, pages 151–158, 1971. Introduces NP-completeness; proves SAT is NP-complete.

[dMU21] Leonardo de Moura and Sebastian Ullrich. The Lean 4 theorem prover and programming language. In *Proceedings of the 28th International Conference on Automated Deduction (CADE)*, volume 12699 of *Lecture Notes in Computer Science*, pages 625–635. Springer, 2021. Introduction to Lean 4's architecture and design.

[FT63] Walter Feit and John G. Thompson. Solvability of groups of odd order. *Pacific Journal of Mathematics*, 13(3):775–1029, 1963. Proves all groups of odd order are solvable (255 pages).

[Gal46] Évariste Galois. Oeuvres mathématiques. *Journal de mathématiques pures et appliquées*, 11:381–444, 1846. Posthumous publication of Galois's work on solvability by radicals.

[GLS18] Daniel Gorenstein, Richard Lyons, and Ronald Solomon. *The Classification of the Finite Simple Groups*. Mathematical Surveys and Monographs. American Mathematical Society, 1994–2018. Multi-volume work completing the classification of finite simple groups.

[Kle93] Felix Klein. Vergleichende Betrachtungen über neuere geometrische Forschungen. *Mathematische Annalen*, 43:63–100, 1893. Originally published as a pamphlet in Erlangen, 1872. English translation: "A comparative review of recent researches in geometry". Establishes the Erlangen Program: geometry as the study of invariants under transformation groups.

[KR65] Kenneth Krohn and John Rhodes. Algebraic theory of machines. I. Prime decomposition theorem for finite semigroups and machines. *Transactions of the American Mathematical Society*, 116:450–464, 1965. Establishes the Krohn-Rhodes decomposition theorem for finite automata.

[Rot12] Joseph J. Rotman. *An Introduction to the Theory of Groups*, volume 148 of *Graduate Texts in Mathematics*. Springer, 4th edition, 2012. Standard graduate text on group theory.

[RS09] John Rhodes and Benjamin Steinberg. *The q-theory of Finite Semigroups*. Springer Monographs in Mathematics. Springer, 2009. Comprehensive treatment of Krohn-Rhodes theory and group complexity.

[The20] The mathlib Community. The Lean mathematical library. In *Proceedings of the 9th ACM SIGPLAN International Conference on Certified Programs and Proofs (CPP)*, pages 367–381, 2020. Overview of Mathlib's design and coverage.