**1. Embedding Independence Results into Entailment Cones**

* **Task**: Begin with well-known results like the Paris-Harrington or Goodstein theorems.
* **Practical Implementation**:
  + **Data Collection**: Compile statements of theorems and their independence proofs.
  + **Cone Construction**: Create entailment cones as directed graphs using Python (e.g., NetworkX or Graphviz).
  + **Analysis**: Map proof paths, identifying where axioms like large cardinals "unlock" proofs.
* **Outcome**: Visualize independence results as "bottlenecks" in the cone structure.

**2. Mapping Hierarchies to Cones**

* **Task**: Represent theorems from reverse mathematics (e.g., RCA₀, ACA₀, WKL₀) within entailment cones.
* **Practical Implementation**:
  + **Graph Construction**: Nodes represent subsystems; edges correspond to logical dependencies. Use graph algorithms to identify branching or collapsing.
  + **Validation**: Compare generated hierarchies with established reverse mathematics results.
* **Outcome**: Produce hierarchical diagrams that align with known results and explore new patterns.

**3. Boolean Operations on Axioms**

* **Task**: Investigate how combining or comparing axioms impacts proof paths.
* **Practical Implementation**:
  + **Define Operations**: Implement set-theoretic operations (union, intersection, complement) on axioms within the cone structure.
  + **Visualize Effects**: Examine how operations modify the cone’s topology. Use algebraic or combinatorial tools to test redundancy or complementarity.
* **Outcome**: Discover relationships between axioms, potentially uncovering novel dependencies.

**4. Topological Analysis of Proof Paths**

* **Task**: Use tools from algebraic topology (e.g., homology, Betti numbers) to analyze cone structures.
* **Practical Implementation**:
  + **Enumerate Paths**: Identify loops (cyclic dependencies) and gaps (missing links) in the graph representation.
  + **Apply Homology**: Compute features like connected components, holes, and higher-dimensional voids.
  + **Homotopy Analysis**: Explore equivalence between proof paths and axioms as base points.
* **Outcome**: Provide a novel classification of axiomatic systems based on topological invariants.

**5. Minimal Axiomatic Systems**

* **Task**: Identify the smallest set of axioms necessary to span an entailment cone.
* **Practical Implementation**:
  + **Redundancy Tests**: Systematically remove axioms and observe their impact on proof paths.
  + **Optimization**: Use algorithms to search for minimal spanning sets of axioms.
* **Outcome**: Propose minimal sets of axioms tailored to specific mathematical domains.

**6. Independence Gaps in Cones**

* **Task**: Study gaps in the entailment cone corresponding to unresolved or independent propositions.
* **Practical Implementation**:
  + **Gap Detection**: Use graph traversal methods to locate stalled paths.
  + **Bridging Gaps**: Introduce stronger axioms (e.g., large cardinals) and observe their effect.
* **Outcome**: Develop a systematic approach to mapping and bridging independence gaps.

**7. Visualizing Axiomatic Structures**

* **Task**: Enhance clarity through graphical and 3D visualizations.
* **Practical Implementation**:
  + **Graph Representation**: Build 2D DAGs (directed acyclic graphs) with tools like Gephi or Python.
  + **3D Visualization**: Use advanced tools like Blender or Wolfram Mathematica for 3D entailment cones.
  + **Highlight Features**: Focus on loops, gaps, and connectivity in visual outputs.
* **Outcome**: Generate intuitive, shareable visuals to communicate findings effectively.

**8. Philosophical Exploration**

* **Task**: Reflect on the implications of your findings for the philosophy of mathematics.
* **Key Questions**:
  + Which axioms recur across cones, and what does this suggest about their "naturalness"?
  + How does the finite-infinite dichotomy influence the structure and interpretation of entailment cones?
* **Outcome**: Develop a philosophical narrative to contextualize your research.

**Implementation Strategy**

1. **Select Focus Areas**: Choose 2–3 key tasks to develop initially, balancing theoretical depth with computational feasibility.
2. **Set Milestones**:
   * **Phase 1**: Build foundational structures (e.g., basic entailment cones, graph models).
   * **Phase 2**: Implement advanced analyses (e.g., Boolean operations, homological features).
   * **Phase 3**: Explore philosophical insights and applications.
3. **Iterate and Refine**: Use insights from each phase to improve and expand your research.

Let me know which area you'd like to begin with, and I can guide you through specific steps, including code samples, theoretical details, or conceptual frameworks.