LeanDojo Paper Summary and Next steps

Overview of LeanDojo

LeanDojo is an open-source platform that combines:

- The **Lean** interactive theorem prover (ITP)
- A large dataset of Lean proofs and intermediate proof states
- Tools to integrate **LLMs** into theorem proving

Its main contributions are:

1. **LeanDojo Environment** – an API to interact with Lean proofs programmatically.
2. **Theorem–Proof Dataset** – containing 96k Lean theorems and 8.1M proof states.
3. **ReProver** – an LLM-based prover that uses retrieval to guide proof search.
4. **Evaluation Benchmark** – for consistent comparison of proof generation systems.

Key Components

LeanDojo Environment

- Lets you start from a **proof goal** (state) and interact with Lean to:
  - Query the current hypotheses
  - Apply tactics
  - Get next proof state
- Supports fine-grained inspection for AI agents.

Dataset

- Proofs are **tokenized into intermediate states**, useful for supervised training.
- Stores **context** (imports, definitions, lemmas) so models can retrieve relevant prior work.

ReProver

- **Retrieval-Augmented Generator** for theorem proving.
- Works like **RAG** (Retrieval-Augmented Generation) in NLP:
  - Retrieves **relevant lemmas** from the Lean corpus using **Dense Passage Retrieval (DPR)**.
  - Conditions an LLM on both the goal and retrieved lemmas.

Architecture:

1. **Retriever**: DPR retrieves k most relevant lemmas.
2. **Generator**: LLM proposes next tactics using retrieved context.
3. **Lean Environment**: Checks tactics and returns the new state.
4. Loop until proof is completed or budget is exhausted.

RAG vs. ReProver

- **RAG** in NLP: Retrieve external documents → generate text using them.
- **ReProver**: Retrieve relevant lemmas → generate proof steps using them.
- In both, retrieval **grounds** generation in relevant facts.

Dense Passage Retrieval (DPR)

- An embedding-based retrieval method:
  - Encodes queries (proof goals) and documents (lemmas) into the same vector space using a dual encoder (BERT-like).
  - Uses inner product or cosine similarity to find the closest matches.
- **Advantage:** Faster and more semantically aware than keyword search.

Potential integrations to our project

1. **Rule Retrieval** (like lemma retrieval):
   - Store all IPO regulatory clauses as structured Lean predicates.
   - Retrieve only the relevant rules for a given case (Issuer data).
2. **Guided Compliance Proof**:

    - Instead of proving a theorem, the system “proves” compliance by chaining relevant rule checks.

3. **DPR for Regulation Matching**:
    - Convert rules and issuer facts into embeddings.
    - Retrieve relevant rules automatically for a given issuer profile.
4. **RAG for Explanations**:
    - Retrieve the text of relevant regulations and generate a human-readable compliance report.

Workflow

1. **Phase 1** – Create a LeanDojo-like environment for compliance:
    - Dataset of rules (as Lean predicates) + past compliance cases.
2. **Phase 2** – Add Retriever:
    - Use DPR to match issuer facts to relevant rules.
3. **Phase 3** – Build Compliance Prover:
    - Like ReProver, but instead of a theorem, the goal is “Issuer is eligible”.
4. **Phase 4** – Explanations:
    - Provide human-readable reasons for pass/fail, citing original regulation text.

LeanDojo / ReProver (Theorem Proving)	**Guided Compliance Proof (Regulatory Checking)**
Theorem statement: goal to prove (e.g., $\forall n, \gcd\ n\ n = n$).	**Eligibility statement**: goal to verify (e.g., *Issuer satisfies all IPO eligibility rules*).
Lemma/premise retrieval (DPR): retrieve a small set of useful lemmas from mathlib based on the current proof state.	**Rule retrieval (DPR/heuristics)**: fetch the most relevant regulatory clauses given issuer facts (industry, capital structure, track record).
Guided proof search: tactics operate on the current proof state using retrieved lemmas; search space is pruned.	**Guided rule application**: evaluate only applicable/high-impact rules first; short-circuit on disqualifiers to avoid wasted checks.
Proof state updates: applying a lemma rewrites/simplifies goals; generates subgoals until solved.	**Compliance state updates**: applying a rule sets pass/fail and may spawn dependent checks (e.g., exemptions, carve-outs).
Proof completion: all subgoals discharged ⇒ theorem proved.	**Compliance proof**: all relevant rules satisfied ⇒ issuer eligible; otherwise surface failing clauses.
Traceability: sequence of lemmas/tactics provides an auditable proof trace.	**Explainability**: list of rules checked, outcomes, and reasons provides auditor-friendly evidence.