Consult the Scholar

The Role of Artificial Intelligence in Theorem Proving

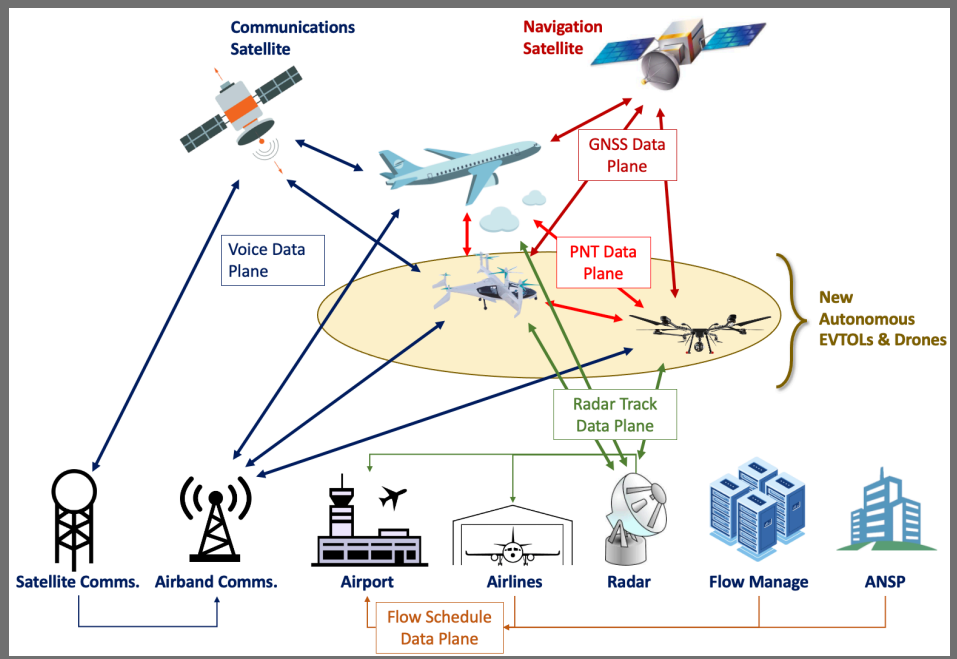
Charles Averill

Software Languages Security Lab
The University of Texas at Dallas
Dartmouth College

May 18th, 2026

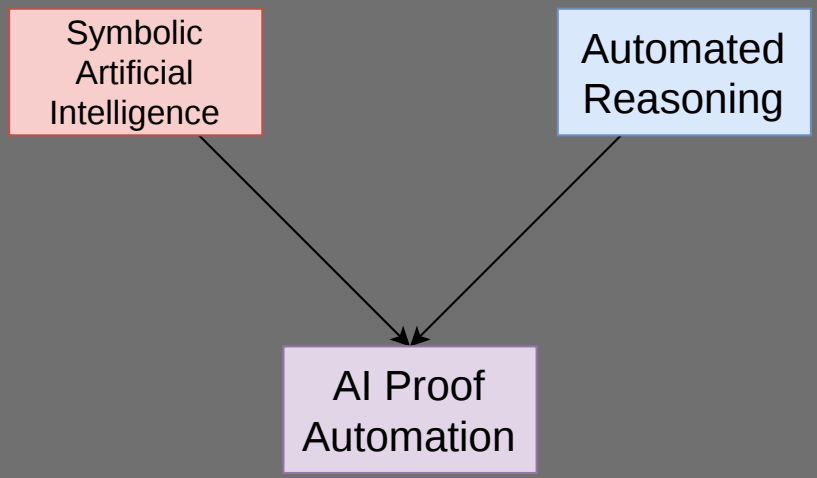
Math is Everywhere

- All modern infrastructure uses advanced math
- Arguably impossible without *automation*
  - Predicting the behavior of complex systems? Linear Algebra!
  - Arranging groups of things? Combinatorics!



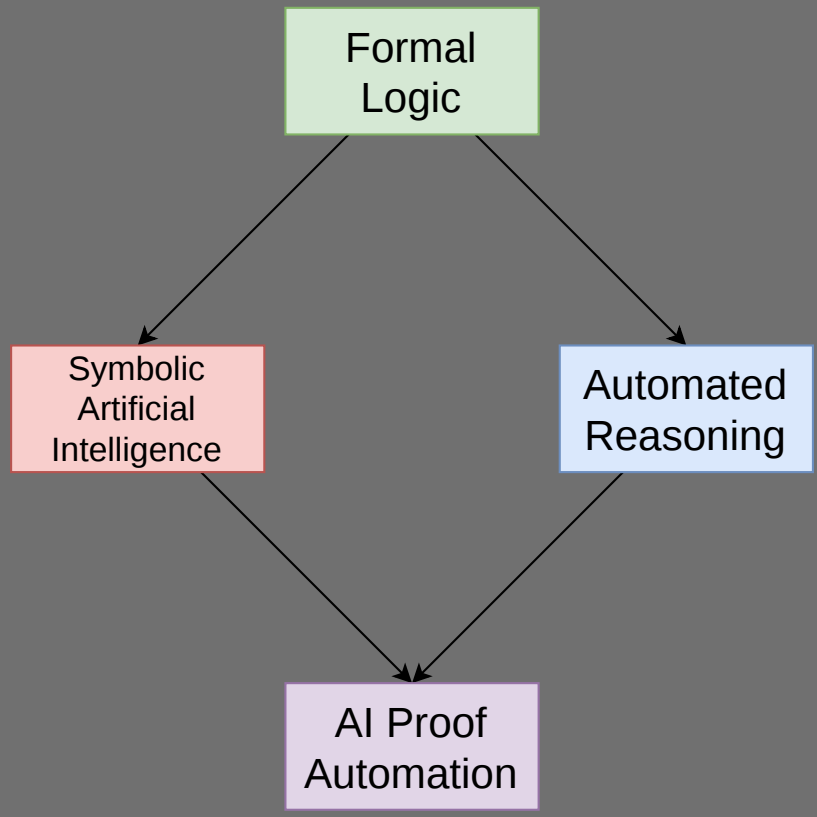
The "Ultimate" Automation

- AI is everywhere
- Useful automation tool because it learns the rules of a system
- No surprise: we're interested in applying AI to modern math



The "Ultimate" Automation

- AI is everywhere
- Useful automation tool because it learns the rules of a system
- No surprise: we're interested in applying AI to modern math
- Yes surprise: AI and mechanized math used to be deeply intertwined!



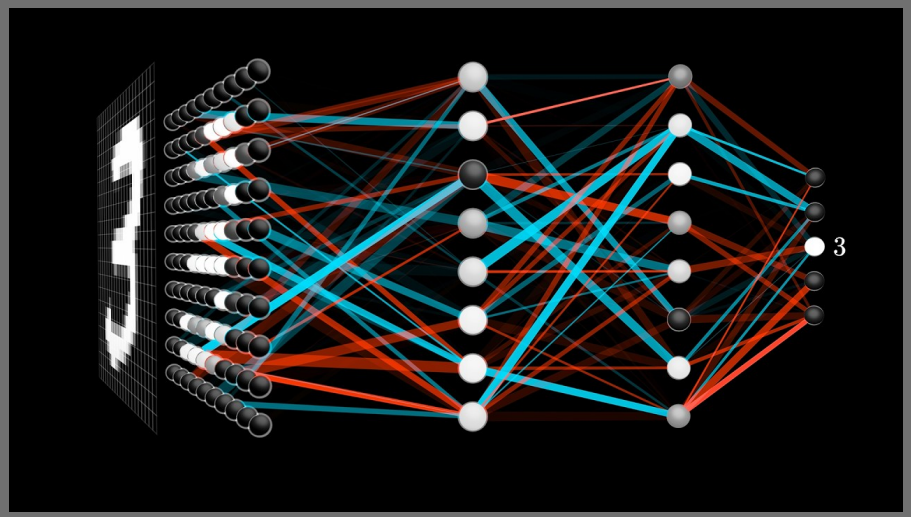
Mechanized Math?

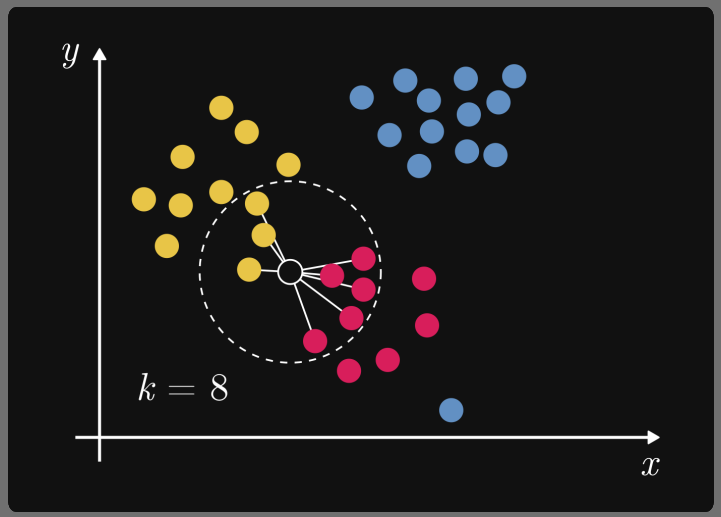
- *Proof assistants*: programs that check the validity of proofs
- Extremely trustworthy by construction
- Started becoming mainstream approx. 10 years ago
- CompCert, sel4, 4 color thm, Kepler conjecture


```
1 goal

  n : nat
  ============================
  n + 0 = n

add_0_r < induction n.
2 goals

  ============================
  0 + 0 = 0

goal 2 is:
 S n + 0 = S n
```

Machine Learning

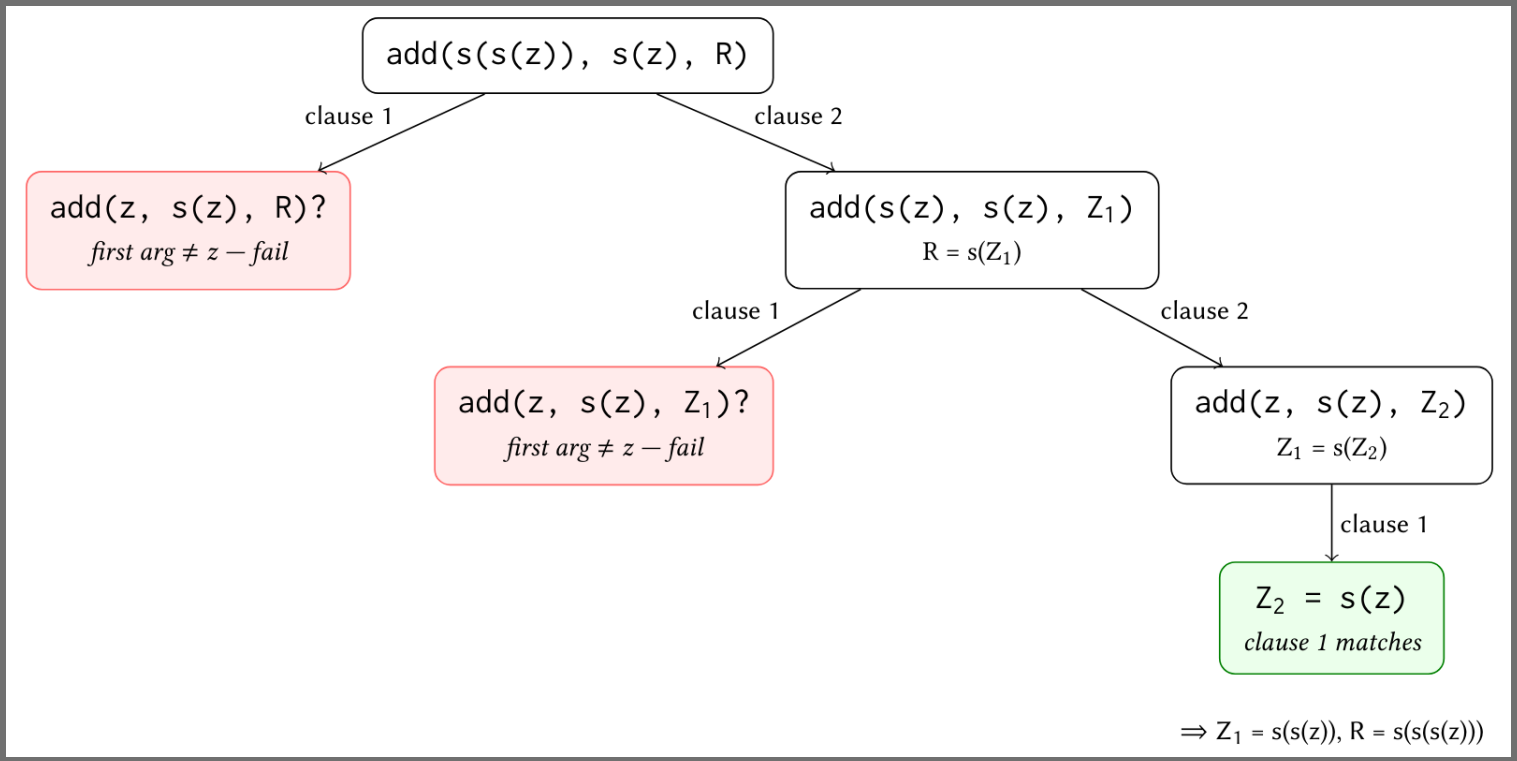
- Class of techniques for learning the rules of a system
- *Neural Network*: groundbreaking generic ML approach capable of learning deep relationships
- NNs can be assembled to fit the need, e.g., *Large Language Models*





Rule-Based Automation

- Encode reasoning strategies in deterministic algorithms
- No training required
- Predictable, suited for specific tasks
- Can be guaranteed to find answers



Proof Automation Tasks

Four primary tasks make up AI proof automation research:

- **Tactic Prediction**: deciding what sequence of reasoning steps to make in a proof
- **Premise Selection**: deciding which known facts are most useful for a specific proof
- **Theory Exploration**: choosing which theorems to prove
- **Autoformalization**: translating natural-language theorems and proofs to proof assistants

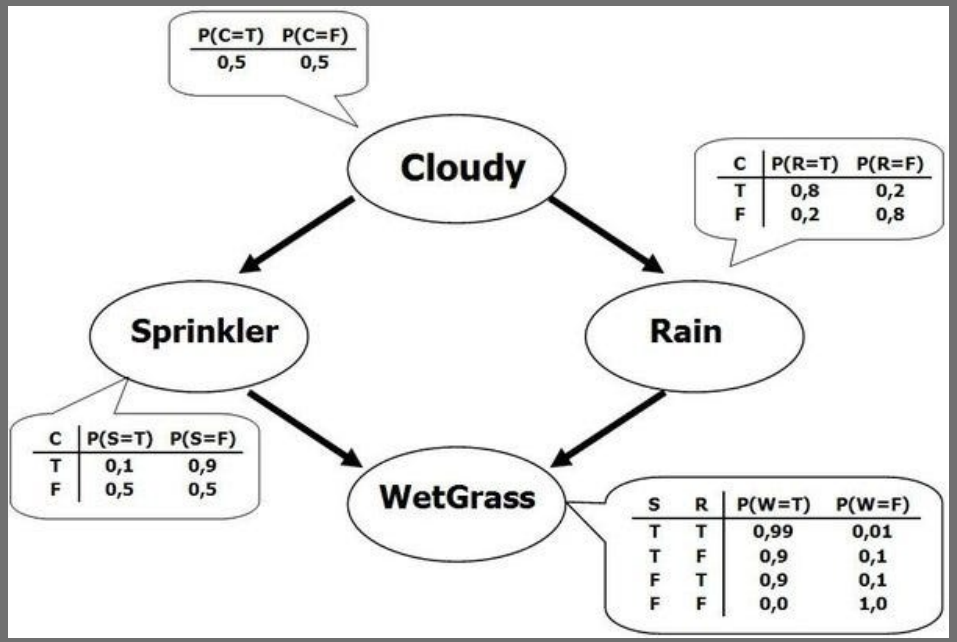
Classical Machine Learning

MaLARea (2007): first case of modern ML for proof automation

- Bayes classifier, takes theorem symbols as input, gives list of premises as output
- Achieved ~50% success on MPTP challenge, ~double that of SOTA

TacticToe (2018): k-NN for premise selection and tactic prediction

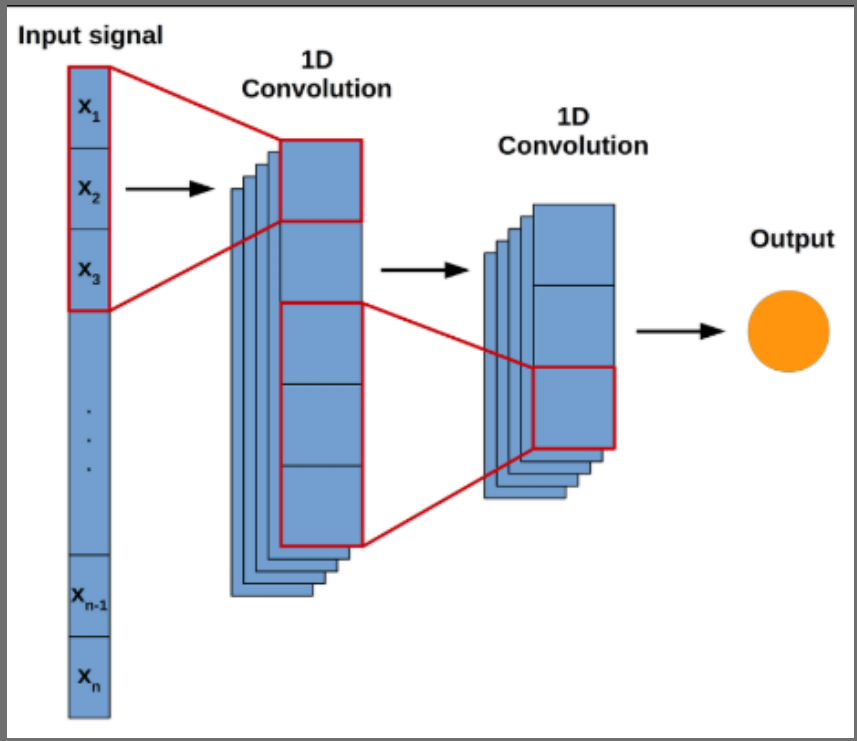
- Proves ~70% of HOL Light stdlib in < 1 min
- Ensemble: MCTS with k-NN heuristics

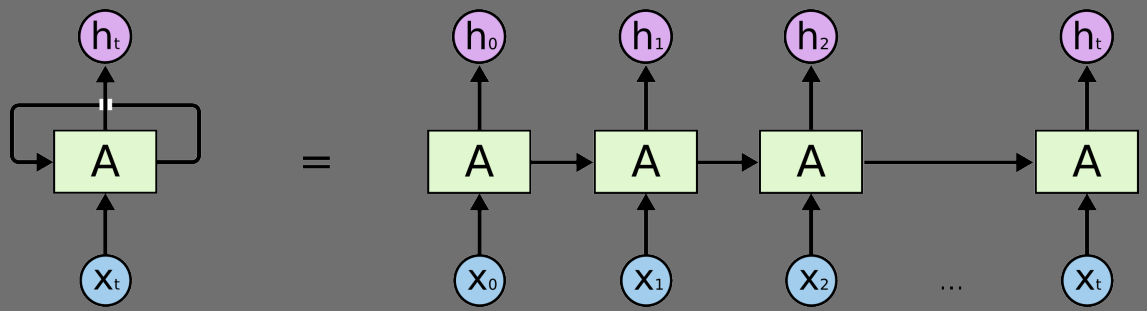


Deep Learning

DeepMath (2016): first case of DL for proof automation

- Evaluated CNNs, LSTMs, GRUs on premise selection for MML
- CNNs perform best (~70% of thms provable by ATPs) - local features most important

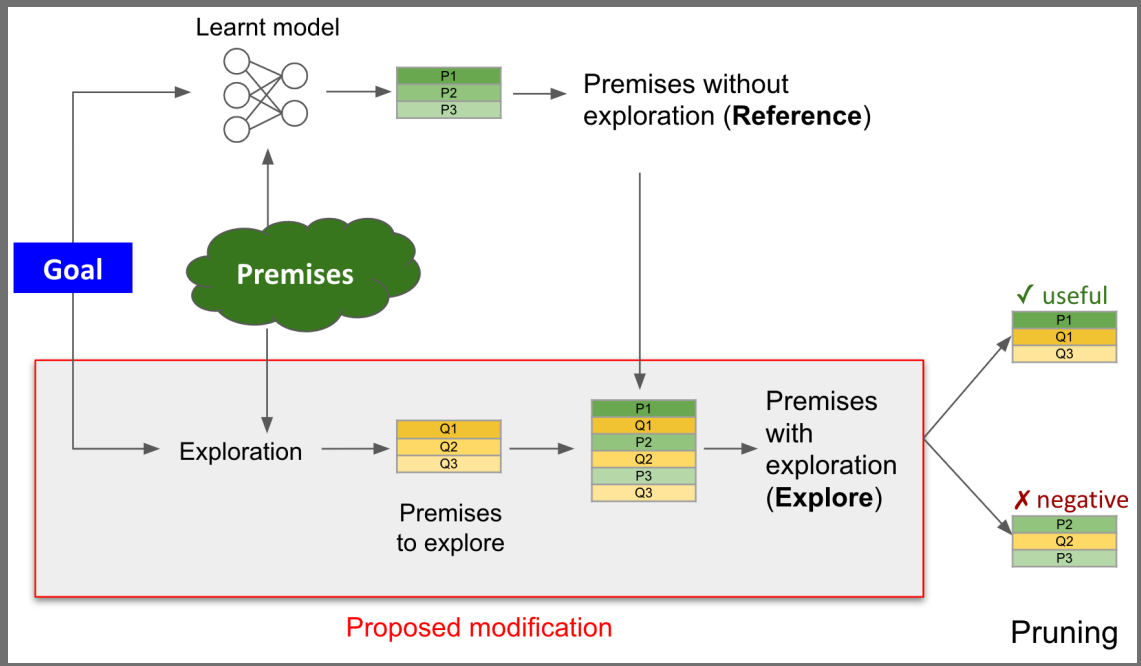




Deep Learning

Learning to Reason in Large Theories Without Imitation (2019)

- Problem: models work best when inputs are close to training data, but they should work on unseen data (hard proofs)
- Idea: *retrieval-augmented* reinforcement learning
- Find relevant premises via similarity metric to supplant model predictions → more RL training data



Large Language Models

GPT-f: first LLM proof automation project

- Takes in *whole proof history* to predict next token
- Found better results by fine-tuning on mathematical text

LeanDojo: Environment for Lean models

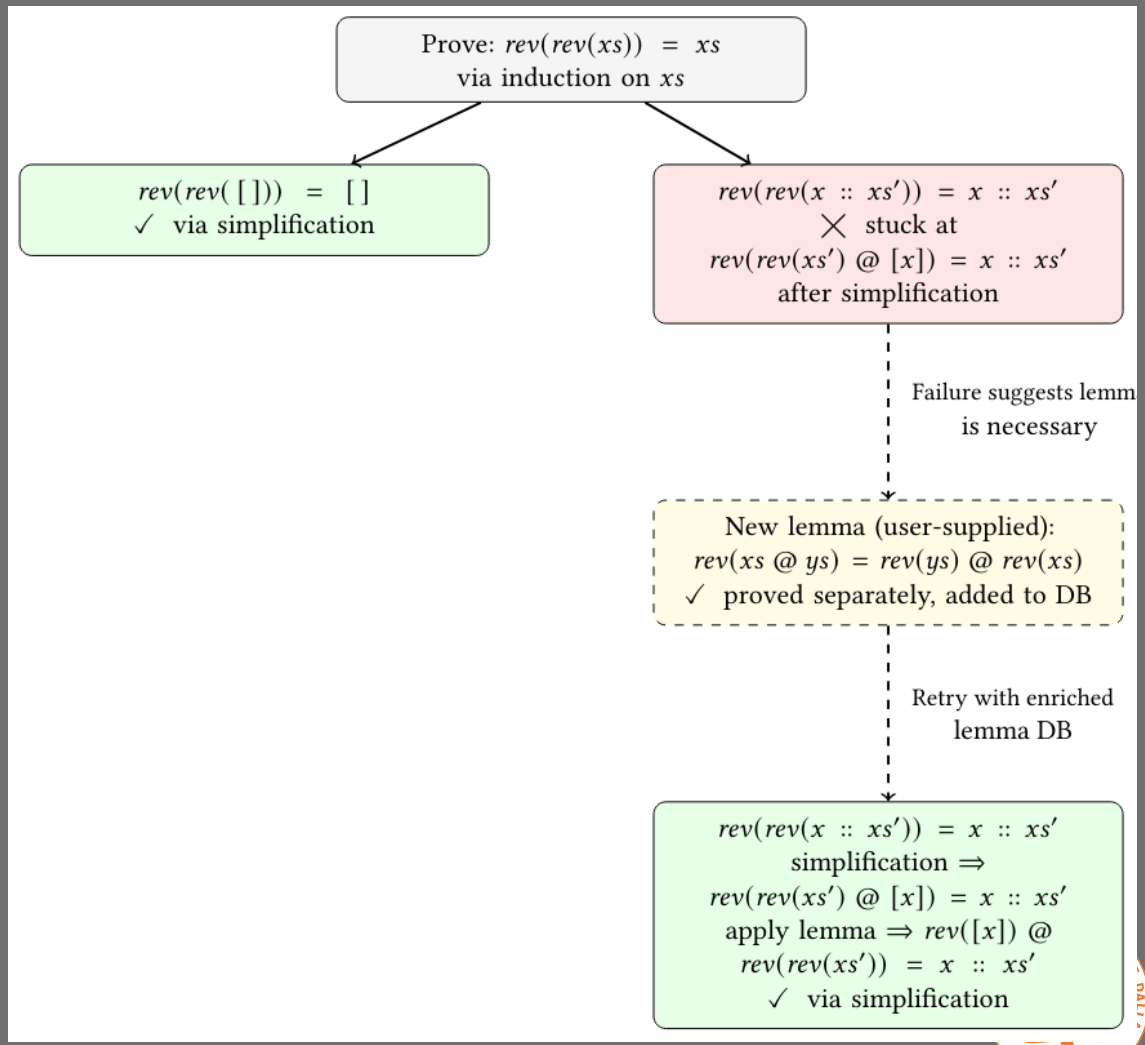
- Identified critical training issue in all modern DL approaches
- Proposes workarounds via artificial diversity measures

Draft, Sketch, Prove

- Problem: translation gap between prompt "prove this theorem" and actual proof script
- Solution: ask LLM to *draft* an informal proof, *sketch* a high-level formal proof, then dispatch to ATP

Honorable Mention - Inductive Automation

- *Nqthm/ACL2*: prominent ATP driven by heuristics
- Heuristics based around simplifying induction/recursion to make proof progress
- *Rippling*: generic inductive automation technique for simplifying inductive goals



Evaluation Metrics

```
Rocq < Search (_ + _).
plus_O_n: forall n : nat, 0 + n = n
plus_n_O: forall n : nat, n = n + 0
plus_n_Sm: forall n m : nat, S (n + m) = n + S m
plus_Sn_m: forall n m : nat, S n + m = S (n + m)
mult_n_Sm: forall n m : nat, n * m + n = n * S m
```

- Notice anything wrong?
- Distinct theorems with nearly-identical statements in train + test sets
- LeanDojo points this out, suggests fix, but incomplete
- What about similar proofs?

Evaluation Metrics

```
Rocq < Search (_ + _).
plus_O_n: forall n : nat, 0 + n = n
plus_n_O: forall n : nat, n = n + 0
plus_n_Sm: forall n m : nat, S (n + m) = n + S m
plus_Sn_m: forall n m : nat, S n + m = S (n + m)
mult_n_Sm: forall n m : nat, n * m + n = n * S m
```

- Notice anything wrong?
- Distinct theorems with nearly-identical statements in train + test sets
- LeanDojo points this out, suggests fix, but incomplete
- What about similar proofs?

Evaluation Metrics

```
Ltac solveh help :=
  intros;
  let IHx' := fresh "IH" in
  multimatch goal with
  | [x : nat |- _] => induction x as [| x' IHx']
  | [b : bool |- _] => destruct b
  | [b : bin |- _] => induction b as [| x' IHx']
  end;
  [simpl; now try rewrite help
  | simpl in *; try easy; try (rewrite help || rewrite <- help); rewrite IHx';
    try (rewrite help); try easy;
    match goal with | [|- ?G] => idtac G end].
Ltac solve := solveh eq_sym.
```

- This (rudimentary) proof procedure proves **70%** of the theorems in one chapter of Software Foundations
- Proofs can be similar too!
- If models memorize proof patterns that can be trivially automated, is that success?

Autoformalization and Theory Exploration

Large-scale autoformalization considered ∼impossible right now

- No way to automate data generation
- No way to automatically evaluate
- Arguably a subjective task

Theory exploration is the opposite of what prediction models do!

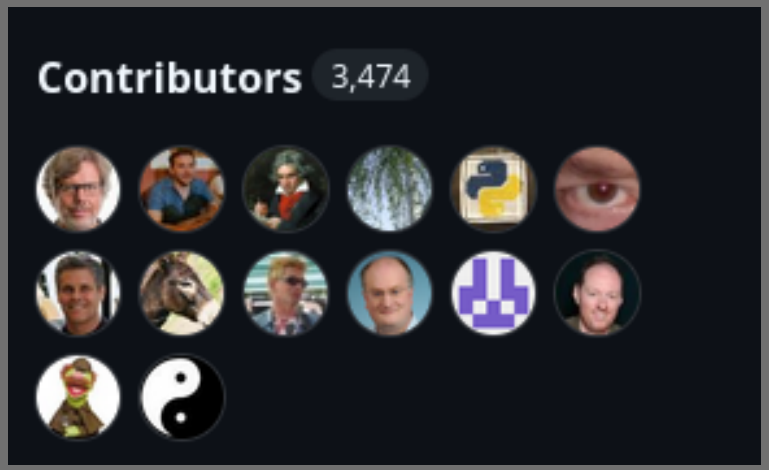
- Standard LLMs will not generate interesting theorems without substantial third-party input
- Maybe RL? Genetic algorithms?

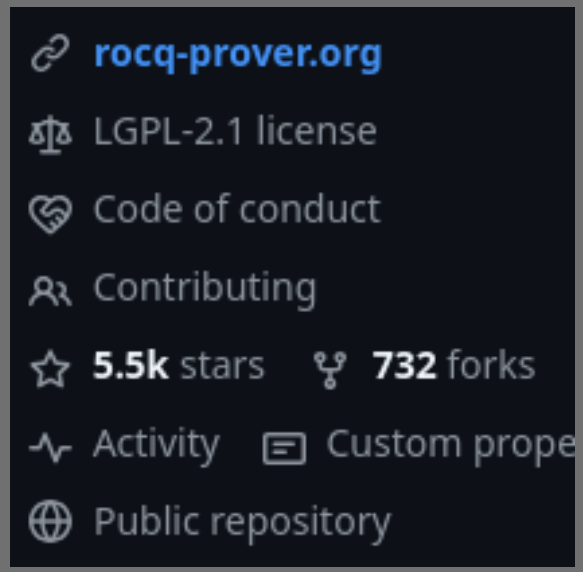
Dataset Generation

- Python code makes up ~20% of all code on GitHub (approx. 84 million repositories)
- There are almost as many Python contributors as there are stars for the Rocq compiler
- Rocq is 7 months older than Python

Dataset Generation

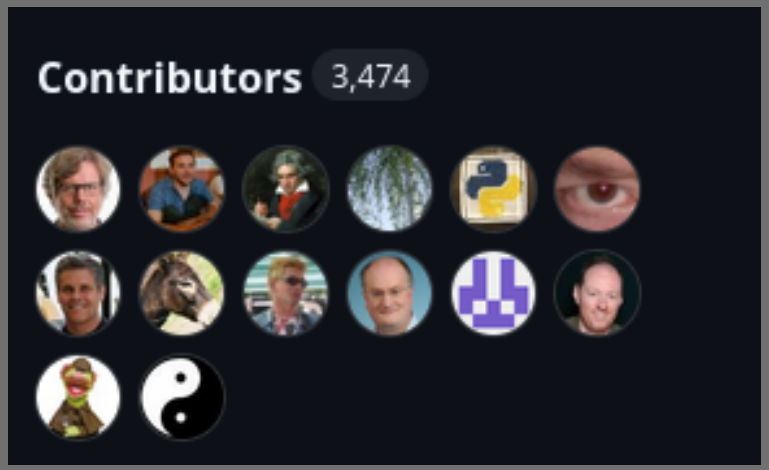
- Python code makes up ~20% of all code on GitHub (approx. 84 million repositories)
- There are almost as many Python contributors as there are stars for the Rocq compiler
- Rocq is 7 months older than Python

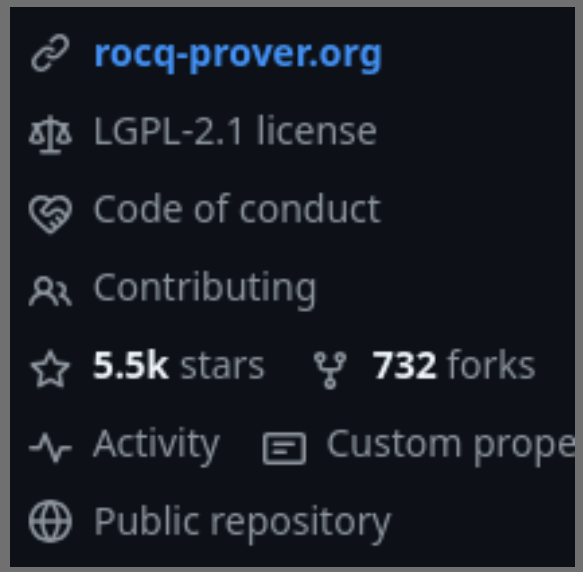




Dataset Generation

- Python code makes up ~20% of all code on GitHub (approx. 84 million repositories)
- There are almost as many Python contributors as there are stars for the Rocq compiler
- Rocq is 7 months older than Python





Thank you!

- Survey: `https://www.charles.systems/writings/qe.pdf`
- Dependency graph: `https://www.charles.systems/qe`