



CAP5602

Hybrid Neuro-Symbolic Theorem Prover

Alex Eduardo Sanchez, Andres Acosta, Brandon Rodriguez, Gianni Martinez, Parker Esco



Problem Statement

Why is theorem proving hard?

- Requires creative reasoning: choosing strategies, identifying lemmas
- Requires formal precision: all steps must be logically correct
- Modern LLMs are good at intuition, but hallucinate
- Proof assistants are perfectly accurate, but not creative

Goal: Build a system combining both strengths.

We address the problem of automating mathematical theorem proving using a hybrid AI system that:

- Takes natural-language math statements
- Converts them into Lean/Coq formal statements
- Uses an LLM to propose proof steps and strategies
- Uses a symbolic proof assistant to validate every step
- Adapts its strategy based on failed attempts



Related Work

LLM-guided theorem proving is a rapidly advancing area that DeepMind, OpenAI, Meta, and other groups are currently publishing.

The aim is to address limitations on human and computer math problem solving capabilities.

Prior similar work:

- DeepMind AlphaGeometry and AlphaProof
- OpenAI GPT-f (Metamath)
- Meta's LeanDojo
- Stanford TacticZero

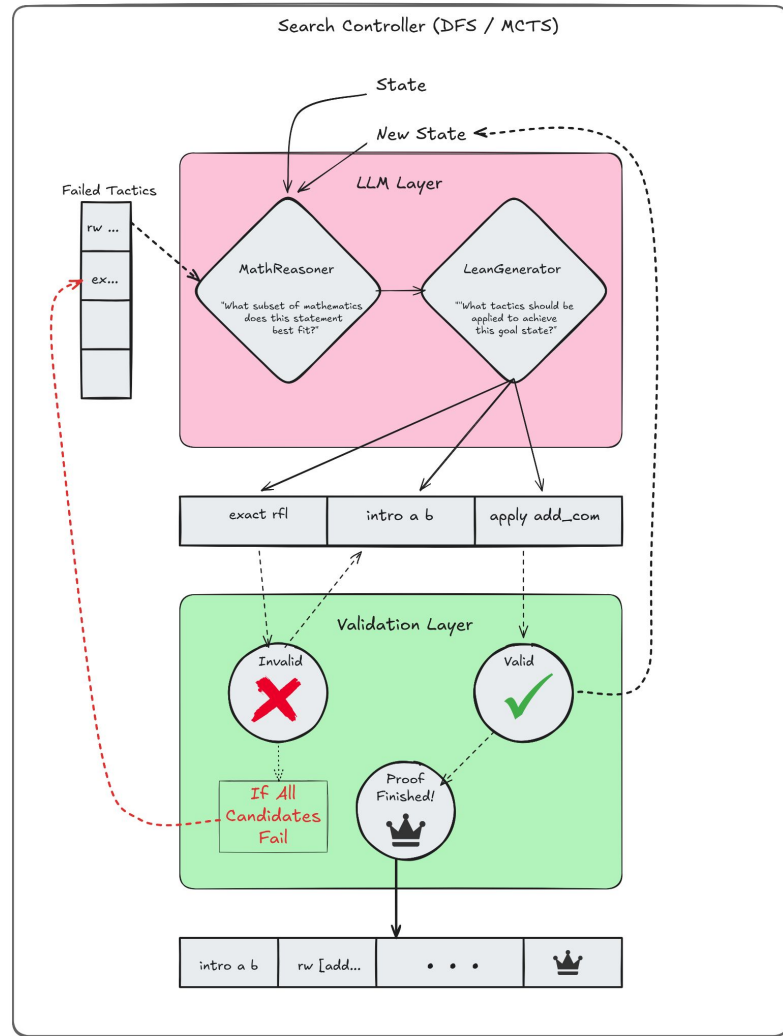
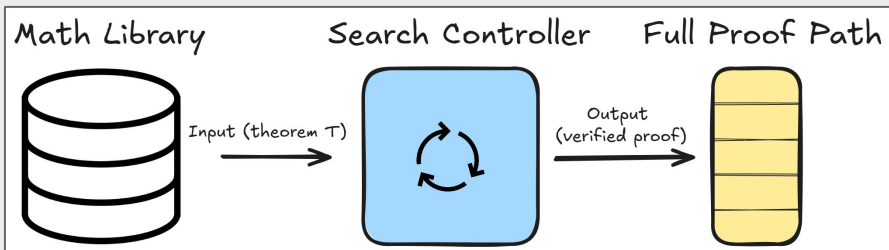


Our Contributions

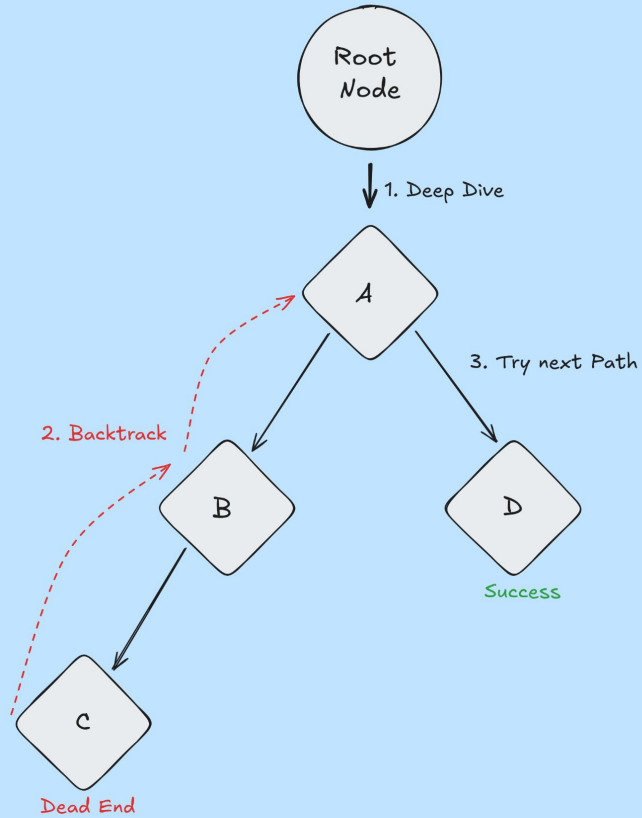
What's new in our project?

- LLM generates strategy, not just raw proof text
- We use adaptive reasoning loops: LLM revises steps based on Lean feedback
- NL to Formal conversion pipeline
- Proof search guided by success/failure signals
- Lightweight prototype for Intro to AI

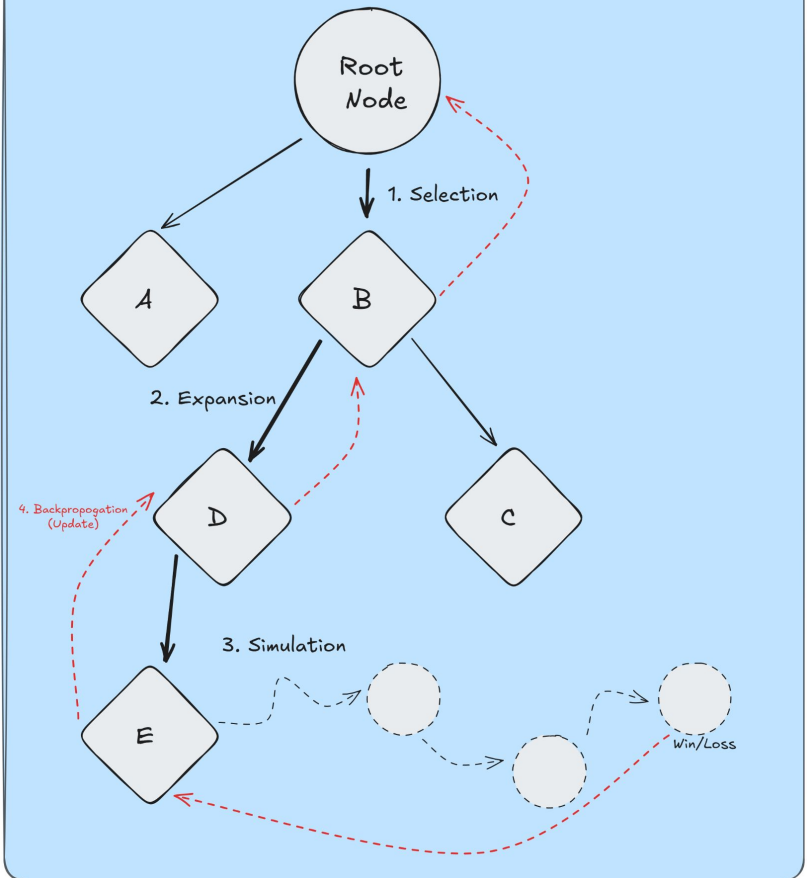
General Architecture



Depth-First Search



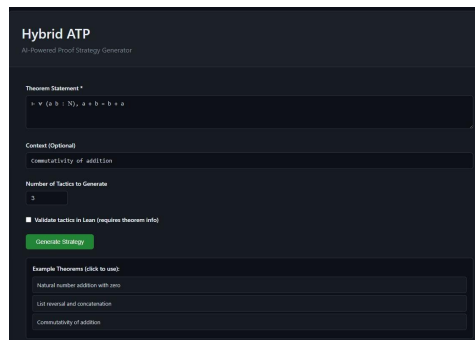
Monte Carlo Tree Search



Architectural and System Design cont.

In order to display the architecture, we designed a front end which mirrors what is seen in the terminal.
Hosted via Flask.

1. Give a Theorem



Hybrid ATP
AI Assisted Proof Strategy Generator

Theorem Statement *

$\forall (a b : \mathbb{N}), a + b = b + a$

Context (Optional)

Commutativity of addition

Number of Tactics to Generate

3

☒ Validate tactics in Lean (requires theorem info)

Generate Strategy

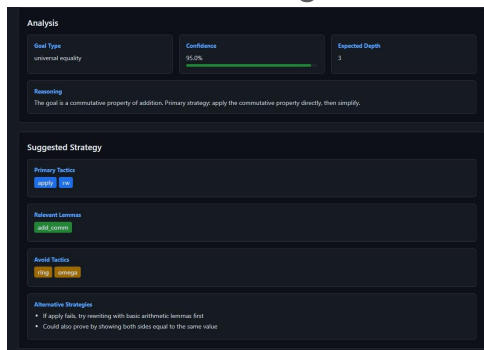
Example Theorems (click to copy)

Natural number addition with zero

List reversal and concatenation

Commutativity of addition

2. LLM-based strategies and analysis



Analysis

Goal Type: universal equality

Confidence: 95.0%

Expected Depth: 3

Reasoning

The goal is a commutative property of addition. Primary strategy: apply the commutative property directly, then simplify.

Suggested Strategy

Primary Tactics

add_comm

Refined Lemmas

add_comm

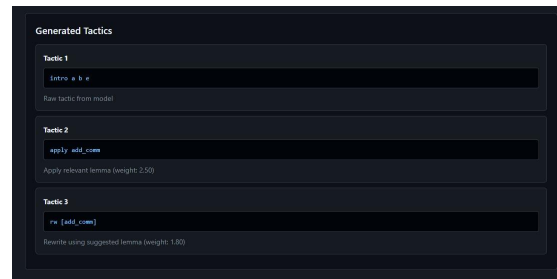
Avoid Tactics

try

Alternative Strategies

- If apply fails, try rewriting with basic arithmetic lemmas first
- Could also prove by showing both sides equal to the same value

3. Generated Lean Code (Tactics)



Generated Tactics

Tactic 1

intro a b a

Raw tactic from model

Tactic 2

apply add_comm

Apply relevant lemma (weight: 2.50)

Tactic 3

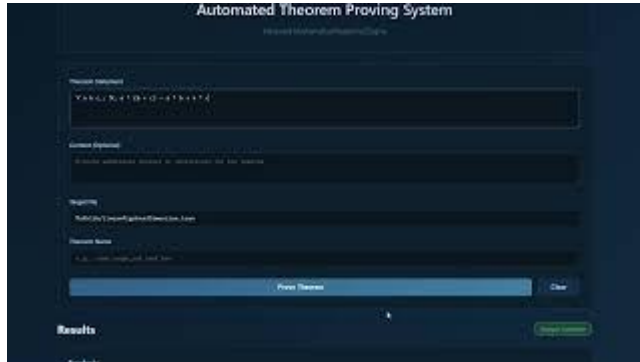
rw [add_comm]

Rewrite using suggested lemma (weight: 1.00)

Demo Video

<https://www.youtube.com/watch?v=OFIABpHIHfg>

- Tested with two theorems (A simple one and a more complex one)





Lessons Learned

- The complexity of compiler integration (particularly the Lean environment)
- Navigating infinitely branching spaces
 - Use pruning and depth bounding
- Resource constraints on local hardware
- Limitations of current math reasoning from LLMs
- Prompt engineering for formal syntax
 - LLMs make symbol predictions, which can violate formal languages like lean
- Importance of feedback loops
 - Proofs require search in addition to generation



Future Work

- Fully implement a MCTS
- Scale to testing from basic arithmetic to more complex domains like Real Analysis
- Implement better LLM models for mathematical reasoning (or fine-tune an open-source model)
- Develop a Fuzzy Cognitive Map (FCM) for causal relationships between mathematical tactics (aiding MathReasoner model)
- Abstract to other proof assistants (Isabelle, Coq, Agda)



Thank You