The Last Mile

How do we make AI theorem provers
which work in the real world for real users
and not just on benchmarks?

Jason Rute
IBM Research / MIT-IBM Watson AI Lab

AI Generated Image (ImageFX)

Disclaimers

- This talk is intentionally provocative
- The opinions are my own, not those of my employer or co-authors
- The ideas are not necessarily novel, and others have said them before
- I could be wrong about the solutions … or even the problems
- I'm one of the worst offenders
- …but I've been thinking about this stuff for a long while!

AI Generated Image (ImageFX)

AI solves 48%+ of HOL/Lean/Isabelle library theorems

On a single CPU, with a time limit of 60 seconds, TacticToe proves 66.4% of the 71 ... brary, whereas E prover with auto-schedule so ... 69.0% by combining the results of TacticToe ar ...

Model	Proved
4-hop GNN, sub-expression sharing *Graph Representations for HOL* Paliwal et al 2019 arxiv	49.95%

Model	Proved
Human Explore *Learning to Reason in Large Theories without Imitation* Bansal et al 2020 arxiv	59.91%

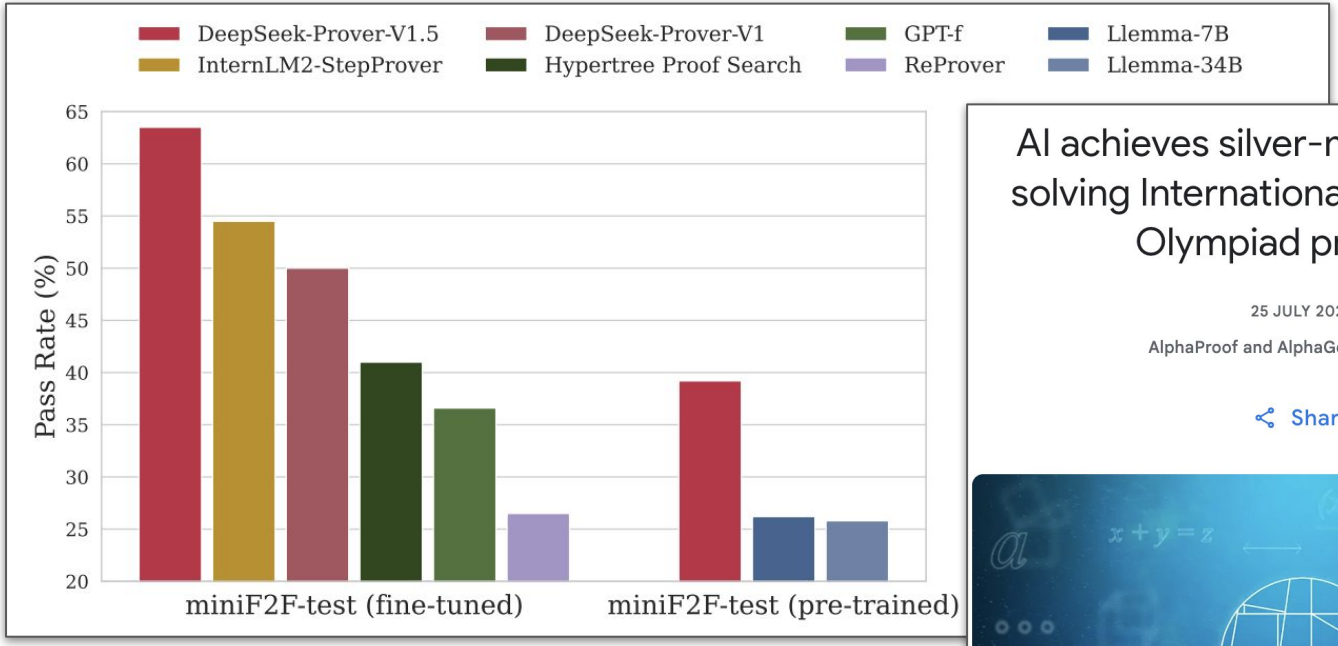
Model	Proved
Zero Explore *Learning to Reason in Large Theories*	56.31%

formalized m ... h a neural theorem prover driven by a T ... PACT improves theorem ... from 32% to 48%.

Method	`random`
`tidy`	23.8
GPT-4	29.0
ReProver (ours)	**51.2**
w/o retrieval	47.6

Method	Proof rate (%)
BM25	30.6
TF-IDF	31.8
OpenAI embed. (Neelakantan et al., 2022)	36.1
Sledgehammer	38.3
Magnushammer (ours)	**59.5**
LISA (Jiang et al., 2021)	33.2
Thor (Jiang et al., 2022a)	57.0
Thor + Magnushammer (ours)	**71.0**

AI can solve competition math problems



AI achieves silver-medal standard solving International Mathematical Olympiad problems

25 JULY 2024

AlphaProof and AlphaGeometry teams

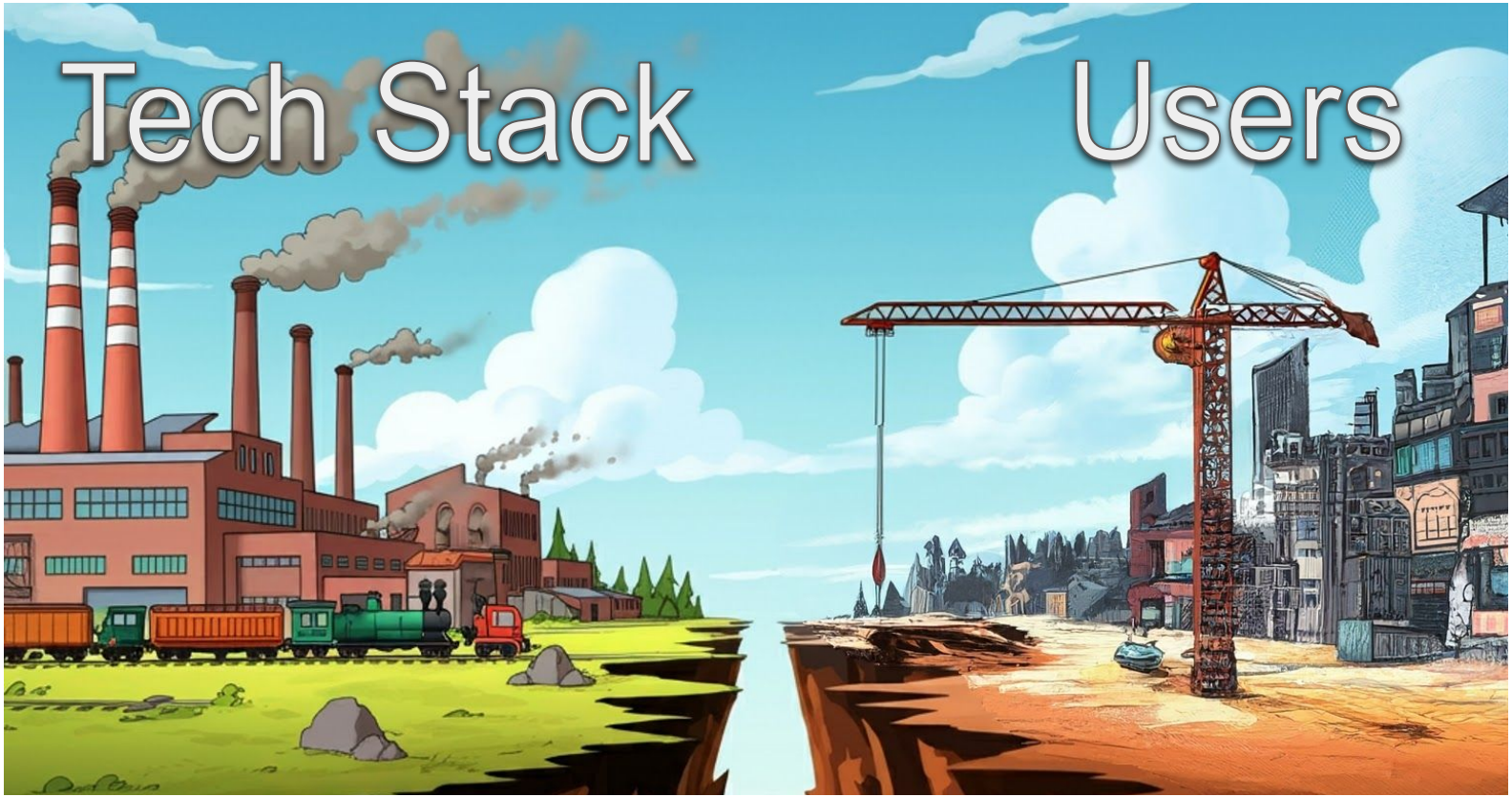
Share

Why don't we have AI assistants for real ITP users?

- Yes we have hammers.
- But why don't we have more?
- Why hasn't the feedback to Lean Copilot, LLMLean, Tactician, CoqPilot, etc been positive?
- Is it because the technology is not good enough yet?

- Claim: The technology is basically good enough, but we aren't building the right sorts of assistants.

The Last Mile



AI Generated Image (ImageFX)

Problems

Parties involved

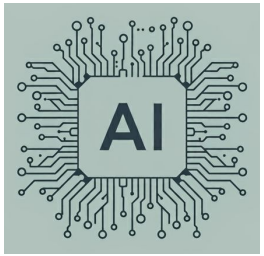
Users

Novice

Expert

ITP Governing Bodies

AI Labs



Industry

Academia

Startups

AI Generated Images (ImageFX)

User Stories: Figuring out what users want

Alex is a new ITP user. He is confused where to start. He is very frustrated that he can't even prove that (xy + yz) / y = x + z. He wants his code editor to suggest next steps, and maybe also a built-in chat bot.

Brenda is an ITP power user. She can prove 10 lemmas before breakfast, but there is so much work to do! She wants tools to make her 10x more efficient. This includes proof automation, but also general purpose coding

Catherine is working on a long term formalization project. She wants tools which will run overnight on a server, closing simple goals, learning from the current code base, and even formalizing parts of her LaTeX blueprint.

Doug is a mathematician exploring new ideas in his field. He is formulating conjectures that he would love to be able to prove or disprove automatically with AI. He is willing to be a guinea pig, partnering with AI experts.

AI Generated Images (ImageFX)

User experience shouldn't take backdoor to benchmarks

Bad UX (Inconvenience > Benefit)

Researchers open sourced their state-of-the-art model and interface code. The installation is a huge pain, so much that most never try it. They provide a model to run locally. It heats up the laptop and barely solves anything in a minute. They also provide a way to use an LLM API, but one can quickly rack up $100 in cost in the first day. No one adopts it.

Good UX (Benefit > Inconvenience)

An ITP ships an AI assistant as part of the standard setup of its system. It is easy to setup with good docs and tech support. It's been designed with user experience in mind. It is far from state-of-the-art, but it provides good, unobtrusive advice when it works. To entice users, there is a free web version to try it out, and the ITP power users and maintainers advertise it.

Benchmarks are not realistic to user experience

- Test problems often are very similar to training problems
  - Lemma's already worked out
  - Training problem was theorem right before the test problem in the library
  - No new definitions or theorems
- Test problems are competition math
  - Competition math can be gamed to some degree
  - Many examples are translated wrong (making them too easy or too hard)
- Few benchmarks on
  - Auto-formalization
  - Program verification
  - Verified code generation
- Data leakage for LLMs
  - Any benchmark from real-world data is already in the training data of every LLM
- Hard to compare methods
  - Different ITPs
  - Different benchmarks
  - Different computational limits

Lack of Communication

- AI researchers and ITP maintainers don't coordinate enough
  - This leads to reinventing the wheel
  - AI systems can't be incorporated into real world systems
  - AI experiments are done on forked version of the ITP
- Startups and ITP maintainers don't coordinate enough
  - Startups just drop stuff
- Symbolic AI and Neural AI researchers don't coordinate enough
  - AI papers consistently beat Hammers on AI paper benchmarks, but no response from Hammer folks
  - No common benchmarks by neural and symbolic AI folks
  - Lean is still investing a lot into building a Hammer
- Almost no one is communicating with users
  - Users should be more involved in making of AI benchmarks
  - User experience should be front and center in design of AI tools

Users don't try tools

- AI tool makes can't get users to try the tools even if they want to
- Many users have a low tolerance for trying experimental tools
- They don't want to
  - try new things
  - sign up for APIs
  - use the command line
  - mess around with lakefiles
  - use WSL in Windows
  - run something overnight to see if it works
  - troubleshoot a bug
- They give up on a tool very quickly

Technical Challenges With Making Real Tools

- How do we keep lemma selection database up-to-date
  - Continual retraining (e.g. in CI)
  - Online updates
- Do I host model locally on machine or in the cloud via an API?
  - API is easier and more powerful, but expensive
  - Local isn't at mercy of developer
- How to make an easy-to-install system?
- How to keep an AI system up-to-date as the library changes?
  - Online learning and/or lemma selection? (Hammer, Tactician, Graph2Tac, Reprover, miniCTX)
  - Continual retraining?
- Incorporate proof search (usually done with tactics) with surrounding code (usually done in the editor)

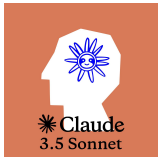
Concrete Proposals for Lean User Community

For Lean users: Keep track of what you want AI to do

- Obvious theorems or goal states that AI tools can't currently solve but should
- Easy auto-formalizations that AI tools can't do

- Be specific and reproducible (MWEs or Git permalinks)
- Ideally, collect these in a central location to turn into benchmarks

For Lean users: Consider trying out Cursor + Sonnet 3.5

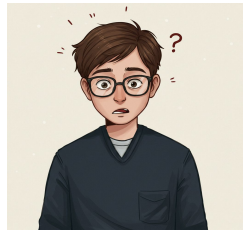


- Pros:
  - Cursor (an IDE) is easy to install (it is a VSCode fork)
  - Sonnet 3.5 (a language model) is a really good code model
  - Sonnet 3.5 has good Lean 4 support
- Cons:
  - Sonnet 3.5 costs money (free first two weeks or so)
  - **Cursor is aggressive in automatically editing code**
    - e.g. it may change a definition earlier in the file
    - it may introduce bugs in your code
    - Alternately: Continue.dev (VSCode plugin) + Sonnet 3.5 for less aggressive experience
  - Privacy: Your code is shared through an API over the internet
  - Cursor (like other IDEs) is a bit aggressive in trying to be the default editor for your system
- Share your wisdom publicly
  - What works? What doesn't?
  - Tips and tricks

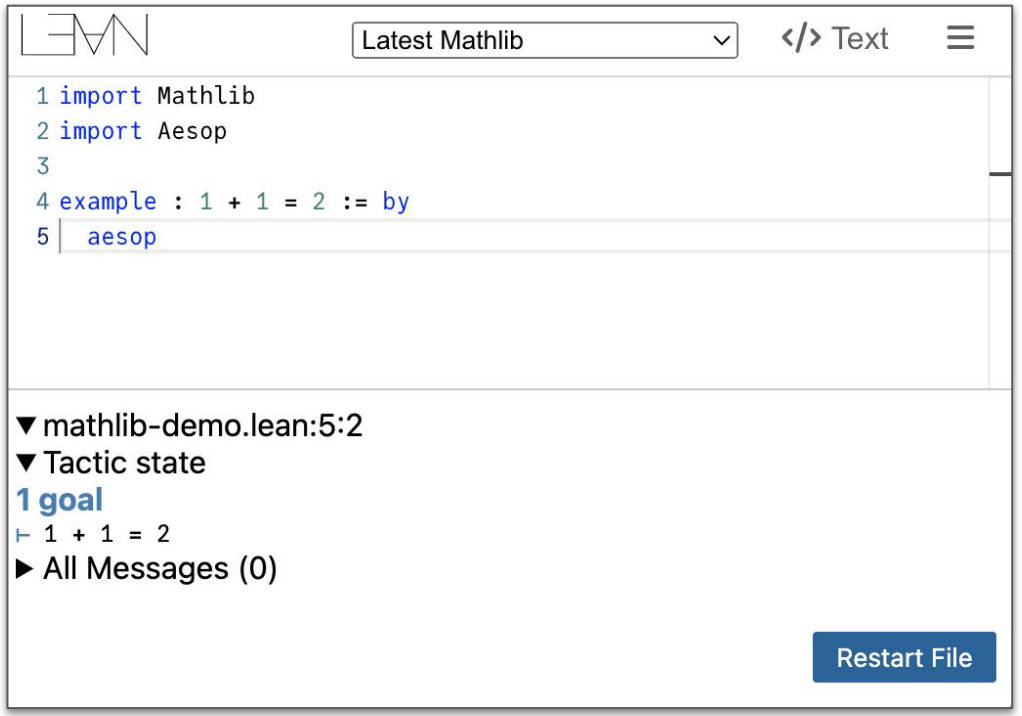
Start Lean AI tools user group

- Group of Lean users who like experimenting with cutting-edge tech
- Willing to install laptop-based Lean tools and give them a go
  - Lean Copilot, LLMLean, LeanAide, Duper, Improver, and others as they come out
- Willing to try tools in other ITPs
  - Coq/Roqc: Tactician, Graph2Tac, Coq Pilot, CoqHammer
  - Isabelle: Sledgehammer, ???
  - HOL4/HOL-Light: ???
- (Optional) If have a GPU, try tools requiring a GPU
  - ReProver, DeepSeek-Prover, InternLM-Prover, etc.
- Give feedback on setup process
- Experiment creatively (e.g. try running them for hours)
- Give feedback on what works and doesn't
- Write getting-started posts on using the tools
- Willing to try a tool if someone asks "are there any tools which can solve X"?

Install AI tactics in an online web editor



- live.lean-lang.org has pre-installed:
  - Lean, Mathlib, Aesop
- Install AI tools also:
  - Lean Copilot, LLMLean, LeanAide
- For AI tool makers:
  - Make your own web editor fork with your tool installed
- Issues:
  - $$$
  - Overuse?
  - LLM API keys



Practical research project: Sorry-filling Server

- Many projects have `sorry`'s in them
- Set up central server (with GPUs) to regularly try to prove `sorry`
- Have different AI and symbolic models compete
  - Lean Copilot, ReProver, DeepSeek-Prover, InternLM-Prover, Duper etc
- Make automatic PR if `sorry` is solve (similar to Lean Agent)
- Important statistics to keep track of:
  - Was sorry solved?
  - How long did a prover take to solve?
  - How much did it cost (especially if using LLM API)?
  - Resource usage? (Memory, threads, GPU utilization, tactics executed, model calls, etc)
- Much easier than have individual user try the tool on his/her laptop
- **Both practical and informative**
- Scientific questions:
  - What are the best models for Lean sorrys?
  - What if we let provers run longer (like hours)?
  - Does this speed up long projects? Does this change user workflow?
  - What problems are some models better suited for
- Practical issues:
  - Who pays? (Grant? Partnership? How much would it really cost?)
  - Ways to keep costs down? (Checkpointing? Bayesian methods to decide what provers to run longer?)
  - `sorry` or new version like `try-sorry`

Things I like

Hammers

- Hammers work in practice!
- Isabelle, Coq, and (soon) Lean have them
- Need engagement between Neural AI and Hammer folks
  - Are the current hammer benchmarks good?
  - What do hammer folks think of neural AI tools?
- What has made hammers so successful that other AI tools can learn from?

Lemma Selection

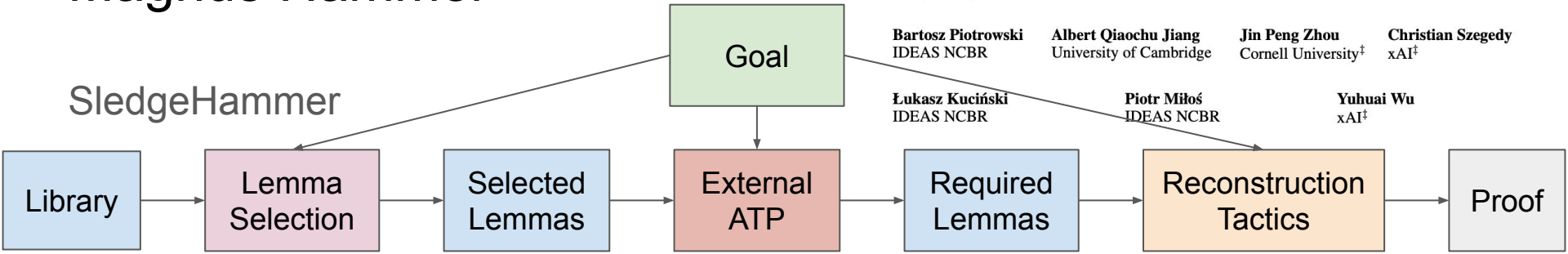
- Universal feature
  - Used in hammers
  - Used in retrieval augmented generation
- ITPs need to solve how to do vector lookup of premises from the library
  - How to make efficient (speed and memory)
  - How small of embeddings can you have
  - How to precompute embeddings during CI
  - How to bulk compute embeddings for a local project
  - How to compute embeddings on the fly for new theorems
  - Should it be done in pure OCaml/C++/Lean or in a neural network?

Magnus Hammer

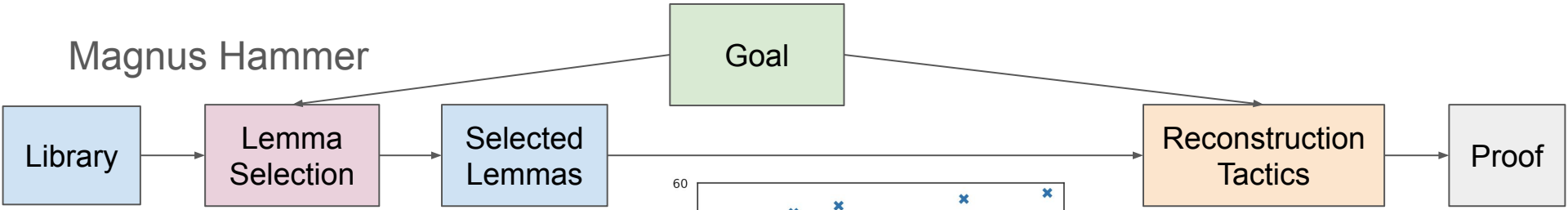
MAGNUSHAMMER: A TRANSFORMER-BASED APPROACH TO PREMISE SELECTION

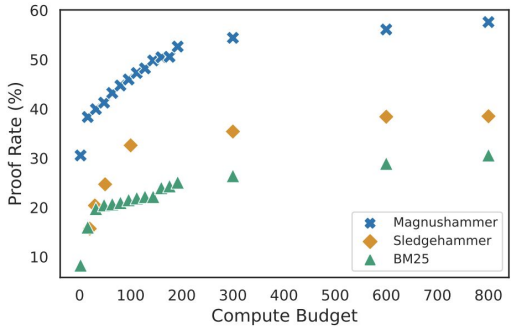
Maciej Mikuła* Google DeepMind†
Szymon Tworkowski* xAI†
Szymon Antoniak* Mistral AI†
Bartosz Piotrowski IDEAS NCBR
Albert Qiaochu Jiang University of Cambridge
Jin Peng Zhou Cornell University‡
Christian Szegedy xAI‡
Łukasz Kuciński IDEAS NCBR
Piotr Miłoś IDEAS NCBR
Yuhuai Wu xAI‡

SledgeHammer



Magnus Hammer





k-NN (TacticToe, HOList, Tactician)

- Record hand-crafted features for each proofstate-goal paper
- Use k-NN to look up closest proofstates
- Copy those tactics (verbatim)
- Tree search with selected tactics
- (Optional) Use online k-NN to take into account recent information
- (Optional) Use locality sensitive hashing forest to search over all tactics

- Works much better than you think!
- Should be a standard baseline in all AI for TP papers!
- Can be written (mostly) as a pure tactic.

Lyra, LeanAide, PALM, Cobblestone, etc

- New class of solvers work by
  - calling a LLM to solve a proof or auto-formalization
  - fixing the errors with existing symbolic automation
  - (optionally) rerunning the LLM to solve the existing

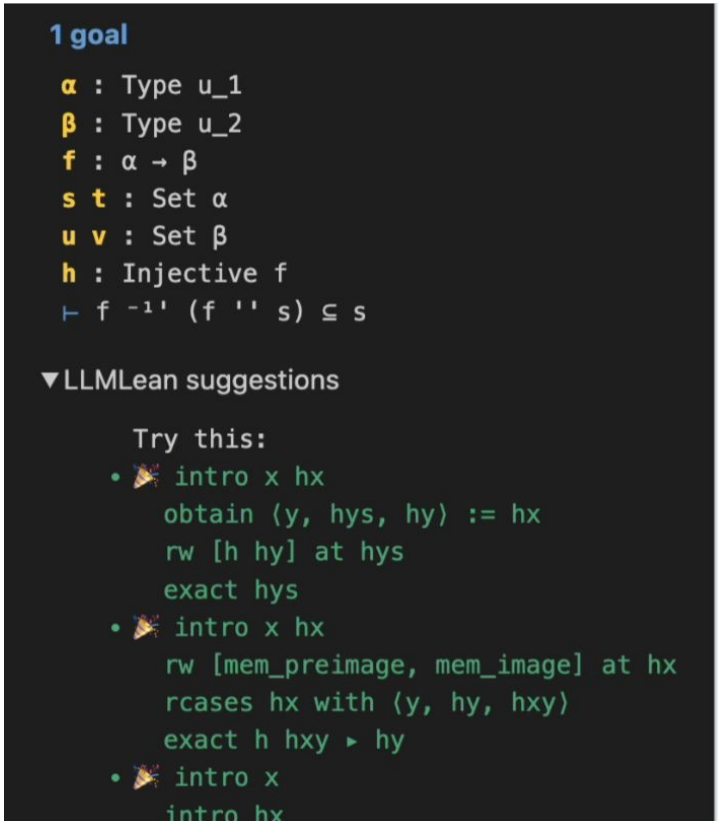
- Practical approach which addresses current limitations:
  - Current LLMs are still error prone (and bad at ITP stuff)
  - Current AI assistants (e.g. Github-Copilot, Cursor) don't have access to goal states
  - LLMs are expensive and should only be used when helpful

Code assistants

- We should test and benchmark general purpose coding assistants like we do ITP specific tools.
- Do they help with day-to-day tasks?
- What are they good and not good at?
- Where do they fail?
- What community advice do we have for better prompting?

Centralized features like "Try this:" in Lean

- Click on tactic to replace tactic call with code
- User-centric feature
- Common tool available to other tactics
- Used in:
  - Standard Mathlib tactics
  - Aesop
  - Lean Copilot
  - LLMLean



Aesop

- Highly customizable search in Lean
- Available for other AI tools
- Used by Lean Copilot
- Easy to hook up your AI model to a Lean proof search

Some things I like: Moogle.ai