Lean:
Past, Present, and Future

Sebastian Ullrich, Lean FRO
FSCD, 2024-07-13
Lean Beginnings

The Strategy Challenge in SMT Solving

Leonardo de Moura\textsuperscript{1} and Grant Olney Passmore\textsuperscript{2,3}
leonardo@microsoft.com, grant.passmore@cl.cam.ac.uk

\textsuperscript{1} Microsoft Research, Redmond
\textsuperscript{2} Clare Hall, University of Cambridge
\textsuperscript{3} LFCS, University of Edinburgh

Abstract. High-performance SMT solvers contain many tightly integrated, hand-crafted heuristic combinations of algorithmic proof methods. While these heuristic combinations tend to be highly tuned for known classes of problems, they may easily perform badly on classes of problems not anticipated by solver developers. This issue is becoming increasingly pressing as SMT solvers begin to gain the attention of practitioners in diverse areas of science and engineering. We present a challenge to the SMT community: to develop methods through which users can exert strategic control over core heuristic aspects of SMT solvers. We present evidence that the adaptation of ideas of strategy prevalent both within the Argonne and LCF theorem proving paradigms can go a long way towards realizing this goal.
Lean Beginnings

Created as a platform for *white-box automation* by Leonardo de Moura

Intended as a frontend-agnostic answer to the Strategy Challenge

A *lean* kernel: minimized type theory compared to similar systems
Lean 0.1 (2014)

An interactive theorem prover

```lean
theorem nat_trans3i {a b c d : Nat} (H1 : a = b) (H2 : c = b) (H3 : c = d) : a = d := trans (trans H1 (symm H2)) H3
```

Valid Lean 0.1 2 3 4 code!

Small set of built-in tactics, more could be added with Lua

```lean
tactic_macro("simp_no_assump", { macro_arg.Ids },
    function (env, ids)
        if ids.size == 0 then ...
```
Lean 2 (2015)

Added support for inductive types, type theory mostly unchanged since

Optional Homotopy Type Theory mode
Lean’s Type Theory

A dependent type theory based on the *Calculus of Inductive Constructions*

Additions for classical mathematics and program verification:

- A definitionally proof-irrelevant universe of propositions $\text{Prop}$
- Quotient types

Adjustments for simplicity and leanness:

- Recursion compiled down to `recursor` higher-order functions
- No universe cumulativity (since Lean 2)
Lean And Mathematicians

Expressive, classical-focused logic

Growing a community and library, first at CMU, then on Zulip

Organization summary

- Number of users: 9,478
- Users active during the last 15 days: 762
- Number of guests: 0
- Total number of messages: 1,452,855
- Number of messages in the last 30 days: 24,687
- File storage in use: 6.3 GB
Start Of My Involvement

leodemoura commented on May 14, 2015

Today, I'm going to cry of happiness :-)  
Thanks a lot for going into the source code and making the necessary modifications!  
They are very welcome!
Lean 3 (2017)

Made Lean a *metaprogramming* language

Removed HoTT support

*Mathlib* spun out as a separate library

```lean
meta def assumption : tactic unit :=
do { ctx ← local_context,
t ← target,
h ← find t ctx,
extact h }
<|> fail "assumption tactic failed"
```
Mathlib
Liquid Tensor Experiment

Nov 2020: Peter Scholze posits formalization challenge

“I spent much of 2019 obsessed with the proof of this theorem, almost getting crazy over it. In the end, we were able to get an argument pinned down on paper, but I think nobody else has dared to look at the details of this, and so I still have some small lingering doubts.”
Liquid Tensor Experiment

Nov 2020: Peter Scholze posits formalization challenge

May 2021: Johan Commelin announces completed Lean formalization of crucial intermediary lemma, with only minor corrections

“[T]his was precisely the kind of oversight I was worried about when I asked for the formal verification. [...] The proof walks a fine line, so if some argument needs constants that are quite a bit different from what I claimed, it might have collapsed.”
Liquid Tensor Experiment

Nov 2020: Peter Scholze posits formalization challenge

May 2021: Johan Commelin announces completed Lean formalization of crucial intermediary lemma, with only minor corrections

July 2022: Completion of the full challenge in Lean
Liquid Tensor Experiment

\textbf{variables} (p' \ p : \mathbb{R}_{\geq 0}) [\text{fact} (0 < p')] [\text{fact} (p' < p)] [\text{fact} (p \leq 1)]

\textbf{theorem} \text{liquid\_tensor\_experiment} (S : \text{Profinite.\{0\}}) (V : \text{pBanach.\{0\}} p) :
\forall \ i > 0, \text{Ext} \ i (\mathcal{M}_{\{p'\}} S) V \equiv 0 :=
Crowd-Sourced Mathematics

“The beauty of the system: you do not have to understand the whole proof of FLT in order to contribute. The blueprint breaks down the proof into many many small lemmas, and if you can formalise a proof of just one of those lemmas then I am eagerly awaiting your pull request.” – Kevin Buzzard on the FLT Project
**Liquid Tensor Experiment**

**Theorem 2.4.15 (Clausen-Scholze)**

Let $0 < p' < p \leq 1$ be real numbers, let $S$ be a profinite set, and let $V$ be a $p$-Banach space. Let $\mathcal{M}_{p'}(S)$ be the space of real $p'$-measures on $S$. Then

$$\text{Ext}^{i}_{\text{Cond}(\text{Ab})}(\mathcal{M}_{p'}(S), V) = 0$$

for $i \geq 1$.

**Proof**

Recall from Lemma 2.4.14 the short exact sequence

$$0 \rightarrow \mathcal{L}_{p'}(S) \rightarrow \mathcal{L}_{r'}(S) \rightarrow \mathcal{M}_{p'}(S) \rightarrow 0.$$ 

Apply to this $\text{Ext}^{*}(\_ , V)$ to obtain a long exact sequence. Note that $T$ acts on $V$ via multiplication by $\frac{1}{2}$ (by Lemma 2.1.2). Hence we can use Lemma 2.4.13 to obtain isomorphisms between the Ext-groups involving $\mathcal{L}_{r'}(S)$, for $i > 0$, and a surjection for $i = 0$. The result follows. 

$\square$
Only The Beginning

**Sphere Eversion**, Massot, Nash, and van Doorn, 2020-2022

**Fermat’s Last Theorem for regular primes**, Brasca et al., 2021-2023

**Unit Fractions**, Bloom and Mehta, 2022

**Consistency of Quine's New Foundations**, Wilshaw and Dillies, 2022-2024

**Polynomial Freiman-Ruzsa Conjecture**, Tao and Dillies, 2023

**Prime Number Theorem And Beyond**, Kontorovich and Tao, 2024-ongoing

**Carleson Project**, van Doorn, 2024-ongoing

**Fermat’s Last Theorem**, Buzzard, 2024-ongoing
Lean 4 (2023)

Current version of Lean

Made Lean a *general-purpose* programming language

Implemented in 120+ kLoC of Lean!

Opened up parser and elaborator for complex notations, embedded languages, …

```
#doc (Post) "Functional induction" =>

%%
authors := ["Joachim Breitner"]
date := (2024, 5, 17)
categories := [technical]
%%
```

Proving properties of recursive function:
Mathlib 4
Software Verification in Lean 4

Formalizing Cedar in Lean: QED

<table>
<thead>
<tr>
<th>Models</th>
<th>Lean LOC</th>
<th>Dafny LOC</th>
<th>L/D</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data</td>
<td>246</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Spec</td>
<td>951</td>
<td>1707</td>
<td>.56</td>
</tr>
<tr>
<td>Validator</td>
<td>532</td>
<td>1189</td>
<td>.45</td>
</tr>
<tr>
<td>Total</td>
<td>1729</td>
<td>2896</td>
<td>.60</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Verification</th>
<th>Lean (s)</th>
<th>Dafny (s)</th>
<th>L/D</th>
</tr>
</thead>
<tbody>
<tr>
<td>All proofs</td>
<td>185</td>
<td>519</td>
<td>.36</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Testing (per request)</th>
<th>Rust (μs)</th>
<th>Lean (μs)</th>
<th>Dafny Java (μs)</th>
</tr>
</thead>
<tbody>
<tr>
<td>abac</td>
<td>7</td>
<td>4</td>
<td>3325</td>
</tr>
<tr>
<td>abac-typed</td>
<td>7</td>
<td>5</td>
<td>3410</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Proofs</th>
<th>Lean LOC</th>
<th>Dafny LOC</th>
<th>L/D</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data</td>
<td>681</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Authz</td>
<td>350</td>
<td>394</td>
<td>.89</td>
</tr>
<tr>
<td>Validator</td>
<td>4686</td>
<td>3110</td>
<td>1.5</td>
</tr>
<tr>
<td>Total</td>
<td>5717</td>
<td>3504</td>
<td>1.6</td>
</tr>
</tbody>
</table>

**SampCert**

A verified implementation using Lean and Mathlib of the discrete Gaussian sampler for differential privacy, the composition and postprocessing of zero concentrated differential privacy, and some simple queries.
AI in Lean

OpenAI: Solving (some) formal math olympiad problems

Meta AI: Teaching AI advanced mathematical reasoning

DeepMind: Scalable AI Safety via Doubly-Efficient Debate

LeanDojo: open source models, datasets, and code for Lean

Morph labs is developing ML models for Lean and moogle.ai
Focused Research Organization (FRO)

A new type of nonprofit startup for science developed by Convergent Research
The Lean FRO

A non-profit organization dedicated to the development of Lean

Missions:

- Address **scalability**, **usability**, and **proof automation** in Lean.
- Support formal mathematics.
- Achieve self-sustainability in 5 years.

Supported by Simons Foundation International, Alfred P. Sloan Foundation, and Richard Merkin

[lean-fro.org](http://lean-fro.org)
The Lean FRO

Leo de Mourn (AWS)  
Chief Architect, Co-Founder

Sebastian Ullrich  
Head of Engineering, Co-Founder

Corinna Calhoun  
Chief Operating Officer

Henrik Böving  
Research Software Engineer

Joachim Breitner  
Senior Research Software Engineer

David Thrane Christiansen  
Senior Research Software Engineer

Johan Commelin  
Mathematical Research Engineer

Markus Himmel  
Research Software Engineer

Marc Huisinga  
Research Software Engineer

Mac Malone  
Research Software Engineer

Kyle Miller  
Research Software Engineer

Kim Morrison  
Senior Research Software Engineer

Sofia Rodrigues  
Research Software Engineer
The Lean FRO: Year 1

**Documentation:** Verso authoring tool, [lean-lang.org/blog/](https://lean-lang.org/blog/)

**System:** incremental proof processing

**Packaging:** Reservoir package registry

**Automation:** functional induction, *omega*

**Library:** verified hash map, bitvector

**Infrastructure:** Mathlib cache hosting, continuous benchmarking

… and many other improvements across the system
The Lean FRO: Roadmap

Documentation: reference manual, metaprogramming guide

System: parallel processing, module system

Packaging/Infrastructure: cloud build & cache

Automation: SMT primitives, and more

Library: more data structures, programming fundamentals
Conclusion

10 years of Lean, a system that grew with the ambitions of its users

Introductory reading:

- [Functional Programming in Lean](https://lean-lang.org/book)
- [Theorem Proving in Lean 4](https://lean-lang.org/book)
- [Mathematics in Lean](https://lean-lang.org/book)

[lean-lang.org](https://lean-lang.org)