Seminar Lean

Starting our project, we embarked on our journey with the Natural Number Game (NNG) to learn about the syntax of Lean 3. We also explored the beta version of the NNG for Lean 4 to compare different versions and we delved into the German Lean game "Formaloversum" to gain new insights.

Additionally, we read into the Lean manual and got curious about the benefits of type theory in theorem proving. Motivated by the foundations of the NNG, we made the decision to create our own mini tutorial world in Lean 4. With some effort, we successfully developed the tutorial and launched the game on a self-hosted web server.

1. What is a Theorem Prover?

Def. (Theorem Prover):

A theorem prover is a software tool or system that verifies the correctness of a mathematical proof. As such, the theorem prover ensures to be reliable and accurate in the mathematical reasoning.
The goal of a theorem prover is to validate a proof based on a set of axioms, rules of inference, and logical deduction.

2. Why do we use Theorem Provers?

A mathematical proof has two roles to fill: first, it convinces the reader that the statement is correct, and second, it explains why the proof is correct and the underlying ideas.
A mathematical proof is constructed in small steps that can be verified easily. Mathematicians are in this regard sometimes lazy and skip proof steps because those are "definitely" true. But sometimes theorems are false, such that a "trivial" small step couldn't be verified.
With that in mind, it can be even worse because when a wrong statement of a theorem is used in another proof, it is like pulling at the bottom card of a card house; it collapses.
Getting back to the two roles of a proof, a theorem prover can not help us explain the proof or even explain the underlying ideas, but a theorem prover is definitely able to help us verify the correctness of a proof by trying to check mechanically that the theorem is true, and by doing so, we transform a previous informal proof into a formal proof.
That is not an easy task because a proof in a textbook is not something a machine can understand, so a translation from an informal proof, which is written in a natural language, is a tedious job to do.

3. Small History of Theorem Provers

The earliest work on computer-assisted proof was in the 1950s. The first theorem provers were automated theorem provers (ATP), which are theorem provers that were supposed to prove an assertion fully automatically. Later, the idea came up for a more interactive arrangement where the user guided the proof. In order to guide a machine proof, there needs to be a language for the user to communicate the proof to the machine. Today, there are many theorem provers out there; some names you can recognize are Coq, HOL, Isabelle, etc. Some cornerstones for interactive theorem proving are Automath, LCF, Minzar and Martin-löf type theory.

4. What is Lean?

Lean is a functional programming language and an interactive theorem prover.
Lean was developed at Microsoft Research in 2013. The idea behind Lean was to help mathematicians and engineers solve complex mathematical problems. Lean has a small, trusted kernel based on dependent type theory. Lean aims to bridge the gap between automated and interactive theorem proving by allowing automated tools in an interactive framework. It is a long-term effort and an open-source project with an active community, hosted on GitHub. The Lean community is actively contributing to Lean's mathematical library, Mathlib and has already digitized half of the undergrad curriculum. The two goals are to be done with the undergraded curriculum in less than 10 years and to prove Fermat's Last Theorem. Lean provides a platform for software verification and development, formalizing mathematics and education. Lean 4 is the latest version.
Recently, a Lean team was able to help Field Medalist Peter Scholze confirm a theorem.

5. The Importance of Type Theory

In the world of computer science and mathematics, type theory plays a crucial role as a formal system for classifying data into different types and defining valid operations on those types. Unlike traditional set theory that relies on axioms for defining sets and their properties, type theory utilizes rules of inference to reason about the relationships between types and terms - it does not need any axioms for an underlying implementation such as sets.

It was first created by Bertrand Russell in the 20th century because of foundational issues in set theory. Russell's paradox highlighted the need for more rigorous foundations in mathematics.

A main advantage of type theory is that everything, and I mean EVERYTHING: terms, functions, propositions, has a type. The Curry-Howard correspondence established a link between computer science and type theory by showing us that there is a one-to-one correspondence between proofs and programs or propositions and types.

This means that any proof can be theoretically represented as a program, and conversely, programs can also be used to extract proofs - something of high importance in the field of proof-carrying code. New avenues have been opened by this link between logic and computer science, as it is now possible to verify the correctness of a program or proof using a richly-typed language. This verification is only possible due to the unique property of most type theory models being intuitionistic. This intuitionistic foundation makes it a great candidate for theorem provers, which use computational aspects in their approach, which would be difficult to implement in systems with non-constructive proofs.

6. Applications

6.1 Lean in Mathematics

Lean has an active community that is trying to formalize all of mathematics into a repository of formalized mathematics proofs called mathlib. It has a wide range of mathematical topics like number theory, analysis, topology, and more. Everyone can contribute to this open-source project.
The goal of mathlib is to provide a reliable foundation of formalized mathematics that can be used in research, education and formal verification.

Mathlib does not only have elementary proofs, it also contains modern proofs such as the theorem of liquid vector spaces by Peter Scholze. He posed a challenge to formalize this theorem and on the 14th July 2022, a year and a half after the challenge, mathlib announced that they successfully formalized the theorem.

6.2 Lean in Computer Science

Lean can be used to verify not just mathematical proofs but also programs. (Keyword: Curry-Howard correspondence) For example, it can be used to verify the correctness of a small compiler. As the entire compiler has been written in Lean, it is possible to verify the compiler with great confidence. It is also possible to verify smaller algorithms in Lean, as it is possible to implement them quite easily. For example, list-reversing-algorithms that work inductively can be verified in Lean.

6.3 Lean in Teaching

Lean can be used to teach logic. This has been demonstrated by the Natural Number Game, which introduces users to the Lean syntax by teaching them about induction and simpler propositional logic proofs.

There is also the github repository Formalising Mathematics which teaches various math theorems by using Lean. It is a 8 week workshop that gets harder over time. Topics vary from propositional logic to analysis and group cohomology.

Theorem provers have also been used in the past by various professors in lectures.

7. Outro

Now that you know about the history of theorem provers and the importance of type theory, you can check out our NNG tutorial world that we have made for our project and maybe listen to our short presentation. We hope you enjoyed this blog post and learned something new about theorem provers and type theory. Have a great day!