Complex Analysis and Cauchy's Integral Theorem in Lean

Topic: Logic, Verification, Mathematics

Location: Vrije Universiteit Amsterdam, The Netherlands

Supervisors:

Dr. Robert Y. Lewis (`r.y.lewis@vu.nl`)
Dr. Jasmin Christian Blanchette (`j.c.blanchette@vu.nl`)

Background:

Proof assistants (also called interactive theorem provers) make it possible to develop computer-checked, formal proofs of theorems. The primary advantage of formal proofs is the extremely high trustworthiness of the result. Proof assistants are employed for hardware and software verification at AMD and Intel. Two recent groundbreaking applications are a verified C compiler and a verified operating system microkernel. They are also increasingly being used by mathematicians to check mathematical proofs.

Lean [1] is a disruptive proof assistant developed at Microsoft Research and Carnegie Mellon University. Lean draws on decades of experience in interactive and automatic theorem provers (e.g., Coq, Isabelle/HOL, and Z3). It is based on type theory, a highly expressive logic with a very rich dependent type system (similar to Coq's) that can capture correctness properties of programs (e.g., "the `quicksort` function returns a sorted list"). It is implemented in C++.

Objective:

To formalize modern mathematics, we must build libraries and theories about older mathematics. There is more to this than simply defining objects and writing theorems. A well-designed library has complete APIs for structures, good documentation, and tools for updating and maintaining code.

Different libraries of formal mathematics focus on different topics. The standard library for Lean, `mathlib` [2], has significant developments in algebra and topology but has little analysis. Lean Forward [3], a project at the Vrije Universiteit Amsterdam, aims to formalize modern results in number theory. But these results depend on background results from many other areas.

There are two main goals of this internship project. **The first goal is to develop formal libraries of complex analysis in Lean.** A target result would be Cauchy's integral theorem, or generalizations like [4]. This theorem is a major component needed for proving results about modular forms, which are a focus of Lean Forward and important for modern number theory. **The second goal of this project is to design and im-**

plement tools for library maintenance. Many programming languages offer syntactic linting tools, but proof assistants offer more opportunities. Lean has a powerful *metaprogramming* framework [5] for inspecting its own definitions. There are useful and novel ways to design a library that can check and correct itself.

This internship is an ideal opportunity to familiarize oneself with proof assistants and to get acquainted with the exciting research taking place at the **Vrije Universiteit Amsterdam**. This work will likely be part of a publication at an international conference (e.g., *Certified Programs and Proofs* or *Interactive Theorem Proving*).

Requirements:

We expect the student to be familiar with the λ-calculus and both imperative (e.g., C/C++) and functional programming (e.g., Haskell or OCaml) and to have a basic understanding of logic. We do *not* expect familiarity with a proof assistant, although students who have used Coq will find Lean familiar. Knowledge of Dutch is not required.

Compensation:

In addition to any compensation offered by the intern's home institution, we offer €250 per month and will reimburse travel expenses to Amsterdam.

References:

[1] Leonardo de Moura, Soonho Kong, Jeremy Avigad, Floris van Doorn, and Jakob von Raumer. *The Lean Theorem Prover (System Description).* CADE 2015, pp. 378–388.

[2] The mathlib Community. *The Lean mathematical library.* https://leanprover-community.github.io/papers/mathlib-paper.pdf

[3] `https://lean-forward.github.io`

[4] Wenda Li and Lawrence C. Paulson. *A Formal Proof of Cauchy's Residue Theorem.* ITP 2016.

[5] Gabriel Ebner, Sebastian Ullrich, Jared Roesch, Jeremy Avigad, Leonardo de Moura. *A Metaprogramming Framework for Formal Verification.* ICFP 2017, pp. 1–29.