Arithmetic and casting in Lean

Topic: Logic, Verification

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.

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:

Constructing formal proofs by hand can be tedious, especially when dealing with simple numeric or arithmetic facts. We often want the system to manage such proofs for us. Another related problem involves moving between related or embedded types, for example from $\mathbb{N}$ to $\mathbb{Z}$ to $\mathbb{Q}$. Many facts are preserved when we embed terms of one type into another type, but the embeddings can confuse automated tools and frustrate users.

Various proof assistants have tools and idioms for handling both of these tasks. Decision procedures for linear arithmetic are well known in the literature and can be implemented in proof-producing ways [2]. Lean has some support for continuous (rational) arithmetic; the problem for discrete (integer) arithmetic is harder, as is supporting quantifiers. Processes for manipulating casts can be incorporated into simplifiers or implemented independently.

The goal of this project is to adapt such tools for integer arithmetic and casting to Lean. This can be approached from many different directions. One is to design a *simplifier loop*, a tool that generalizes Lean's simplifier to incorporate simplifier procedures (*simprocs*). Both arithmetic and cast management can be implemented in this framework. The tools can also be approached independently, e.g. integer arithmetic can be designed as an extension of the current rational arithmetic or in a novel way.

Lean features a powerful *metaprogramming* framework that is used to write proof-producing automation using the language of Lean itself. The tools described here will be implemented in this language.

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. 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] Frédéric Besson. *Fast Reflexive Arithmetic Tactics: The Linear Case and Beyond.* TYPES 2006, pp. 48–62.

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