Proof Theory

Natural Deduction, Props as Types, Sequent Calculus, Ordinal Analysis, Automated Prover, Verification

There are a number of fascinating sub-fields in proof theory and in this course we explore the best of them, starting with structural proof theory (e.g. natural deduction & sequent calculus), which has some very powerful features. We are particular interested in the junction of proof theory with programming. Many believe the future of programming will be significantly influenced by automated verification of correctness of code - but what does that mean and how can it be achieved? There are already open source projects pointing the way to much more extensive use of formal verification.

We are also curious as to what impact treating a proof as a mathematical object has, and using mathematics itself to explore a proof. Automated provers and proof assistants are improving in quality all the time and we see what the latest techniques offer. We also look at other branches of proof theory, such as proof complexity, ordinal analysis and provability logic.

Modern proof theory has huge potential to revolutionize our approaches to the rigor with which we judge statements made in mathematics and programming.

Target Audience

This course is aimed at mathematicians and modern developers who need a better grasp of proof theory

Prerequisites

Good foundational mathematical education along with some programming experience.

Attendees can select which programming language they wish to use, as all concepts will be developed from first principles.

Contents of One-Day Training Course

Overview of Proof Theory

What is proof theory?
Many branches – structural proof theory, provability logic, proof mining, automated theorem proving, ...
Structural proof theory includes natural deduction/sequent calculus/hilbert
A proof as a mathematical object- can be manipulated and reasoned about like any other mathematical object

Formal Verification

Mathematically proving that code works in all circumstances will always be more desirable than unit testing it for known scenarios

Natural Deduction Intro

Judgment, evidence and witnesses
Depending on what kind of logic that interests us, differing judgments needed
Introduction rule
Elimination rule
Multiple premises and a single conclusion

Advanced Natural Deduction

Alternative representation styles
Assumptions / context / use of turnstile: ⊢
Real world ND usage: defining a language such as WHLSL (OTT) & WebAssembly

Proposition As Types

Relationship to Lambda Calculus
Propositions as types
Proofs as programs
Normalization as evaluation of programs
Nirvana: code is (provable) logic is code

Sequent Calculus

A sequent consists of propositions to the left (ANDed), a turnstile and propositions to the right (OR)
Cut elimination
Importance for linear logic
Supports multiple premises AND multiple conclusions (unlike natural deduction)

Automated Proving

Proof assistants
Automated theorem provers

Proof Verification

Deciding if a proof is correct
Comparing proofs

Proof Complexity

Determining complexity critical for practical automation
Identifying the number of steps needed for a valid proof
As problems get larger, proof size explodes

Ordinal Analysis

Created by Gerhard Gentzen, ordinal analysis helps with consistency of proofs
An infinitary proof calculus
Upper bounds & lower bounds

Provability Logic

'it is proved that ..'
Relationship to modal logic
The Gödel-Löb logic of provability

Project: Using Proof Theory

We conclude this course by exploring how proof theory can help us create a modern programming language