An introduction to proof theory

Alexandre Miquel
LIP, ENS de Lyon
alexandre.miquel@ens-lyon.fr

Background

Proof theory is the branch of mathematical logic that studies the structure of mathematical proofs, seen as mathematical objects. Historically, proof-theory was introduced by Hilbert, following the work of Peano, Frege, Russell and Dedekind. Together with model theory, set theory and recursion theory, proof theory constitutes one of the four pillars of the foundations of mathematics.

In the last fifty years, proof theory has been deeply renewed by the discovery of a strong correspondence between the concepts of proof theory and of functional programming, known as the *Curry-Howard correspondence* [1, 4, 2]. According to this correspondence, each formal proof can be seen as a functional program (a.k.a. the proofs-as-programs paradigm) that meets a specification given by the proved formula (a.k.a. the formulæ-as-types paradigm). The study of this correspondence had major theoretical outputs in logic, such as the discovery of type theory and of linear logic. On the practical side, the Curry-Howard correspondence led to the development of proof assistants such as Coq, MinLog, Agda, Plastic, NuPrl (based on type theory and its variants) while influencing the design of functional programming languages such as SML, Caml or Haskell.

The aim of this course is to introduce the main concepts and tools of proof theory, while explaining the connections with functional programming and type theory (i.e. the family of formalisms underlying proof assistants such as Coq). The main questions we shall address are the following ones:

1. What is a proof? What is the difference between a constructive proof and a non constructive proof? Why and how do proofs compute?

2. How to transform a proof into a program? How to get from a (possibly non constructive) proof of an existential statement $(\exists x)A(x)$ an effective procedure that computes an x such that $A(x)$ (i.e. a *witness*).

Organization

The whole course will last four weeks. It will consist of 8 lectures of 2 hours (4 hours per week) and 4 lectures devoted to exercises (2 hours per week). To validate the course, students will be given after the last lecture a list of exercises they will solve and return for correction after a fixed amount of time.

Outline of the course

The series of lectures is split in two parts:

Part I: Deduction systems This part is devoted to the formal definition of the notion of *proof* and to the study of its structural properties. For that, we shall present two deduction systems—*natural deduction* and *sequent calculus*, while introducing (and motivating) the fundamental distinction between *intuitionistic logic* and *classical logic*. The main tool we shall introduce is the procedure of *cut elimination*, from which we shall deduce two important structural properties of intuitionistic logic: the *disjunction property* and the *witness property*. We shall conclude this part by presenting several methods to translate classical logic into intuitionistic logic.

- Lecture 1: Natural deduction for intuitionistic logic
- Lecture 2: Sequent calculus for classical logic
- Lecture 3: Translating classical logic into intuitionistic logic

Part II: Typed systems This part is devoted to the correspondence between formal proofs and functional programs (the Curry-Howard correspondence). We shall first introduce the Curry-Howard isomorphism in the intuitionistic propositional calculus, and then show how it naturally extends to more powerful formalisms, from first-order arithmetic to higher-order arithmetic. We shall conclude this course by presenting the theory of Pure Type Systems, which is the logical foundation of modern proof assistants.

- Lecture 4: The Curry-Howard isomorphism
- Lecture 5: Gödel's system T for first-order arithmetic
- Lecture 6: Girard's system F for second-order arithmetic
- Lecture 7: From second-order logic to higher-order logic
- Lecture 8: From higher-order logic to Pure Type Systems (PTSs)

References

[1] H. B. Curry and R. Feys. *Combinatory Logic, vol. 1.* North-Holland. Amsterdam, 1958.

[2] J.-Y. Girard, Y. Lafont and P. Taylor. *Proofs and Types.* Cambridge University Press. 1989.

[3] K. Gödel. Über eine bisher noch nicht benützte Erweiterung des finiten Standpunktes. *Dialectica*, 12:280–287. 1958.

[4] W. A. Howard. *The formulae-as-types notion of construction*, Privately circulated notes. 1969.

[5] S. C. Kleene, On the interpretation of intuitionistic number theory. *Journal of Symbolic Logic*, 10:109–124. 1945.

[6] G. Kreisel. Interpretation of analysis by means of functionals of finite type. In A. Heyting, editor, *Constructivity in Mathematics*, 101–128. North-Holland, 1959.

[7] J.-L. Krivine, *Lambda-calculus, types and models.* Masson. 1993.

[8] A. S. Troelstra. *Metamathematical Investigations of Intuitionistic Arithmetic and Analysis.* Springer, 1973. With contributions of A. S. Troelstra, C. A. Smoryński, J. I. Zucker and W. A. Howard.