Implementing a propositional prover

LE QUANG Dung[*]
École Polytechnique

Abstract
In this report, I present the principal accomplishments realized in the project titled "Implementing a Propositional Prover" within the framework of the INF551 course. Additionally, I address the challenges confronted during the course of the project. Subsequently, I extrapolate insights gained from the experiential aspects of executing this project.

Contents

1 Introduction

The Curry-Howard theorem establishes a correspondence between logical propositions and λ-terms in the lambda calculus. Consequently, to demonstrate the validity of a proposition, one may identify the corresponding λ-terms. The objective of this project is to develop a prover for propositional logic that facilitates the verification and expeditious elaboration of proofs. Additionally, we undertake the implementation of dependent types, wherein both terms and types are integrated into coherent expressions.

2 Simply typed lambda calculus

The two main parts in the simply typed lambda calculus are:

- Simple types: includes types, generated by variables along with some other symbols such as arrows, product, coproduct,...
- λ-terms: includes terms, generated by variables, terms, with two typical operations: abstraction and application.

In this part, I have done building simple types as well as λ-terms. From there I performed basic problems such as type checking, type inference, and provided proofs based on Curry-Howard correspondence for some basic propositions.

Besides, in section 3, I implemented type `Nat` for natural numbers. However, I only stopped at adding introduction and elimination rules for type Nat without proposing definitions for the predecessor and addition functions.

3 Interactive prover

After creating the necessary environments and adding introduction and elimination rules, I performed proofs for the proposed propositions. The proofs have been saved in the `proofs` folder.

4 Dependent type

In this part, I have completed the construction of type expr, from which important functions such as `to_string`, `subst`, `normalize`,

[*]Contact: dung.le-quang@polytechnique.edu

`infer`, etc. have been built. I have not yet given an answer to question 5.14, about the definition Addition, multiplication and some of their properties.

References