Proofs

Arthur Ryman, arthur.ryman@gmail.com

October 17, 2018

Abstract

This article is a Z Notation specification for proofs and proof checking. It has been type checked by *f*UZZ. The definitions that appear here are taken from Lemmon's book, *Beginning Logic*. The purpose of this specification is to guide the development of a proof checker aimed at Z specifications

1 Introduction

For a long time I have thought that it would be extremely useful to be able to write formal proofs concerning the mathematical objects defined in Z specifications. There are some very mature proof assistants available. I know something about Coq, but unfortunately it's style of proof is very different that that one finds in mathematical papers. A Coq proof consists of a list of tactics that represent higher level aggregates of deductions. This makes sense for a proof assistant since its job is to help the user discover proofs.

While a proof assistant might make sense is some contexts, proper development of a mathematical paper consists of a gradual introduction of concepts and lemmas leading to the main results. The proofs should, in some sense, write themselves. The focus of a mathematical paper should be on explanation and clarity. The proofs should be easy to read. I'd therefore really like something that would let me write and check natural looking proofs.

In contrast to Coq, the style of proof presented by Lemmon is very clear and explicit. However, the task of checking such a proof could easily be delegated to a program, much the same way that *f*UZZ type checks Z.

I believe that the kernel of a proof checker could be very small. It is basically an engine driven by a set of deduction rules. The engine simply needs to check that each deduction rule gets applied correctly. Even if it turns out that writing such an engine is too much work, the exercise of developing at least a simple version should give me a greater appreciation of tools like Coq and enable me to use them more productively.

My plan of attack is to formalize the concept of proof as described by Lemmon, starting with the Propositional Calculus.

2 Propositional Variables

The Propositional Calculus focuses on the form of statements and arguments, without being concerned about the subject matter described by those statements. This feat of abstraction is accomplished by using *propositional variables* that stand for arbitrary statements. The only real restriction here is that these statements must be either *true* or *false* in some given context.

2.1 P `\propP`, Q `\propQ`, R `\propR`, S `\propS`, and T `\propT`

Traditionally, these statements are represented by the letters P, Q, R, S, and T.

2.2 $'$ `\propPrime`

If more statements need to be represented then the letters are decorated with one or more primes, e.g. P', Q'', etc.

2.3 $\mathit{PropVar}$

Let $\mathit{PropVar}$ denote the set of propositional variables.

$$\mathit{PropVar} ::= P \mid Q \mid R \mid S \mid T \mid (\_') \langle\langle \mathit{PropVar} \rangle\rangle$$

Example. *P', Q'', and R are propositional variables.*

$$P' \in \mathit{PropVar}$$

$$Q'' \in \mathit{PropVar}$$

$$R \in \mathit{PropVar}$$