Dependently Typed Languages in Statix

Jonathan Brouwer <j.t.brouwer@student.tudelft.nl>
Jesper Cockx <j.g.h.cockx@tudelft.nl>
Aron Zwaan <a.s.zwaan@tudelft.nl>

September 16, 2022

Abstract

Text

1 Introduction

Spoofax is a textual language workbench: a collection of tools that enable the development of textual languages. When working with the Spoofax workbench, the Statix meta-language can be used for the specification of static semantics.

Dependently typed languages are different from other languages because they allow types to be parameterized by values. This allows more rigorous reasoning over types and the values that are inhabited by a type. This expressiveness also makes dependent type systems more complicated to implement. Especially, deciding equality of types requires evaluation of the terms they are parameterized by.

This goal of this paper is to investigate how well Statix is fit for the task of defining a simple dependently-typed language. We want to investigate whether typical features of dependently typed language can be encoded concisely in Statix. The goal is not to show that Statix can implement it, but that implementing it is easier in Statix than in a general-purpose programming language.

We will first show the base language and explain the way that Statix was used to implement this language. Next, we will explore several features and see how well they can be expressed in Statix.

Features - Base language (sec 2) - Name collision avoidance (sec 3) - Language parametric services (sec 4) - Inference (sec 5) - Data types (sec 6)

2 Base Language

The base language that was implemented is the Calculus of Constructions, with a syntax somewhat similar to that of Haskell.

- Base language is calculus of constructions with lets + type assertions (move type assertions to sec 5?) - Describe syntax of the base language? - Describe rules of base language (statix syntax or mathy?) - Scope graphs for substitutions + scopes

3 Name Collisions

Ways: - Uniquify at the start, doesn't work (example) - Rename terms using statix rules (works, complex) - Using scope graphs

References