Categorical Semantics of Type Theories

Harley D. Eades III
Computer Science
The University of Iowa

April 27, 2013

Abstract

Category theory and its applications to type theory are well known and have been explored extensively. In this manuscript we present basic category theory and give a categorical semantics to a large class of type theories. In this document we emphasize rigor and give as much detail as possible with respect to the abilities of the authors. This document will also be self contained.

Contents

List of Figures

0.1 Introduction

We give an amazingly rich and engaging introduction.

Chapter 1

Category Theory

We detail all the results in category theory needed to understand everything in this document.

Chapter 2

Simple Type Theories

2.1 A Metaframework

We will use Martin-Löf's Type Theory for our metaframework throughout this chapter. This will allow use to give precise types to all of the structures in our object languages. In fact all of the type theories discussed in this chapter can be rigorously defined within this framework where binding can be encoded using either de Bruijn indecies or using the locally nameless representation [1] [**?**].

2.2 The Theory of Constants

We begin our journey into the world of categorical semantics of type theories by first showing how to interpret a simple algebraic theory consisting of a countably infinite set of variables, a finite set of constant types, a finite set of i-ary function symbols, a typing judgment, and a definitional equality judgment. This theory is called the theory of constants. We first give a formal definition of this theory in the presentation we will adopt for the reminder of this document. The syntax for the theory of constants is defined in Figure 2.1.

The free variables of the theory can be defined at the metalevel as de Bruijn indices, but we will use mathematical notation to simplify the presentation. They have type Term and are classified by constant types of type Type. The i-ary function symbols have type $\mathbf{Term}^i \Rightarrow \mathbf{Term}$. The constant types of the theory of constants have meta-type Type. Judgments are metastatements describing what type a term can be assigned. All of the judgements we will define can be defined at the metalevel as an inductive datatype where each rule of the judgment is defined as a constructor. If we call the typing judgment has_type at the type level then its type is $(\Gamma : [\mathbf{Term} \times \mathbf{Type}]) \Rightarrow (t : \mathbf{Term}) \Rightarrow (U : \mathbf{Type}) \Rightarrow \mathbf{Type}$. We denote this judgment by $\Gamma \vdash t : U$. The type of the definitional equality judgment is similar. We define the type assignment judgment in Figure 2.2 and the definitional equality judgment in Figure 2.3.

[1] We prefer the latter.

$$
\begin{array}{rccl}
(Types) & T & ::= & S \mid U \\
(Terms) & t & ::= & x \mid c \mid f\, x_1 \ldots x_i \\
(Contexts) & \Gamma & ::= & x : T \mid \Gamma_1, \Gamma_2
\end{array}
$$

Figure 2.1: Syntax of the theory of constants

$$\frac{}{\Gamma, x : S, \Gamma' \vdash x : S}\ \textsc{Var} \qquad \frac{}{x_1 : S_1, \ldots, x_i : S_i \vdash f\ x_1 \ldots x_i : U}\ \textsc{Fun}$$

$$\frac{}{\Gamma \vdash c : U}\ \textsc{Const}$$

Figure 2.2: Type assignment for the theory of constants

$$\frac{}{\Gamma \vdash t = t : T}\ \textsc{Refl} \qquad \frac{\Gamma \vdash t_1 = t_2 : T}{\Gamma \vdash t_2 = t_1 : T}\ \textsc{Sym}$$

$$\frac{\Gamma \vdash t_1 = t_2 : T \quad \Gamma \vdash t_2 = t_3 : T}{\Gamma \vdash t_1 = t_3 : T}\ \textsc{Trans} \qquad \frac{\Gamma \vdash t_1 = t_2 : T_1 \quad \Gamma, x : T_1 \vdash t = t' : T_2}{\Gamma \vdash [t_1/x]t = [t_2/x]t' : T_2}\ \textsc{Subst}$$

Figure 2.3: Definitional equality for the theory of constants