Some title

Someone

November 14, 2016

Abstract

This is the paper's abstract …

1 Introduction

1.1 What are numbers?

1.2 Positional numeral systems

Outline The remainder of the thesis is organized as follows.

2 A gental introduction to dependently typed programming in Agda

There are already plenty of tutorials and introductions of Agda[1]. Nonetheless, we will provide a simple and self-contained tutorial in this section, covering the part (and only the part) we need in this work.

Some of the more advenced constructions (such as views and universes) used in the following sections will be introduced along the way.

We assume that all readers have some basic understanding of Haskell, and those who are familiar with Agda and dependently typed programming may skip this chapter.

2.1 Some basics

[introduce some backgrounds of Agda]

2.2 Dependent types

2.3 Simply typed programming in Agda

In the beginning there was nothing Unlike in other programming languages, there are no "built-in" datatypes[1] such as *Int*, *String*, or *Bool*. The reason is that they can all be created out of thin air, so why bother?

Let there be datatype Datatypes are introduced with `data` declarations. Here is a classical example, the type of booleans.

```
data Bool : Set where
    true  : Bool
    false : Bool
```

Three names are brought into scope:

1. `Bool`: the name of the datatype
2. `true`: a constructor of the datatype
3. `false`: another constructor of the datatype

The declaration also explicitly spicifies the types of these newly introduced entities.

1. `Bool` has the type of `Set`[2]
2. `true` has the type of `Bool`
3. `false` has the type of `Bool`

Pattern matching Similar to Haskell, datatypes are eliminated with pattern matching.

Here's a function that pattern matches on `Bool`.

```
not : Bool → Bool
not true  = false
not false = true
```

[1] Small types, to be precise

[2] `Set` is the type of small types, and `Set` is the type of `Set`, and so on. They form a hierarchy of types.

[1] U. Norell. Dependently typed programming in agda. In *Advanced Functional Programming*, pages 230–266. Springer, 2009.