Lab exercise 1 : T-diagrams

Ferdinand van Walree(3874389) and Matthew Swart(5597250)

November 29, 2015

1 Introduction

In this document we explain the design and implementation of a DSL for generating T-diagrams in Latex.

2 Design

This section gives a short eplanation of the pipeline of our 'compiler'. There are four programs that together form the pipeline. Those are ParseTDiag.hs, TypeCDiag.hs, TDiag2Picture.hs and PpPicture.hs. ParseTDiag.hs is responsible for parsing the DSL as defined in the project description. TypeCDiag.hs is our implemented typesystem and typechecks the generated ATerms given as output from ParseTDiag.hs. TypeCDiag simply returns the ATerms it receives from ParseTDiag.hs if the input has been considered well-typed. TDiag2Picture.hs receives it input from TypeCDiag.hs and translates the ATerms into a structure containing latex code for generating T-diagrams. Finally PpPicture.hs is passed ATerms from TDiag2Picture.hs and generates the latex code for generating T-diagrams.

3 Typing-system

We won't be talking about our implementation just yet. First we'll tell you what kind of T-diagram structures we do and do not allow. First of all, all non-sensical diagrams as described in the project documentation are considered ill-typed:

- Executing a platform
- Executing a program, interpreter, or compiler on a program or a compiler
- Executing a program, interpreter, or compiler on a nonmatching platform or interpreter
- Compiling a platform

- Compiling a program, interpreter, or compiler with a program, a platform or an interpreter
- Compiling a program, interpreter, or compiler with a compiler for an incompatible source language