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

Unfortunately we had to place some more restrictions on the diagrams that we allow. We also consider the following diagrams to be ill-typed:

- Executing a program, interpreter, or compiler on a compiled interpreter
- Compiling an executed program, interpreter or compiler
- Compiling a program, interpreter, or compiler on an executed compiler
- Compiling a compiled program, interpreter or compiler using a compiled compiler
- Compiling a program, interpreter, or compiler using a left recursive compilation
- Compiling a right recursive compilation using a compiler

Now the last two items probably won't make a lot of sense just yet. So let us explain. If we are given the input Compile L with R end, where L and R are diagrams. If L is a Compile, then we say that Compile L with R end is left recursive. likewise if R is a Compile, then we say that it is right recursive. So the third-last diagram: Compiling a compiled program, interpreter or compiler using a compiled compiler, basically says that we do not allow a compile to be both left recursive and right recursive at the same time. The second-last diagram says that we cannot Compile some diagrams using a compilation that contains left recursive compilation. In otherwords, we cannot compile some diagram using a compiled compiler that was compiled using another compiler. The last diagram says that we cannot Compile a right recursive compilation using a compiler. In other words, the diagram that we are trying to compile must not have been compiled with a compiled compiler.

We have a formal specification of our typesystem, which you can find in typesystem.pdf.

4 Implementation

4.1 TypeCDiag

4.1.1 AG Attributes

4.1.2 Type checking

4.2 TDiag2Picture

4.2.1 AG Attributes

4.2.2 Translation