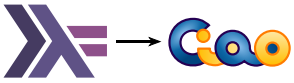

Haskell to Ciao: Translation of Haskell code for Static Resource Analysis in CiaoPP

David Munuera Mazarro
david.munuera@imdea.org

The goals of this project are:

- Creating a transpiler (Source-to-Source compiler) that takes Haskell code (or, more specifically, GHC Core code) and yields code for an equivalent Ciao Prolog program
- Perform a static resource analysis running the CiaoPP engine over the Ciao code provided by the translation
- Draw conclusions from the analysis results and take them to the original Haskell domain from which we started

The utmost objective of the project is to be able to take some Haskell code, perhaps meeting some conditions imposed by the limitations of the translation, and obtain resource usage properties inferred by the CiaoPP engine accurately matching said Haskell code.

There are a few challenges posed by this task. First of all, the translation has to be **correct**, because otherwise the results and conclusions taken from the analysis would only be valid for the Ciao code resulting from the translation, and not necessarily all the way back to the original code. In addition, we should aim for the translation to be **as complete as possible** in order to be able to analyze real, useful Haskell programs and not just little example snippets. But there may not be an immediate translation for certain basic Haskell features into basic Ciao features (for example, Haskell's basic types are different from Ciao's), and for every such situation we encounter, we will either have to:

a) Find a workaround translation that remains compatible with CiaoPP, or

b) If the above proves to be difficult or not feasible, carry the burden into the transpiler and regard it as a limitation

Moving back to the correctness of the translation. Ideally, the translation should be formally proven to be correct. However, I do not currently have the theoretical knowledge to do so, and the way I intend to approach the project is by trial-and-error; using the transpiler to translate example programs, check that they behave as they should, and if that's not the case, tweak the transpiler as needed until the behavior of the translation matches that of the original program.