Using Partial Evaluation to Generate Compilers

Niklas Deworetzki

November 18, 2020

1 Introduction

2 Instruments of Partial Evaluation

3 Multistage Computations

When solving computational problems, the required calculations can be performed either in a single computational stage, or can be separated into multiple stages. This is especially obvious when working with compiled languages, since they require the help of a compiler to generate executable program code, that is able to perform calculations. Since multistage computations will be a recurring theme in the following sections, we now introduce a notation to reason about it in a formal way. This notation was adopted from .

Given a program to solve an arbitrary computational program, this program is defined by its source code written in a language S. Applying this program to an input will produce an output by performing the computations described in the program.

$$\texttt{output} \;=\; [\![\texttt{source}]\!]_{\texttt{S}}\ \texttt{input}$$

Since a programming language S usually cannot be executed directly by the underlying hardware, we can use an interpreter written in another language L to perform the calculations. This interpreter then accepts the source program as well as the original input to produce an output.

$$\texttt{output} \;=\; [\![\texttt{interpreter}]\!]_{\texttt{L}}\ [\texttt{source}, \texttt{input}]$$

Alternatively a compiler can be used, that first accepts the source program to produce a target program, which then can be applied to the original input.

$$\begin{aligned} \texttt{output} &= [\![[\![\texttt{compiler}]\!]_{\texttt{L}}\ \texttt{source}]\!]_{\texttt{T}}\ \texttt{input} \\ &= [\![\texttt{target}]\!]_{\texttt{T}}\ \texttt{input} \end{aligned}$$

The nested brackets make it obvious, that multiple stages of computation are present.

4 Futamura Projections