Paper

Simon van Hus
6147879
s.vanhus@students.uu.nl

Abstract

Abstract.

1 Tracing

When we trace a function we evaluate it whilst keeping track of every basic operation, so we have a linear list of operations to reach the result. This sounds fairly straightforward, but is not quite as trivial as it seems. Tracing is rarely a well-defined term, which is interesting because it's a technique that is used quite a lot.

To define tracing let us start by defining a source language, with which to write a function we'd want to trace. This source language is a basic lambda calculus expressed in a Haskell-like syntax in Listing 1. This listing shows support for real numbers, functions, and Boolean values as types, and lambda calculus terms. The $op(t_1, \cdots, t_n)$ term represents a basic mathematical operation of the type $\mathbb{R}_n \to \mathbb{R}$. It should also be noted that the type $\sigma \to \tau$

Now with source language defined, we need to formalize exactly what we expect a trace function to do with a program written in the source language. We have a couple of criteria for the result of our tracing function:

1. All branches should be eliminated
2. All functions should be eliminated
3. A record of each performed operation should be kept

We can assure that these specifications are met, by defining a target language which excludes operations that violate these criteria. For instance, if we eliminate lambdas and if-then-else statements, we have already fulfilled criteria 1 and 2. The third criteria doesn't talk explicitly about what we should eliminate from the source language, but it does tell us we need to preserve all basic operations. After all, if we resolve even those, we'll end up with just the final result – which would be the same as just evaluating the program.

So, we define a target language that meets these criteria in Listing 2. We can clearly see that we're left only with the basic operations ($op(t_1, \cdots, t_n)$), real numbers, and Booleans. Now all that remains is to actually transform a program in the source language to a trace in the target language.

Types:

$\sigma, \tau := \mathbb{R} \mid \sigma \to \tau \mid \top \mid \bot$

Terms:

$$
\begin{aligned}
& s, t := \\
& \quad s\ t \\
& \mid \lambda(x : \tau).t \\
& \mid op(t_1, \cdots, t_n) \\
& \mid r \in \mathbb{R} \\
& \mid b \in \{\top, \bot\} \\
& \mid \texttt{let}\ s\ \texttt{in}\ t \\
& \mid \texttt{if}\ s : \{\top, \bot\}\ \texttt{then}\ t_\top\ \texttt{else}\ t_\bot
\end{aligned}
$$

Listing 1: Source language

Types:

$\sigma, \tau := \mathbb{R} \mid \sigma \to \tau \mid \top \mid \bot$

Terms:

$$
\begin{aligned}
& \quad op(t_1, \cdots, t_n) \\
& \mid r \in \mathbb{R} \\
& \mid b \in \{\top, \bot\}
\end{aligned}
$$

Listing 2: Target language