Normalization by Evaluation in the Delay Monad

Andreas Abel¹ and James Chapman²

¹ Chalmers and Gothenburg University, Sweden
andreas.abel@gu.su
² University of Strathclyde, Glasgow, Scotland
james.chapman@strath.ac.uk

We present an Agda formalization of a normalization proof for simply-typed lambda terms. The normalizer consists of two coinductively defined functions in the delay monad: One is a standard evaluator of lambda terms to closures, the other a type-directed reifier from values to \(\eta\)-long \(\beta\)-normal forms. Their composition, normalization-by-evaluation, is shown to be a total function a posteriori, using a standard logical-relations argument. The normalizer is then shown to be sound and complete. The completeness proof proof is dependent on termination. We also discuss a variation on this normalizer where environments used by the evaluator contain delayed values which can be proven complete independently of termination using weak bisimilarity. This approach would be a realisation of an aim of this work to present a modular proof of normalization where termination, soundness and completeness are independent.

The successful formalization serves as a proof-of-concept for coinductive programming and reasoning using sized types and copatterns [3], a new and presently experimental feature of Agda [4].

Termination of a normalizer was described in [2]. The soundness and completeness proofs are new[1] and the alternative normalizer with delayed environments and accompanying normalization proof is ongoing work.

Delay Monad and potential non-termination. The delay monad [5] captures the idea of a computation that may return a value eventually or not at all. We represent functions that have not yet been proven terminating and are therefore untrusted as functions from values of type \(A\) to delayed computations of type \(\text{Delay } B\). Proving termination (asserting a basic level of trustworthiness) amounts to proving that for any input value the delayed computation will converge to a value. Given a constructive proof of termination one can derive a function from values of type \(A\) to values of type \(B\).

Normalization algorithm. The normalization algorithm consists of two main components: (1) an evaluator that takes typed terms to intermediate values given an environment explaining the variables; and (2) a typed directed reifier that takes intermediate values to syntact \(\eta\)-long \(\beta\)-normal forms. Neither component is apriori terminating but we can nonetheless combine them using monadic bind.

\[
\begin{align*}
\text{eval} & : \text{Tm } \Gamma \sigma \to \text{Env } \Delta \Gamma \to \text{Delay } (\text{Val } \Delta \sigma) \\
\text{reify} & : \text{Val } \Delta \sigma \to \text{Delay } (\text{Nf } \Delta \sigma) \\
\text{nf} \ t & = \text{eval } \text{id } t \gg\gg \text{reify}
\end{align*}
\]

Normalization theorem. We prove three theorems about the normalization algorithm:

\[
\begin{align*}
\text{termination} &: \forall (t : \text{Tm } \Delta \sigma) \to \exists (n : \text{Val } \Delta \sigma). \text{nf } t \Downarrow n \\
\text{soundness} &: \forall (t : \text{Tm } \Delta \sigma) \to t \equiv_{\beta\eta} \text{nf } t \\
\text{completeness} &: \forall (t t' : \text{Tm } \Delta \sigma) \to t \equiv_{\beta\eta} t' \to \text{nf } t \equiv \text{nf } t'
\end{align*}
\]
Decoupling soundness and completeness from termination amounts to a lifting of the soundness predicate and completeness relation to the Delay monad, i.e., saying that the predicate/relation would hold eventually. In the relation case this is bisimilarity. For the algorithm specified above this is possible for soundness but not completeness. For a modified algorithm where environments contain delayed values completeness should also be possible but this presents technical challenges such as potentially moving to a sized version of the Delay monad which is not well supported by current versions of Agda and moving from reasoning up to equality to reasoning up to weak bisimilarity.

References


