Normalization by Evaluation in the Delay Monad

Andreas Abel[1] and James Chapman[2]

[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 η-long β-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 $\mathsf{Delay}\ B$. 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 reifyer that takes intermediate values to syntact η-long β-normal forms. Neither component is apriori terminating but we can nonetheless combine them using monadic bind.

$$\begin{array}{lcl}
\mathsf{eval} & : & \mathsf{Tm}\ \Gamma\ \sigma\ \rightarrow\ \mathsf{Env}\ \Delta\ \Gamma\ \rightarrow\ \mathsf{Delay}\ (\mathsf{Val}\ \Delta\ \sigma) \\
\mathsf{reify} & : & \mathsf{Val}\ \Delta\ \sigma\ \rightarrow\ \mathsf{Delay}\ (\mathsf{Nf}\ \Delta\ \sigma) \\
\mathsf{nf} & : & \mathsf{Tm}\ \Delta\ \sigma\ \rightarrow\ \mathsf{Delay}\ (\mathsf{Nf}\ \Delta\ \sigma) \\
\mathsf{nf}\ t & = & \mathsf{eval}\ \mathsf{id}\ t\ \ggg\!=\ \mathsf{reify}
\end{array}$$

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

$$\begin{array}{lcl}
\mathsf{termination} & : & \forall\,(t\quad : \mathsf{Tm}\ \Delta\ \sigma) \rightarrow \exists\,(n : \mathsf{Val}\ \Delta\ \sigma).\ \mathsf{nf}\,t \Downarrow n \\
\mathsf{soundness} & : & \forall\,(t\quad : \mathsf{Tm}\ \Delta\ \sigma) \rightarrow t\ \cong_{\beta\eta}\ \mathsf{nf}\ t \\
\mathsf{completeness} & : & \forall\,(t\ t' : \mathsf{Tm}\ \Delta\ \sigma) \rightarrow t\ \cong_{\beta\eta}\ t'\ \rightarrow\ \mathsf{nf}\ t\ \equiv\ \mathsf{nf}\ t'
\end{array}$$

Decoupling soundess and completeness from termination amounts to a lifing of the soundness predicate and completeness relation to the 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 bisimularity.

References

[1] Andreas Abel and James Chapman. Normalization by evaluation in the delay monad: Formalization. `http://github.com/andreasabel/continuous-normalization`.

[2] Andreas Abel and James Chapman. Normalization by evaluation in the delay monad: A case study for coinduction via copatterns and sized types. In Paul Levy and Neel Krishnaswami, editors, Proceedings 5th Workshop on *Mathematically Structured Functional Programming,* Grenoble, France, 12 April 2014, volume 153 of *Electronic Proceedings in Theoretical Computer Science*, pages 51–67. Open Publishing Association, 2014.

[3] Andreas Abel and Brigitte Pientka. Well-founded recursion with copatterns and sized types. *Journal of Functional Programming*, 26:61, 2016. ICFP 2013 special issue.

[4] AgdaTeam. The Agda Wiki, 2016.

[5] Venanzio Capretta. General recursion via coinductive types. *Logical Methods in Computer Science*, 1(2), 2005.