Types 2006

Nottingham

Semantic normalisation proofs

Ulrich Berger

Swansea

Basic idea

Interpret λ-terms with recursively defined constants in a domain model such that

$$[M] \neq \perp \quad \text{implies} \quad \mathrm{SN}(M)$$

Running example

Gödel's system T

Simply typed λ-calculus with primitive recursion in all types:

$$\begin{aligned} \mathrm{R}\ x\ f\ 0 &\mapsto x \\ \mathrm{R}\ x\ f\ \mathrm{S}(k) &\mapsto f\ k\ (\mathrm{R}\ x\ f\ k) \end{aligned}$$

Main components of the method

Basic SN: If A is recursion free, then $\mathrm{SN}(A)$.

Continuity: $[M] = \bigsqcup_n [M_n]$ where $M_n := M[\mathrm{R}_n/\mathrm{R}]$ and

$$
\begin{array}{rcl}
\mathrm{R}_{n+1}\ x\ 0 & \mapsto & x \\
\mathrm{R}_{n+1}\ x\ f\ \mathrm{S}(k) & \mapsto & f\ k\ (\mathrm{R}_n\ x\ f\ k)
\end{array}
$$

Strictness: If $[A] \neq \bot$, then $\mathrm{R}_0 \notin A$ (note $[\mathrm{R}_0] = \bot$).

$[M] \neq \bot$ implies $\mathrm{SN}(M)$:

$$
\begin{array}{rccccc}
[M] \neq \bot\text{:} & M & \rightarrow & M' & \rightarrow & \ldots \\
A \text{ recursion free, } [A] \neq \bot\text{:} & A & \rightarrow & A' & \rightarrow & \ldots
\end{array}
$$

Applications

Gödel's T: Suffices to show that all terms are total and hence $\neq \perp$.

Note that totality is a semantic analogue to the method of reducibility candidates.

The method can also be applied to prove normalisation w.r.t. restriced reduction: Make operators strict only at argument places where reduction is allowed.

Bar recursion (Spector, Berardi/Bezem/Coquand):

$$\Phi\ s = \text{if } Y\hat{s} < |s| \text{ then } Hs \text{ else } Gs(\lambda x.\Phi(s{*}x))$$

$$\Psi\ p = Y(\lambda k.\text{if } p \downarrow k \text{ then } p[k] \text{ else } Gk(\lambda x.\Psi(p{*}(k,x))))$$

Variant by Coquand and Spiwack

In an algebraic domain we have

$$[M] = \bigsqcup \{U \text{ finite} \mid U \sqsubseteq [M]\}$$

The relation

$$M\!:\!U \;\;:\Leftrightarrow\;\; U \sqsubseteq [M]$$

has an inductive definition similar to the typing rules for intersection types. In fact, an adaptation of the usual candidate method yields:

Theorem (Coquand/Spiwack). If $M\!:\!U$, then $\mathrm{SN}(M)$.

Note that no basic SN assumption is made.

Comparison

Coquand/Spiwack

- The candidate proof is done once and for all.
- For a specific type system it suffices to prove totality (technically easier than candidate method; amounts to embedding the system into intersection types).
- Suitable for formalisation in type theory.

B

- Does not include termination proof for underlying type system.
- More abstract and hence open to systems other than typed λ-calculi ($\rightarrow$ CSL'05).

Conclusion

- Termination proof for a recursion scheme is reduced to a semantic totality argument, i.e. the intuitive raison d'être for the scheme.
- Continuity (magically) reduces a complicated recursion to a simple ω-iteration.
- Further work:
  - Relax the syntactic restrictions on rewrite rules, allowing e.g. $(x+y)+z \mapsto x+(y+z)$
  - Corecursion
  - Dependent types ($\longrightarrow$ Coquand/Spiwack)
  - Abstract from λ-calculus to more general systems.

References

[1] C. Spector. Provably recursive functionals of analysis: a consistency proof of analysis by an extension of principles in current intuitionistic mathematics. In F. D. E. Dekker, editor, *Recursive Function Theory: Proc. Symposia in Pure Mathematics*, volume 5, pages 1–27. American Mathematical Society, Providence, Rhode Island, 1962.

[2] W.W. Tait. Normal form theorem for barrecursive functions of finite type. In J.E. Fenstad, editor, *Proceedings of the Second Scandinavian Logic Symposium*, pages 353–367. North–Holland, Amsterdam, 1971.

[3] H. Vogel. Ein starker Normalisationssatz für die barrekursiven Funktionale. *Archive for Mathematical Logic*, 18:81–84, 1985.

[4] G. D. Plotkin. LCF considered as a programming language. *Theoretical Computer Science*, 5:223–255, 1977.

[5] M. Bezem. Strong normalization of barrecursive terms without using infinite terms. *Archive for Mathematical Logic*, 25:175–181, 1985.

[6] J. van de Pol and H. Schwichtenberg. Strict functionals for termination proofs. In M. Dezani-Ciancaglini and G. Plotkin, editors, *Typed Lambda Calculi and Applications*, volume 902 of *LNCS*, pages 350–364. Springer Verlag, Berlin, Heidelberg, New York, 1995.

[7] F. Blanqui, J-P. Jouannaud, and M. Okada. The calculus of algebraic constructions. In P. Narendran and M. Rusinowitch, editors, *Proceedings of RTA'99*, number 1631 in LNCS, pages 301–316. Springer Verlag, Berlin, Heidelberg, New York, 1999.

[8] S. Berardi, M. Bezem, and T. Coquand. On the computational content of the axiom of choice. *Journal of Symbolic Logic*, 63(2):600–622, 1998.

[9] T. Coquand and A. Spiwack. Proof of strong normalisation using domain theory. 2006.

[10] B. A computational interpretation of open induction. In F. Titsworth, editor, *Proceedings of the Nineteenth Annual IEEE Symposium on Logic in Computer Science*, pages 326–334. IEEE Computer Society, 2004.

[11] B., Strong normalization for applied lambda calculi. Logical Methods in Computer Science 1(2), 1–14, 2005.

[12] B., An abstract strong normalization theorem. Proceedings of CSL'05, Oxford, LNCS 3634, 2005.