A calculus of single substitutions for simple type theory

Ambrus Kaposi, Norbert Luksa

Department of Programming Languages and Compilers , Eötvös Loránd University

{akaposi|luksan}@inf.elte.hu

Church's simple type theory (or simply typed lambda calculus) is the main foundational language for describing type systems for programming languages. Polymorphic type systems [3] and dependent type systems [6] both build on the foundation of simple type theory. Traditionally [4], simple type theory is described as a relation on preterms together with an operational semantics given as a binary relation on preterms. Typing and operational semantics are usually related through preservation: if a term has type A and is related to another term by the operational semantics, then this new term also has type A.

Algebraic descriptions of type theory [2] were recently back-ported to simple type theory using the term simply typed category with families (CwF) [1]. In an algebraic description, there are only well-typed terms which are quotiented by definitional equality, hence type preservation is trivial. In this work we investigate an alternative description of simple type theory, which does not build on a category naturally giving rise to parallel substitutions, but on single weakenings and single substitutions. In our calculus, parallel substitutions are not even mentioned. Our calculus is close to the usual informally written syntax of simple type theory. The main substitution operator takes terms $t : \mathsf{Tm}(\Gamma \triangleright \mathsf{B} +\!+ \Delta)\mathsf{A}$, and $u : \mathsf{Tm}\Gamma\mathsf{B}$ and results in a substituted term $t[\Gamma, u, \Delta] : \mathsf{Tm}(\Gamma +\!+ \Delta)\mathsf{A}$ where the $+\!+$ operator concatenates contexts.

Compared to simply typed CwFs, our calculus has fewer operators (four fewer as we don't need to talk about parallel substitutions) but six more equations (as we need to describe interactions of single substitutions and weakenings). Contextual CwFs have the property that every context is built out of a finite number of types. We show that models of our calculus are equivalent to contextual simply typed CwFs with a base type and function space. We expect that our calculus is close to calculi with only normal forms such as hereditary substitutions [5]. We plan to investigate its generalisation to dependent types, obtaining a telescopic variant of CwFs.

References

[1] Simon Castellan, Pierre Clairambault, and Peter Dybjer. "Categories with Families: Unityped, Simply Typed, and Dependently Typed". In: *CoRR* abs/1904.00827 (2019). arXiv: 1904.00827. URL: http://arxiv.org/abs/1904.00827.

[2] Peter Dybjer. "Internal Type Theory". In: *Lecture Notes in Computer Science*. Springer, 1996, pp. 120–134.

[3] Jean-Yves Girard. "The System F of Variable Types, Fifteen Years Later." In: *Theor. Comput. Sci.* 45.2 (1986), pp. 159–192. URL: http://dblp.uni-trier.de/db/journals/tcs/tcs45.html#Girard86.

[4] Robert Harper. *Practical Foundations for Programming Languages*. 2nd. New York, NY, USA: Cambridge University Press, 2016.

[5] Chantal Keller and Thorsten Altenkirch. "Hereditary Substitutions for Simple Types, Formalized". In: *Proceedings of the 3rd ACM SIGPLAN Workshop on Mathematically Structured Functional Programming, MSFP@ICFP 2010, Baltimore, MD, USA, September 25, 2010*. Ed. by Venanzio Capretta and James Chapman. ACM, 2010, pp. 3–10. ISBN: 978-1-4503-0255-5. DOI: 10.1145/1863597.1863601. URL: https://doi.org/10.1145/1863597.1863601.

[6] Per Martin-Löf. "Constructive Mathematics and Computer Programming". In: *Proc. Of a Discussion Meeting of the Royal Society of London on Mathematical Logic and Programming Languages*. London, United Kingdom: Prentice-Hall, Inc., 1985, pp. 167–184. ISBN: 0-13-561465-1. URL: http://dl.acm.org/citation.cfm?id=3721.3731.