The System SOL: Second-Order Laboratory

Makoto Hamana[1], Tatsuya Abe[2], Yuito Murase[3], Kazuhiko Sakaguchi[4]

[1] Department of Computer Science, Gunma University, Japan
hamana@cs.gunma-u.ac.jp
[2] STAIR Lab, Chiba Institute of Technology, Japan
abet@stair.center
[3] Department of Computer Science, University of Tokyo, Japan
murase@lyon.is.s.u-tokyo.ac.jp
[4] Department of Computer Science, University of Tsukuba, Japan
murase@lyon.is.s.u-tokyo.ac.jp

SOL is a Haskell-based tool that assists the proofs of confluence and strong normalisation of higher-order computation. SOL is intended to be a tool that is applicable to the modern theories of higher-order programming languages. This aim is demonstrated in the first author's recent article [Ham17b], which is presented at ICFP'17.

The system SOL supports multiple formats as input, including, CoCo's HRS format, XML higher-order rule format in TPDB, and SOL's original format. The last format is written in an embedded domain specific language using the feature of Template Haskell.

Based on the foundation of second-order algebraic theories [FH10] and its computational counter part [Ham16, Ham17b], we implemented various results on higher-order syntax and computation in SOL, including Knuth and Bendix's criterion for confluence, deterministic second-order pattern matching [YHT04] for matching, and Function-as-Constructor Unification (FCU) [LM16] for unification. We implemented it by extending Nipkow's functional implementation of higher-order pattern unification [Nip93], which we report in [Ham17a]. Termination analysis is based on the General Schema criterion [Bla00].

References

[Bla00] F. Blanqui. Termination and confluence of higher-order rewrite systems. In *Rewriting Techniques and Application (RTA 2000)*, LNCS 1833, pages 47–61. Springer, 2000.

[FH10] M. Fiore and C.-K. Hur. Second-order equational logic. In *Proc. of CSL'10*, LNCS 6247, pages 320–335, 2010.

[Ham16] M. Hamana. Strongly normalising cyclic data computation by iteration categories of second-order algebraic theories. In *Proc. of FSCD'16*, volume 52 of *the Leibniz International Proceedings in Informatics (LIPIcs)*, pages 21:1–21:18, 2016.

[Ham17a] M. Hamana. A functional implementation of function-as-constructor higher-order unification. *Proc. 31st International Workshop on Unification (UNIF'17)*, 2017. to appear.

[Ham17b] M. Hamana. How to prove your calculus is decidable: practical applications of second-order algebraic theories and computation. *Proceedings of the ACM on Programming Languages*, 1(1):22:1–22:28, 9 2017. to appear in ICFP'17.

[LM16] T. Libal and D. Miller. Functions-as-Constructors Higher-Order Unification. In *Proc. of FSCD 2016*, volume 52 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 26:1–26:17, 2016.

[Nip93] T. Nipkow. Functional unification of higher-order patterns. In *Proc. of (LICS'93)*, pages 64–74, 1993.

[YHT04] T. Yokoyama, Z. Hu, and M. Takeichi. Deterministic second-order patterns. *Inf. Process. Lett.*, 89(6):309–314, 2004.