ACP: System Description for CoCo 2019

Takahito Aoto[1] and Masaomi Yamaguchi[2]

[1] Institute of Science and Technology, Niigata University
aoto@ie.niigata-u.ac.jp
[2] Graduate School of Information Sciences, Tohoku University
masaomi.yamaguchi.t4@dc.tohoku.ac.jp

A primary functionality of ACP is proving confluence of term rewriting systems (TRSs). ACP integrates multiple direct criteria for guaranteeing confluence of TRSs. It also incorporates divide–and–conquer criteria by which confluence or non-confluence of TRSs can be inferred from those of their components. Several methods for disproving confluence are also employed. For some criteria, it supports generation of proofs in CPF format that can be certified by certifiers. The internal structure of the prover is kept simple and is mostly inherited from the version 0.11a, which has been described in [2].

This year we have added a decision procedure of UNC for shallow TRSs. The decidability of UNC for shallow TRSs has been shown in [4]; our new efficient procedure and a correctness proof are reported in [5]. We have also added a functionality to deal with commutation problems. Our (dis)proofs of commutation are based on a development closed criterion [6] and a simple search for counter examples. Lastly, we have also added a confluence checking using ordered rewriting [3].

ACP is written in Standard ML of New Jersey (SML/NJ) and the source code is also available from [1]. It uses a SAT prover such as MiniSAT and an SMT prover YICES as external provers. It internally contains an automated (relative) termination prover for TRSs but external (relative) termination provers can be substituted optionally. Users can specify criteria to be used so that each criterion or any combination of them can be tested. Several levels of verbosity are available for the output so that users can investigate details of the employed approximations for each criterion or can get only the final result of prover's attempt.

References

[1] ACP (Automated Confluence Prover). http://www.nue.ie.niigata-u.ac.jp/tools/acp/.

[2] T. Aoto, J. Yoshida, and Y. Toyama. Proving confluence of term rewriting system automatically. In *Proc. of 20th RTA*, volume 5595 of *LNCS*, pages 93–102. Springer-Verlag, 2009.

[3] U. Martin and T. Nipkow. Ordered rewriting and confluence. In *Proc. 10th CADE*, volume 449 of *LNAI*, pages 366–380. Springer-Verlag, 1990.

[4] N. R. Radcliffe, L. F. T. Moreas, and R. M. Verma. Uniqueness of normal forms for shallow term rewrite systems. *ACM Transactions on Computational Logic*, 18(2):17:1–17:20, 2017.

[5] M. Yamaguchi. An efficient decision procedure of the UN property for shallow term rewriting systems. Bachlor's thesis, Niigata University, 2019.

[6] J. Yoshida, T. Aoto, and Y. Toyama. Automating confluence check of term rewriting systems. *Computer Software*, 26(2):76–92, 2009.