# ACP: System Description for CoCo 2013

Takahito Aoto
RIEC
Tohoku University
aoto@nue.riec.tohoku.ac.jp

Yoshihito Toyama
RIEC
Tohoku University
toyama@nue.riec.tohoku.ac.jp

ACP is an automated confluence prover for term rewriting systems (TRSs) that has been developed in Toyama–Aoto group in RIEC, Tohoku University. This description is based on version 0.40. ACP integrates multiple direct criteria for guaranteeing confluence of TRSs. It 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 a TRS to which direct confluence criteria do not apply, the prover decomposes it into components using divide–and–conquer criteria, and tries to apply direct confluence criteria to each component. Then the prover combines these results to infer the (non-)confluence of the whole system.

ACP is written in Standard ML of New Jersey (SML/NJ) and is provided as a heap image that can be loaded into SML/NJ runtime systems. 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. The input TRS is specified in the (old) TPDB format. 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. The source code and a list of implemented criteria are found on the webpage [1].

The internal structure of the prover is kept simple and is mostly inherited from the version 0.11a, which has been described in [3]. Compared to the previous version, confluence criterion for strongly weight-preserving and root-E-closed term rewriting systems [5], the one for weakly-non-overlapping non-collapsing shallow term rewriting systems [7] and the one for quasi-left-linear term rewriting systems [8] are newly incorporated. The part of disproving confluence has been rewritten and confluence disproving methods applying tree automata (growing) approximation [4, 6] and based on interpretation and ordering [2] are incorporated.

## References

[1] ACP (Automated Confluence Prover). http://www.nue.riec.tohoku.ac.jp/tools/acp/.

[2] T. Aoto. Disproving confluence of term rewriting systems by interpretation and ordering (extended abstract). In *Proc. of 2nd IWC*, 2013. Full version is accepted for FroCoS 2013.

[3] 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.

[4] I. Durand and A. Middeldorp. Decidable call by need computations in term rewriting. In *Proc. of 14th CADE*, volume 1249 of *LNAI*, pages 4–18. Springer-Verlag, 1997.

[5] H. Gomi, M. Oyamaguchi, and Y. Ohta. On the Church-Rosser property of root-E-overlapping and strongly depth-preserving term rewriting systems. *Transactions of IPSJ*, 39(4):992–1005, 1998.

[6] F. Jacquemard. Decidable approximations of term rewriting systems. In *Proc. of 7th RTA*, volume 1103 of *LNCS*, pages 362–376. Springer-Verlag, 1996.

[7] M. Sakai and M. Ogawa. Weakly-non-overlapping non-collapsing shallow term rewriting systems are confluent. *IPL*, 110:810–814, 2010.

[8] T. Suzuki, T. Aoto, and Y. Toyama. Confluence proofs of term rewriting systems based on persistency (in japanese). *Computer Software*. To appear.