CoCo 2024 Participant: crest 0.8*

Jonas Schöpf and Aart Middeldorp

Department of Computer Science, University of Innsbruck, Innsbruck, Austria
{jonas.schoepf,aart.middeldorp}@uibk.ac.at

The Constrained REwriting Software Tool (crest, for short) is a tool for automatically proving (non-)confluence of logically constrained rewrite systems (LCTRSs). LCTRSs are an extension of term rewrite systems (TRSs) where rules are equipped with constraints. These must be satisfied in order for a rule to be applied. Analysis methods for plain TRSs cannot be applied in this setting, hence special techniques, adapted for LCTRSs, are used.

The development of crest started in the beginning of 2023 as part of the ARI project[1] and initial experiments were presented at CADE-29 in 2023 [3]. More information and executables of crest can be found at

http://cl-informatik.uibk.ac.at/software/crest/

Currently, crest supports (non-)confluence [2, 3, 4] and limited (non-)termination analysis [2, 1]. The confluence methods mostly rely on closure properties of (parallel) critical pairs. In particular confluence analysis via (weak) orthogonality [2], strong closedness [3], (almost) parallel closedness [3], (almost) development closedness [4], closure of parallel critical pairs [4] or joinable critical pairs of terminating LCTRSs [5] is implemented. Moreover, non-confluence is shown by finding a witness for two different normal forms originating from the same peak.

Input problems in the new ARI format are accepted.[2] However, crest extends the format slightly, e.g. by allowing theory symbols, which are only available in the SMT-LIB logic of quantifier-free bit-vector arithmetic.[3] Also most variable sort annotations, which are mandatory in the LCTRS format, can be omitted due to the implemented type inference algorithm. Reasoning about the theory part of an LCTRS is utilized via the SMT solver Z3.[4]

Our tool crest participates in the LCTRS category of CoCo 2024.

References

[1] Cynthia Kop. Termination of LCTRSs. *CoRR*, abs/1601.03206, 2016. doi: https://doi.org/10.48550/ARXIV.1601.03206.

[2] Cynthia Kop and Naoki Nishida. Term rewriting with logical constraints. In Pascal Fontaine, Christophe Ringeissen, and Renate A. Schmidt, editors, *Proc. 9th International Symposium on Frontiers of Combining Systems*, volume 8152 of *Lecture Notes in Artificial Intelligence*, pages 343–358, 2013. doi: https://doi.org/10.1007/978-3-642-40885-4_24.

[3] Jonas Schöpf and Aart Middeldorp. Confluence criteria for logically constrained rewrite systems. In Brigitte Pientka and Cesare Tinelli, editors, *Proc. 29th International Conference on Automated Deduction*, volume 14132 of *Lecture Notes in Artificial Intelligence*, pages 474–490, 2023. doi: https://doi.org/10.1007/978-3-031-38499-8_27.

*This research is funded by the Austrian Science Fund (FWF) project I 5943-N.

[1] https://ari-informatik.uibk.ac.at/

[2] https://project-coco.uibk.ac.at/ARI/lctrs.php

[3] https://smt-lib.org/logics-all.shtml#QF_BV

[4] https://www.microsoft.com/en-us/research/project/z3-3/

[4] Jonas Schöpf, Fabian Mitterwallner, and Aart Middeldorp. Confluence of logically constrained rewrite systems revisited. *CoRR*, abs/2402.13552, 2024. doi: https://doi.org/10.48550/ARXIV.2402.13552.

[5] Sarah Winkler and Aart Middeldorp. Completion for logically constrained rewriting. In Hélène Kirchner, editor, *Proc. 3rd International Conference on Formal Structures for Computation and Deduction*, volume 108 of *Leibniz International Proceedings in Informatics*, pages 30:1–30:18, 2018. doi: https://doi.org/10.4230/LIPIcs.FSCD.2018.30.