# AProVE'26: Confluence Analysis in a Termination Tool

J.-C. Kassing[1,2], A. Schüßler[1,3], T. Sokolowski[1,4], and J. Giesl[1,5]

[1] RWTH Aachen University, Aachen, Germany
[2] kassing@cs.rwth-aachen.de
[3] alwin.schuessler@rwth-aachen.de
[4] tobias.sokolowski@rwth-aachen.de
[5] giesl@informatik.rwth-aachen.de

AProVE (Automated Program Verification Environment) is a tool for fully automatic program verification. Its primary focus is on proving termination, analyzing (worst-case) complexity, and verifying safety or infeasibility for a variety of programming languages, including term rewriting. For further details on AProVE's general approach to these analyses, see [4].

Termination and confluence are two closely related properties. On the one hand, local confluence lets us reduce the termination analysis of overlay systems from arbitrary rewrite sequences to innermost ones, which has been shown to be substantially easier. On the other hand, confluence becomes decidable once termination is guaranteed. This interplay makes AProVE's powerful termination analysis a natural foundation for confluence analysis.

AProVE has participated in the annual Confluence Competition in 2025 for the first time. At first, it supported only a small set of techniques that had originally been implemented for the termination analysis of term rewrite systems. Since its initial participation, however, we have added more techniques to both prove and disprove confluence. This year, we mostly integrated techniques based on polynomial interpretations, since such interpretations already have been implemented in AProVE, due to their usage in termination analysis.

AProVE relies on the following techniques, where those labeled with ⋆ have been newly implemented since last year's competition:

- *Termination-based Confluence Analysis*: We use termination analysis to check whether the well-known decision procedure for confluence of terminating systems is applicable.
- *Modularity*: We exploit classical results on the modularity of confluence. More precisely, we implemented several of the modularity criteria presented in [3] and [6].
- ⋆ *Decreasing Diagrams Based on Polynomial Interpretations*: Polynomial interpretations can be used to automate the decreasing diagrams technique of [7] via [1].
- ⋆ *Utilization of Redundant Rewrite Rules*: Rules that can be simulated by other rules are called redundant, see [5]. Confluence analysis can benefit from both adding and removing such redundant rules.
- *Disproving via Dependency Graph Approximations*: To disprove joinability of critical pairs, AProVE reuses techniques originally designed for proving infeasibility in dependency graph approximations, e.g., checking unifiability between the target term and an approximation of the top part of the source term that is preserved during rewrite steps.
- ⋆ *Disproving Confluence via Polynomial Interpretations*: Polynomial interpretations can not only be used to prove confluence but also to disprove confluence. We mostly rely on the techniques described in [2].

# References

[1] Takahito Aoto. Automated Confluence Proof by Decreasing Diagrams based on Rule-Labelling. In *Proc. RTA'10*, LIPIcs 6, pages 7–16, 2010.

[2] Takahito Aoto. Disproving Confluence of Term Rewriting Systems by Interpretation and Ordering. In *Proc. FroCoS'13*, LNCS 8152, pages 311–326, 2013.

[3] Franz Baader and Tobias Nipkow. *Term Rewriting and All That.* Cambridge University Press, Cambridge, 1998.

[4] Jürgen Giesl, Cornelius Aschermann, Marc Brockschmidt, Fabian Emmes, Florian Frohn, Carsten Fuhs, Jera Hensel, Carsten Otto, Martin Plücker, Peter Schneider-Kamp, Thomas Ströder, Stephanie Swiderski, and René Thiemann. Analyzing Program Termination and Complexity Automatically with AProVE. *J. Autom. Reason.*, 58(1):3–31, 2017.

[5] Julian Nagele, Bertram Felgenhauer, and Aart Middeldorp. Improving Automatic Confluence Analysis of Rewrite Systems by Redundant Rules. In *Proc. RTA'15*, LIPIcs 36, pages 257–268, 2015.

[6] Enno Ohlebusch. *Advanced Topics in Term Rewriting.* Springer, New York, NY, 2002.

[7] Vincent van Oostrom. Confluence by Decreasing Diagrams. *Theoretical Computer Science*, 126:259–280, 1994.