AProVE25: Confluence Analysis in a Termination Tool

Jan-Christoph Kassing[1] and Tobias Sokolowski[2]

[1] RWTH Aachen University, Aachen, Germany
Kassing@cs.rwth-aachen.de
[2] RWTH Aachen University, Aachen, Germany
Tobias.Sokolowski@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 of different programming languages including term rewriting. For further details on AProVE's general approach for these analyses, see [3].

Termination and confluence are two closely related properties. Local confluence allows us to reduce the analysis of termination from arbitrary rewrite sequences to innermost rewrite sequences, a task that has been shown to be substantially easier. Moreover, the dependency graph heavily relies on proving infeasibility so that better computable approximations may yield a more exact model of the actual dependency graph. On the other hand, confluence is a decidable property if termination is guaranteed. Therefore, AProVE has already implemented some techniques for confluence, and we want to present the power of the currently implemented methods and, in the future, to improve confluence and reachability analysis within AProVE.

AProVE relies on three main techniques:

- *Termination-based Confluence Analysis*: We use our termination analysis to check whether the well-known decision procedure for confluence is applicable.
- *Modularity*: We use basic results on modularity of confluence from the last century. To be precise, we implemented different modularity results mentioned in [1] and [4].
- *Joinability and Reachability Analysis*: To disprove joinability of critical pairs, AProVE uses techniques originally designed for proving infeasibility within dependency graph approximations, e.g., checking for unifiability between the target term and an approximation of the top part of the source term that remains the same during rewrite steps.

In the future, we want to investigate the problem of confluence within the probabilistic setting, a question raised in recent years and investigated in, e.g., [2], but which has received relatively little attention. Due to the complex interplay of probabilities and non-deterministic choices, we believe that this is an interesting direction for the confluence community.

References

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

[2] Alejandro Díaz-Caro and Guido Martínez. Confluence in Probabilistic Rewriting. *Electronic Notes in Theoretical Computer Science*, 338:115–131, October 2018.

[3] 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.

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