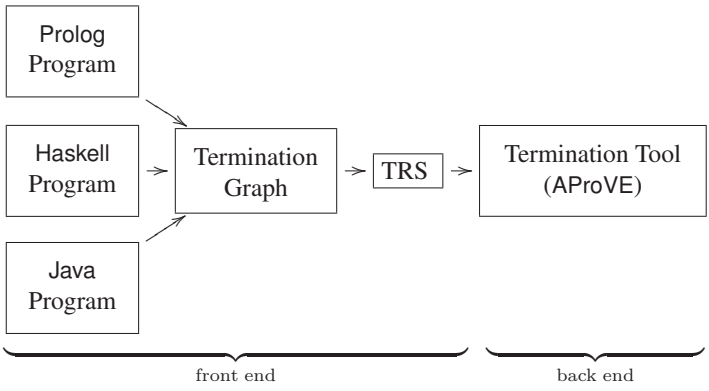
Automated Termination Analysis: From Term Rewriting to Programming Languages

Jürgen Giesl

LuFG Informatik 2
RWTH Aachen
Ahornstr. 55
52074 Aachen, Germany
giesl@informatik.rwth-aachen.de

Termination is a crucial property of programs. Therefore, techniques to analyze termination automatically are highly important for program verification. Traditionally, techniques for automated termination analysis were mainly studied for declarative programming paradigms such as logic programming and term rewriting. However, in the last years, several powerful techniques and tools have been developed which analyze the termination of programs in many programming languages including Java, C, Haskell, and Prolog.

In order to re-use the wealth of existing tools and techniques developed for termination analysis of term rewriting (see e.g., [GTSKF06, Zan03]), we developed a *transformational* methodology to prove termination of programs in different languages. In a *front end*, the program is automatically transformed into a term rewrite system (TRS) such that termination of the TRS implies termination of the original program. To obtain TRSs which are suitable for automated termination proofs, the front end proceeds in two steps. In the first step, the program is executed symbolically to generate a so-called *termination graph*. This graph represents all possible evaluations of the program in a finite way. In the second step, the edges of the graph are transformed to rewrite rules. Finally, existing rewriting techniques are used in the *back end* to prove termination of the resulting TRS.



We developed such approaches to prove termination of Prolog [GSSK+12, SKGS+10], Haskell [GRSK+11], and Java [BMOG12, BOG11, BOvG10, BSOG12, OBvG10], and integrated them into our termination tool AProVE [GST06]. As shown at the annual *International Termination Competition*,[1] AProVE is currently not only the most powerful tool for automated termination analysis of TRSs, but also for Prolog, Haskell, and Java. This shows that the proposed methodology for rewrite-based automated termination analysis indeed leads to competitive results.

References

[BMOG12] M. Brockschmidt, R. Musiol, C. Otto, and J. Giesl. Automated Termination Proofs for Java Programs with Cyclic Data. In *Proc. CAV '12*, LNCS 7358, pages 105–122, 2012.

[BOG11] M. Brockschmidt, C. Otto, and J. Giesl. Modular Termination Proofs of Recursive Java Bytecode Programs by Term Rewriting. In *Proc. RTA '11*, LIPIcs 10, pages 155–170, 2011.

[BOvG10] M. Brockschmidt, C. Otto, C. von Essen, and J. Giesl. Termination Graphs for Java Bytecode. In *Verification, Induction, Termination Analysis*, LNCS 6463, pages 17–37, 2010.

[BSOG12] M. Brockschmidt, T. Ströder, C. Otto, and J. Giesl. Automated Detection of Non-Termination and `NullPointerExceptions` for Java Bytecode. In *Proc. FoVeOOS '11*, LNCS 7421, pages 123–141, 2012.

[GRSK+11] J. Giesl, M. Raffelsieper, P. Schneider-Kamp, S. Swiderski, and R. Thiemann. Automated Termination Proofs for Haskell by Term Rewriting. *ACM TOPLAS*, 33(2):7:1–7:39, 2011.

[GSSK+12] J. Giesl, T. Ströder, P. Schneider-Kamp, F. Emmes, and C. Fuhs. Symbolic Evaluation Graphs and Term Rewriting – A General Methodology for Analyzing Logic Programs. In *Proc. PPDP '12*, pages 1–12. ACM Press, 2012.

[GST06] J. Giesl, P. Schneider-Kamp, and R. Thiemann. AProVE 1.2: Automatic Termination Proofs in the Dependency Pair Framework. In *Proc. IJCAR '06*, LNAI 4130, pages 281–286, 2006.

[GTSKF06] J. Giesl, R. Thiemann, P. Schneider-Kamp, and S. Falke. Mechanizing and Improving Dependency Pairs. *Journal of Automated Reasoning*, 37(3):155–203, 2006.

[OBvG10] C. Otto, M. Brockschmidt, C. von Essen, and J. Giesl. Automated Termination Analysis of Java Bytecode by Term Rewriting. In *Proc. RTA '10*, LIPIcs 6, pages 259–276, 2010.

[SKGS+10] P. Schneider-Kamp, J. Giesl, T. Ströder, A. Serebrenik, and R. Thiemann. Automated Termination Analysis for Logic Programs with Cut. *Theory and Practice of Logic Programming*, 10(4-6):365–381, 2010.

[Zan03] H. Zantema. Termination. In Terese, editor, *Term Rewriting Systems*, pages 181–259. Cambridge University Press, 2003.

[1] See `http://termination-portal.org/wiki/Termination_Competition`