Dependency Pairs for Expected Runtime Complexity of Probabilistic Term Rewriting*

Jan-Christoph Kassing
RWTH Aachen University, Aachen, Germany

Leon Valentin Spitzer
RWTH Aachen University, Aachen, Germany

Jürgen Giesl
RWTH Aachen University, Aachen, Germany

Probabilistic term rewrite systems (PTRSs) have been introduced in [3,4]. A PTRS is *almost-surely terminating* (AST) if every evaluation (or "reduction") terminates with probability 1. A strictly stronger notion is *positive* AST (PAST), where every reduction must consist of a finite expected number of rewrite steps. An even stronger notion is *strong* AST (SAST) which requires that for every term t, the supremum over the expected number of rewrite steps of all possible reductions starting in t is finite. It is well known that SAST implies PAST and that PAST implies AST.

Currently, the only approach to analyze SAST of PTRSs automatically is the direct application of polynomial or matrix interpretations to the whole PTRS [3]. However, already for non-probabilistic term rewrite systems, such a direct application of orderings is limited in power. For a powerful approach, one should combine orderings in a modular way, as in the *dependency pair* (DP) framework, which is one of the most powerful approaches to analyze termination and runtime complexity of term rewrite systems, see, e.g., [1,2,5,7,10].

Therefore, we already adapted the DP framework to the probabilistic setting in order to prove AST, both for innermost and full rewriting [9]. Moreover, in the non-probabilistic setting, DPs were extended to analyze complexity instead of just termination, see, e.g., [2,10]. But up to now there did not exist any DP framework to prove SAST or PAST, or to infer bounds on the expected runtime of PTRSs.

Therefore, we develop the first DP framework for SAST and expected innermost runtime complexity of PTRSs by lifting the DP framework for AST from [9] accordingly. To evaluate the our novel framework, we implemented it in the tool AProVE [6] and demonstrate its power compared to existing techniques for proving SAST.

Related Version Extended abstract of our PPDP '25 paper [8]

References

1 Thomas Arts and Jürgen Giesl. Termination of term rewriting using dependency pairs. *Theor. Comput. Sc.*, 236(1-2):133–178, 2000. doi:10.1016/S0304-3975(99)00207-8.

2 Martin Avanzini and Georg Moser. A combination framework for complexity. *Inf. Comput.*, 248:22–55, 2016. doi:10.1016/j.ic.2015.12.007.

3 Martin Avanzini, Ugo Dal Lago, and Akihisa Yamada. On probabilistic term rewriting. *Sci. Comput. Program.*, 185, 2020. doi:10.1016/j.scico.2019.102338.

4 Olivier Bournez and Florent Garnier. Proving positive almost-sure termination. In *Proc. RTA '05*, LNCS 3467, pages 323–337, 2005. doi:10.1007/978-3-540-32033-3_24.

5 Jürgen Giesl, René Thiemann, Peter Schneider-Kamp, and Stephan Falke. Mechanizing and improving dependency pairs. *J. Autom. Reason.*, 37(3):155–203, 2006. doi:10.1007/s10817-006-9057-7.

* funded by the DFG Research Training Group 2236 UnRAVeL

6 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. doi:10.1007/s10817-016-9388-y.

7 Nao Hirokawa and Aart Middeldorp. Automating the dependency pair method. *Inf. Comput.*, 199(1-2):172–199, 2005. doi:10.1016/j.ic.2004.10.004.

8 Jan-Christoph Kassing, Leon Valentin Spitzer, and Jürgen Giesl. Dependency pairs for expected innermost runtime complexity and strong almost-sure termination of probabilistic term rewriting. In *Proc. PPDP '25*, 2025. doi:10.1145/3756907.3756917.

9 Jan-Christoph Kassing and Jürgen Giesl. The annotated dependency pair framework for almost-sure termination of probabilistic term rewriting. *Sci. Comput. Program.*, 251, 2026. doi:10.1016/J.SCICO.2025.103417.

10 Lars Noschinski, Fabian Emmes, and Jürgen Giesl. Analyzing innermost runtime complexity of term rewriting by dependency pairs. *J. Autom. Reason.*, 51:27–56, 2013. doi:10.1007/s10817-013-9277-6.