# TERM REWRITING: BASIC CONCEPTS, TOOLS, AND APPLICATIONS

SARAH WINKLER

Term rewriting is a simple but powerful model of computation which has numerous applications in computer science, mathematics, logic, programming languages, and biology. This lecture series introduces some foundational notions of first-order term rewriting: termination, confluence, completion, and complexity. For each of these properties, basic techniques to establish them are discussed [1, 4, 3]. The practical relevance of these concepts is illustrated by examples from different fields:

- **Termination** of a rewrite system means the absence of infinite rewrite sequences. Interpretations and reduction orders are presented as termination techniques. Termination of programs or expression simplification (e.g. in compilers) can be seen as the respective property of a rewrite system [6], and reduction orders play a key role in theorem proving [5].
- The **derivational complexity** of a terminating rewrite system, i.e., the maximal number of steps to a result, is often relevant for theoretical considerations; and important to reason about programs in performance-critical contexts [2]. We discuss matrix interpretations as a method to establish complexity bounds, and mention bounds imposed by other termination methods.
- **Confluence** expresses a kind of determinism of the rewriting process. We discuss critical pairs and orthogonality as confluence techniques. As applications, we mention determinism of simple games [8] and expression simplification [7].
- **Completion** can transform a set of equations into an equivalent terminating and confluent rewrite system. If time permits, a simple completion procedure will be presented, which will be shown to constitute a theorem proving method.

Automatic tools to determine these properties will be demonstrated, too; also to give an impression of the vast number of available techniques beyond the scope of this course. Parts of the lecture will be dedicated to solve exercises (with and without tools) to get hands-on experience.

## References

[1] F. Baader and T. Nipkow. *Term Rewriting and All That*. Cambridge University Press, 1998. 10.1017/CBO9781139172752.

[2] M. Brockschmidt, F. Emmes, S. Falke, C. Fuhs, and J. Giesl: Analyzing Runtime and Size Complexity of Integer Programs. In *ACM Transactions on Programming Languages and Systems*. 38(4): 13:1-13:50 (2016). 10.1145/2866575

[3] D. Hofbauer and C. Lautemann. Termination proofs and the length of derivations. In *Proc. 3rd RTA*, vol. 355 of LNCS, pp. 167–177, 1989. 10.1007/3-540-51081-8_107.

[4] G. Moser, A. Schnabl and J. Waldmann. Complexity analysis of term rewriting based on matrix and context dependent interpretations. In *Proc. FSTTCS 2008*, vol. 2 of LIPIcs, pp. 304–315, 2008. 10.4230/LIPIcs.FSTTCS.2008.1762.

[5] R. Nieuwenhuis and A. Rubio. Paramodulation-Based Theorem Proving, In *Handbook of Automated Reasoning*. North Holland, 2001. 10.1016/B978-044450813-3/50009-6.

[6] N. Nishida and S. Winkler. Loop Detection by Logically Constrained Term Rewriting, In *Proc. 10th VSTTE*, vol. 11294 of LNCS, pp. 309–321, 2018. 10.1007/978-3-030-03592-1_18.

[7] S. Winkler and A. Middeldorp. Completion for logically constrained rewriting. In *Proc. 3rd FSCD*, vol. 108 of LIPIcs, pp. 30:1–30:18, 2018. 10.4230/LIPIcs.FSCD.2018.30.

[8] S. Winkler and A. Middeldorp. Tools in Term Rewriting for Education. *EPTCS*, to appear. Available from http://profs.scienze.univr.it/winkler/papers/SWAM-ThEdu19.pdf.

Università di Verona, Italy
*E-mail address*: sarahmaria.winkler@univr.it