Termination of Term Rewriting: Foundation, Formalization, Implementation, and Competition

Akihisa Yamada ✉
National Institute of Advanced Industrial Science and Technology, Tokyo, Japan

Abstract

Automated termination analysis is a central topic in the research of term rewriting. In this talk, I will first review the theoretical foundation of termination of term rewriting, leading to the recently established tuple interpretation method. Then I will present an Isabelle/HOL formalization of the theory. Although the formalization is based on the existing library IsaFoR (Isabelle Formalization of Rewriting), the present work takes another approach of representing relations (predicates rather than sets) so that the notation is more human friendly. Then I will present a unified implementation of the termination analysis techniques via SMT encoding, leading to the termination prover NaTT. Many tools have been developed for termination analysis and have been competing annually in termCOMP (Termination Competition) for two decades. At the end of the talk, I will share my experience in organizing termCOMP in the last five years.

2012 ACM Subject Classification Theory of computation → Rewrite systems; Theory of computation → Automated reasoning

Keywords and phrases Term rewriting, Termination, Isabelle/HOL, Competition

Digital Object Identifier 10.4230/LIPIcs.FSCD.2023.4

Category Invited Talk

1 Foundation

Ensuring the termination [10] of term rewrite systems (TRSs) has many important applications and actively studied for a half century. A foundational way of ensuring termination used to be *reduction orders*, in short, overestimating the rewrite step by a term ordering whose termination (well-foundedness) is known. Later, the dependency pair method [1] generalized reduction orders to *reduction pairs* [1, 17, 15], where, in short, one part overestimates the rewrite step and the other part ensures termination.

The semantic method (interpretation method) [29] for defining reduction orders/pairs uses universal algebras, mapping terms into a set with a known terminating relation, such as $\langle \mathbb{N}, > \rangle$. Instances of semantic approaches are polynomial interpretations [27], matrix interpretations [18, 11], max-polynomial interpretations [12], and arctic interpretations [24]. Tuple interpretations [43, 23] encompass these instances, and the use of *derivers* [41, 28] allows to uniformly prove monotonicity (essential for overestimating rewrite steps) [43].

The syntactic approach such as recursive path ordering [9] and lexicographic path ordering [21] analyzes the term structure purely syntactically, and usually ensure termination by Kruskal's tree theorem [26]. The Knuth-Bendix ordering [22] mixes the semantic and syntactic approaches. The weighted path order (WPO) [45] gives a unified way of capturing syntactic and semantic methods.

2 Formalization

There are number of methods for ensuring termination, and they are implemented in automated termination analysis tools such as AProVE [13], TTT2 [25], matchbox [40], MU-TERM [16], etc. However, it cannot be taken for granted that the implementations and the paper proofs are correct. Therefore there have been efforts in formally verifying the paper proofs and tool outputs using proof assistants such as Coq [4] and Isabelle/HOL [30].

Coccinelle/CiME3 [8] and CoLoR/Rainbow [5] are pairs whose first components are Coq libraries of proofs for termination methods, and the second components are tools that translate termination tool's outputs into Coq theories whose validity are then checked by Coq. IsaFoR/CeTA [38] is another pair, whose first component is an Isabelle/HOL library of termination methods (and more), and the second is a tool that checks the tool outputs. The correctness of the latter is verified by Isabelle; therefore achieves the same level of assurance as the Coq ones without needing to execute the proof assistants in the run time. In this talk I will present a recent development based on IsaFoR for sorted term rewriting and algebras, which formalizes the correctness of the deriver-based tuple interpretation method.

3 Implementation

NaTT started as an OCaml [31] code for experimenting WPO. This implementation follows the "lazy" approach [47, 32, 48, 7] and delegates "solving" tasks to SMT solvers [3]. The code was turned into a full termination prover by adding other basic techniques and improving efficiency by being more lazy in calling SMT solvers [44]: for instance, in the SMT interface of NaTT, the OCaml code `x *^ Delay (fun _ -> hardFormula)` represents an SMT formula for x multiplied by *hardFormula*, but *hardFormula* will not be evaluated if $x = 0$ is known. NaTT participates in the Termination Competition (since "full run" of 2013) and keeps winning the second place in the TRS Standard category after the champion, AProVE. Since then more techniques are implemented into NaTT [19, 46, 2, 35, 43, 42], making the implementation harder and harder to maintain. In the meantime, the C++ programming language kept evolving, and hence I fully reimplement NaTT in the modern C++20 [20]. For instance, in the new C++ implementation, the above lazy formula is achieved by `x * []{ return hardFormula; }`.

4 Competition

At the International Workshop on Termination (WST) 2003, Albert Rubio organized a session where termination tool developers demonstrate their tools on problems written on a blackboard. The community decided to turn this event into the International Termination Competition (termCOMP) [14]. Benchmarks are collected, using an agreed textual format called the WST format [6], into the Termination Problem DataBase (TPDB) [39]. Editions from 2004 to 2007 were run on local servers organized by Claude Marché, and from 2008 to 2013 are by René Thiemann. In 2009, the benchmark format has been changed to an XML format (see [39]). From 2014 to 2017 are organized by Johannes Waldmann, who migrated the tool execution platform to the StarExec environment [36], and wrote the web server code, star-exec-presenter [33], using Haskell and SQL. In 2018 I succeeded the organization and developed starexec-master [34] web frontend in pure PHP (so that most web servers and engineers can understand). Since then I have gradually improve visuals and collected metrics (Figure 1 on page 3), keep past results in a public repository [37], and migrated TPDB also into GitHub. In the talk I would like to share some lessons learned by organizing the competition.

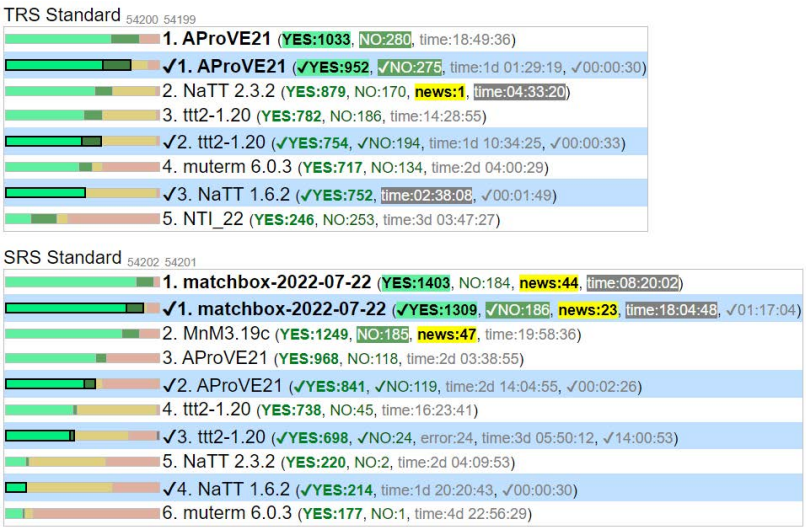

Figure 1 An excerpt of results of termCOMP 2022. Rows with check mark indicates the certified configurations. The "news" scores indicate improvements over the virtual best solver of the past, that is, the number of open problems closed by the solver this year. Those scores were used to award the special "advancing-the-state-of-the-art" medals.

References

1 T. Arts and J. Giesl. Termination of term rewriting using dependency pairs. *Theor. Compt. Sci.*, 236(1-2):133–178, 2000. doi:0.1016/S0304-3975(99)00207-8.

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

3 Clark W. Barrett and Cesare Tinelli. Satisfiability modulo theories. In Edmund M. Clarke, Thomas A. Henzinger, Helmut Veith, and Roderick Bloem, editors, *Handbook of Model Checking*, pages 305–343. Springer, 2018. doi:10.1007/978-3-319-10575-8_11.

4 Yves Bertot and Pierre Castéran. *Interactive theorem proving and program development: Coq'Art: the calculus of inductive constructions*. Springer Science & Business Media, 2013.

5 Frédéric Blanqui and Adam Koprowski. CoLoR: a Coq library on well-founded rewrite relations and its application to the automated verification of termination certificates. *Math. Struct. Comput. Sci.*, 21(4):827–859, 2011. doi:10.1017/S0960129511000120.

6 H. Zantema C. Marché, A. Rubio. Termination problem data base: format of input files, 2005. URL: https://www.lri.fr/~marche/tpdb/format.html.

7 M. Codish, J. Giesl, P. Schneider-Kamp, and R. Thiemann. SAT solving for termination proofs with recursive path orders and dependency pairs. *J. Autom. Reasoning*, 49(1):53–93, 2012.

8 Evelyne Contejean, Pierre Courtieu, Julien Forest, Olivier Pons, and Xavier Urbain. Automated certified proofs with cime3. In Manfred Schmidt-Schauß, editor, *RTA 2011*, volume 10 of *LIPIcs*, pages 21–30. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2011. doi:10.4230/LIPIcs.RTA.2011.21.

9 Nachum Dershowitz. Orderings for term-rewriting systems. *Theor. Compt. Sci.*, 17(3):279–301, 1982. doi:10.1016/0304-3975(82)90026-3.

10 Nachum Dershowitz. Termination of rewriting. *J. Symb. Comput.*, 3(1–2):69–115, 1987.

11 Jörg Endrullis, Johannes Waldmann, and Hans Zantema. Matrix interpretations for proving termination of term rewriting. *J. Autom. Reason.*, 40(2-3):195–220, 2008. doi:10.1007/s10817-007-9087-9.

12 Carsten Fuhs, Jürgen Giesl, Aart Middeldorp, Peter Schneider-Kamp, René Thiemann, and Harald Zankl. Maximal termination. In Andrei Voronkov, editor, *RTA 2008*, volume 5117 of *LNCS*, pages 110–125. Springer, 2008. doi:10.1007/978-3-540-70590-1_8.

13 Jürgen Giesl, Marc Brockschmidt, Fabian Emmes, Florian Frohn, Carsten Fuhs, Carsten Otto, Martin Plücker, Peter Schneider-Kamp, Thomas Ströder, Stephanie Swiderski, and René Thiemann. Proving termination of programs automatically with AProVE. In Stéphane Demri, Deepak Kapur, and Christoph Weidenbach, editors, *IJCAR 2014*, volume 8562 of *LNCS*, pages 184–191. Springer, 2014. doi:10.1007/978-3-319-08587-6_13.

14 Jürgen Giesl, Albert Rubio, Christian Sternagel, Johannes Waldmann, and Akihisa Yamada. The termination and complexity competition. In Dirk Beyer, Marieke Huisman, Fabrice Kordon, and Bernhard Steffen, editors, *TACAS 2019, Part III*, volume 11429 of *LNCS*, pages 156–166. Springer, 2019. doi:10.1007/978-3-030-17502-3_10.

15 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.

16 Raúl Gutiérrez and Salvador Lucas. mu-term: Verify termination properties automatically (system description). In Nicolas Peltier and Viorica Sofronie-Stokkermans, editors, *IJCAR 2020 (2)*, volume 12167 of *LNCS*, pages 436–447. Springer, 2020. doi:10.1007/978-3-030-51054-1_28.

17 Nao Hirokawa and Aart Middeldorp. Dependency pairs revisited. In Vincent van Oostrom, editor, *RTA 2004*, volume 3091 of *LNCS*, pages 249–268. Springer, 2004. doi:10.1007/978-3-540-25979-4_18.

18 Dieter Hofbauer and Johannes Waldmann. Termination of string rewriting with matrix interpretations. In Frank Pfenning, editor, *RTA 2006*, volume 4098 of *LNCS*, pages 328–342. Springer, 2006. doi:10.1007/11805618_25.

19 José Iborra, Naoki Nishida, Germán Vidal, and Akihisa Yamada. Relative termination via dependency pairs. *J. Autom. Reason.*, 58(3):391–411, 2017. doi:10.1007/s10817-016-9373-5.

20 Programming languages: C++, ISO/IEC 14882:2020.

21 Sam Kamin and Jean-Jacques Lévy. Two generalizations of the recursive path ordering, 1980. Unpublished note.

22 Donald E. Knuth and Peter Bendix. Simple word problems in universal algebras. In *Computational Problems in Abstract Algebra*, pages 263–297. Pergamon Press, New York, 1970. doi:10.1016/B978-0-08-012975-4.50028-X.

23 Cynthia Kop and Deivid Vale. Tuple interpretations for higher-order complexity. In Naoki Kobayashi, editor, *FSCD 2021*, volume 195 of *LIPIcs*, pages 31:1–31:22. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPIcs.FSCD.2021.31.

24 Adam Koprowski and Johannes Waldmann. Max/plus tree automata for termination of term rewriting. *Acta Cybern.*, 19(2):357–392, 2009.

25 Martin Korp, Christian Sternagel, Harald Zankl, and Aart Middeldorp. Tyrolean Termination Tool 2. In Ralf Treinen, editor, *RTA 2009*, volume 5595 of *LNCS*, pages 295–304. Springer, 2009. doi:10.1007/978-3-642-02348-4_21.

26 J.B. Kruskal. Well-quasi-ordering, the tree theorem, and Vazsonyia's conjecture. *Transactions of the American Mathematical Society*, 95(2):210–225, 1960.

27 D. Lankford. Canonical algebraic simplification in computational logic. Technical Report ATP-25, University of Texas, 1975.

28 Salvador Lucas and Raúl Gutiérrez. Automatic synthesis of logical models for order-sorted first-order theories. *J. Autom. Reason.*, 60(4):465–501, 2018. doi:10.1007/s10817-017-9419-3.

29 Z. Manna and S. Ness. On the termination of Markov algorithms. In *the 3rd Hawaii International Conference on System Science*, pages 789–792, 1970.

30 Tobias Nipkow, Lawrence C. Paulson, and Markus Wenzel. *Isabelle/HOL - A Proof Assistant for Higher-Order Logic*, volume 2283 of *LNCS*. Springer, 2002. doi:10.1007/3-540-45949-9.

31 Ocaml. URL: https://v2.ocaml.org/.

32 Peter Schneider-Kamp, René Thiemann, Elena Annov, Michael Codish, and Jürgen Giesl. Proving termination using recursive path orders and SAT solving. In Boris Konev and Frank Wolter, editors, *FroCoS 2007*, volume 4720 of *LNCS*, pages 267–282. Springer, 2007. doi:10.1007/978-3-540-74621-8_18.

33 star-exec-presenter (software), 2014. URL: https://github.com/jwaldmann/star-exec-presenter/.

34 starexec-master (software). URL: https://github.com/TermCOMP/starexec-master.

35 Christian Sternagel and Akihisa Yamada. Reachability analysis for termination and confluence of rewriting. In Tomás Vojnar and Lijun Zhang, editors, *TACAS (1) 2019*, volume 11427 of *LNCS*, pages 262–278. Springer, 2019. doi:10.1007/978-3-030-17462-0_15.

36 Aaron Stump, Geoff Sutcliffe, and Cesare Tinelli. StarExec: A cross-community infrastructure for logic solving. In Stéphane Demri, Deepak Kapur, and Christoph Weidenbach, editors, *IJCAR 2014*, volume 8562 of *LNCS*, pages 367–373. Springer, 2014. doi:10.1007/978-3-319-08587-6_28.

37 Termination competitions results (repository). URL: https://termcomp.github.io/.

38 René Thiemann and Christian Sternagel. Certification of termination proofs using CeTA. In Stefan Berghofer, Tobias Nipkow, Christian Urban, and Makarius Wenzel, editors, *TPHOLs 2009*, volume 5674 of *LNCS*, pages 452–468. Springer, 2009. doi:10.1007/978-3-642-03359-9_31.

39 The termination problem data base. URL: http://termination-portal.org/wiki/TPDB.

40 Johannes Waldmann. Matchbox: A tool for match-bounded string rewriting. In Vincent van Oostrom, editor, *RTA 2004*, volume 3091 of *LNCS*, pages 85–94. Springer, 2004. doi:10.1007/978-3-540-25979-4_6.

41 T.J. Watson, J.A. Goguen, J.W. Thatcher, and E.G. Wagner. An initial algebra approach to the specification, correctness, and implementation of abstract data types. In *Current Trends in Programming Methodology*. Prentice Hall, 1976.

42 Akihisa Yamada. Term orderings for non-reachability of (conditional) rewriting. In Jasmin Blanchette, Laura Kovács, and Dirk Pattinson, editors, *IJCAR 2022*, volume 13385 of *LNCS*, pages 248–267. Springer, 2022. doi:10.1007/978-3-031-10769-6_15.

43 Akihisa Yamada. Tuple interpretations for termination of term rewriting. *J. Autom. Reason.*, 66(4):667–688, 2022. doi:10.1007/s10817-022-09640-4.

44 Akihisa Yamada, Keiichirou Kusakari, and Toshiki Sakabe. Nagoya termination tool. In Gilles Dowek, editor, *RTA-TLCA 2014*, volume 8560 of *LNCS*, pages 466–475. Springer, 2014. doi:10.1007/978-3-319-08918-8_32.

45 Akihisa Yamada, Keiichirou Kusakari, and Toshiki Sakabe. A unified ordering for termination proving. *Sci. Comput. Program.*, 111:110–134, 2015. doi:10.1016/j.scico.2014.07.009.

46 Akihisa Yamada, Christian Sternagel, René Thiemann, and Keiichirou Kusakari. AC dependency pairs revisited. In Jean-Marc Talbot and Laurent Regnier, editors, *CSL 2016*, volume 62 of *LIPIcs*, pages 8:1–8:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2016. doi:10.4230/LIPIcs.CSL.2016.8.

47 Harald Zankl. *Lazy Termination Analysis*. PhD thesis, University of Innsbruck, 2009.

48 Harald Zankl, Nao Hirokawa, and Aart Middeldorp. KBO orientability. *J. Autom. Reasoning*, 43(2):173–201, 2009.