SAT Instances for Termination Analysis with AProVE*

Carsten Fuhs

LuFG Informatik 2, RWTH Aachen University, Germany
fuhs@informatik.rwth-aachen.de

Abstract. Recently, SAT solving has become the backbone for tackling the search problems in automated termination analysis for term rewrite systems and for programming languages. Indeed, even since the last SAT competition in 2007, many new termination techniques have been published where automation heavily relies on the efficiency of modern SAT solvers. Here, a successful satisfiability proof of the SAT instance results in a step in the modular termination proof and simplifies the termination problem to be analyzed.
The present SAT benchmark submission was created using the automated termination prover AProVE. The CNFs stem from termination proof steps using various recent termination techniques. All instances of this submission are satisfiable, and any speed-up for SAT solvers on these instances will directly lead to performance improvements also for automated termination provers.

1 Introduction

Termination is one of the most important properties of programs. Therefore, there is a need for suitable methods and tools to analyze the termination behavior of programs automatically. In particular, there has been intensive research on techniques for termination analysis of *term rewrite systems* (TRSs) [2]. Instead of developing many separate termination techniques for different programming languages, it is a promising approach to transform programs from different languages into TRSs instead. Then termination tools for TRSs can be used for termination analysis of many different programming languages, cf. e.g. [11,19,20].

The increasing interest in termination analysis for TRSs is also demonstrated by the *International Competition of Termination Tools*,[1] held annually since 2004. Here, each participating tool is applied to the examples from the *Termination Problem Data Base (TPDB)*[2] and gets 60 seconds per termination problem to prove or disprove termination. Thus, in order for a termination prover to be competitive, one needs efficient search techniques for finding termination (dis)proofs automatically.

* Description of benchmark instances submitted to the *SAT Competition 2009*.

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

[2] The current version 5.0.2 of this standard database for termination problems is available at http://dev.aspsimon.org/projects/termcomp/downloads/.

However, many of the arising search problems in automated terminating analysis for TRSs are NP-complete. Due to the impressive performance of modern SAT solvers, in recent years it has become common practice to tackle such problems by encoding them to SAT and by then applying a SAT solver on the resulting CNF. This way, performance improvements by orders of magnitude over existing dedicated search algorithms have been achieved, and also for new termination techniques, SAT solving is the method of choice for automation (cf. e.g. [3,4,5,6,7,8,9,14,16,17,21,23]).

Nowadays, techniques like the *Dependency Pair framework* [1,12,13,15] allow for *modular* termination proofs. This means that it is not necessary to show termination of a term rewriting system in a single proof step, but instead one can show termination of the different functions of the system *separately* and *incrementally*. In this setting, one can use SAT solving in such a way that a successful satisfiability proof of the encoded SAT instance results in an incremental step in the modular termination proof which allows to simplify the termination problem to be analyzed.

On the other hand, also speed-ups on unsatisfiable instances are beneficial for automated termination analysis. The faster one finds out that a particular termination technique does not succeed on a given termination problem (e.g., by a SAT solver returning UNSAT for an encoding of this technique for the termination problem), the more time is left to apply other techniques from the plethora of available termination analysis methods.

Nevertheless, this benchmark submission only contains satisfiable instances which contribute directly to successful termination proofs.

2 Benchmark Instances

The present SAT benchmark submission was created using the automated termination prover AProVE [10], which can be used to analyze the termination behavior of term rewriting systems, logic programs [20], and Haskell 98 programs [11].

AProVE was the most powerful termination prover for TRSs in all the termination competitions from 2004 – 2008. In AProVE, SAT encodings are performed in two stages:

1. First, the search problem is encoded into a propositional formula with arbitrary junctors. The formula is represented via a directed acyclic graph such that identical subformulas are shared.
2. Afterwards, this propositional formula is converted into an equisatisfiable formula in CNF. This is accomplished using SAT4J's [18] implementation of Tseitin's algorithm [22].

The submitted CNFs are named `AProVE09-`n`.dimacs`. For the analyzed termination problems from the TPDB, Fig. 1 provides details on the encoded termination technique and on the termination problem for each n.

Fig. 1. Details on the submitted SAT instances from TPDB problems

n	Encoded technique	Termination problem
01	Recursive Path Order [3,4,21]	`TRS/Cime/mucrl1.trs`
02	Recursive Path Order [3,4,21]	`TRS/TRCSR/inn/PALINDROME_complete_noand_C.trs`
03	Recursive Path Order [3,4,21]	`TRS/TRCSR/PALINDROME_complete_iGM.trs`
04	Matrix Order [5,16]	`SRS/secret06/matchbox/3.srs`
05	Matrix Order [5,16]	`SRS/Trafo/hom01.srs`
06	Matrix Order [5,16]	`SRS/Waldmann07b/size-12-alpha-3-num-535.srs`
07	Matrix Order [5,16]	`SRS/Zantema/z049.srs`
08	Matrix Order [5,16]	`SRS/Zantema/z053.srs`
09	Matrix Order [5,16]	`TRS/secret05/cime5.trs`
10	Polynomial Order [6]	`TRS/CSR_Maude/bool/RENAMED-BOOL_nokinds.trs`
11	Max-Polynomial Order [7]	`TRS/secret05/cime1.trs`
12	Max-Polynomial Order [7]	`TRS/Zantema/z09.trs`
13	Non-Monotonic Max-Pol. Order [7]	`TRS/aprove08/log.trs`
14	Rational Polynomial Order [9]	`SRS/Zantema/z117.srs`
15	Rational Polynomial Order [9]	`TRS/endrullis08/morse.trs`
16	Rational Polynomial Order [9]	`TRS/SchneiderKamp/trs/thiemann17.trs`
17	Rational Polynomial Order [9]	`TRS/TRCSR/inn/Ex49_GM04_C.trs`
18	Rational Polynomial Order [9]	`TRS/TRCSR/inn/Ex5_DLMMU04_C.trs`
19	Bounded Increase [14]	`TRS/SchneiderKamp/trs/cade14.trs`
20	Arctic Matrix Order [17]	`SRS/Endrullis/04.srs`
21	Arctic Matrix Order [17], alt. enc.	`SRS/Endrullis/04.srs`

For termination analysis, TRSs are a very suitable representation of algorithms on user-defined data structures. However, another main challenge in termination analysis of programs are algorithms on pre-defined data types like integers. Using standard representations of integers as terms leads to problems in efficiency and power for termination analysis with termination tools for TRSs.

Therefore, very recently we extended TRSs by built-in integers [8]. This combines the power of TRS techniques on user-defined data types with a powerful treatment of pre-defined integers. To automate the corresponding constraint-based termination techniques for this new formalism in AProVE, we again perform a reduction to SAT. For the empirical evaluation of these contributions, we collected a set of integer termination problems from the literature and from applications. This collection can be found on the web page of the evaluation at `http://aprove.informatik.rwth-aachen.de/eval/Integer/`.

Fig. 2 again provides details on the technique and on the analyzed problems.

3 Conclusion

SAT solving has become a key technology for automated termination provers. Thus, any improvements in efficiency of SAT solvers on the submitted SAT instances will also have a direct impact on efficiency and power of the respective termination tool.

Fig. 2. Details on the SAT instances from Integer TRSs

n	Encoded technique	Termination problem
22	Integer Max-Polynomial Order [8]	`Beerendonk/19.itrs`
23	Integer Max-Polynomial Order [8]	`CADE07/A14.itrs`
24	Integer Max-Polynomial Order [8]	`patrs/pasta/a.10.itrs`
25	Integer Max-Polynomial Order [8]	`VMCAI05/poly4.itrs`

References

1. Thomas Arts and Jürgen Giesl. Termination of term rewriting using dependency pairs. *Theoretical Computer Science*, 236(1-2):133–178, 2000.
2. Franz Baader and Tobias Nipkow. *Term Rewriting and All That*. Cambridge University Press, 1998.
3. Michael Codish, Vitaly Lagoon, and Peter J. Stuckey. Solving partial order constraints for LPO termination. *Journal on Satisfiability, Boolean Modeling and Computation*, 5:193–215, 2008.
4. Michael Codish, Peter Schneider-Kamp, Vitaly Lagoon, René Thiemann, and Jürgen Giesl. SAT solving for argument filterings. In *Proceedings of the 13th International Conference on Logic for Programming, Artificial Intelligence and Reasoning (LPAR 2006)*, volume 4246 of *LNAI*, pages 30–44, Phnom Penh, Cambodia, 2006.
5. Jörg Endrullis, Johannes Waldmann, and Hans Zantema. Matrix interpretations for proving termination of term rewriting. *Journal of Automated Reasoning*, 40(2-3):195–220, 2008.
6. Carsten Fuhs, Jürgen Giesl, Aart Middeldorp, René Thiemann, Peter Schneider-Kamp, and Harald Zankl. SAT Solving for Termination Analysis with Polynomial Interpretations. In *Proceedings of the 10th International Conference on Theory and Applications of Satisfiability Testing (SAT 2007)*, volume 4501 of *LNCS*, pages 340–354, Lisbon, Portugal, 2007.
7. Carsten Fuhs, Jürgen Giesl, Aart Middeldorp, René Thiemann, Peter Schneider-Kamp, and Harald Zankl. Maximal termination. In *Proceedings of the 19th International Conference on Rewriting Techniques and Applications (RTA 2008)*, volume 5117 of *LNCS*, pages 110–125, Hagenberg, Austria, 2008.
8. Carsten Fuhs, Jürgen Giesl, Martin Plücker, Peter Schneider-Kamp, and Stephan Falke. Proving termination of integer term rewriting. In *Proceedings of the 20th International Conference on Rewriting Techniques and Applications (RTA 2009)*, LNCS, Brasília, Brazil, 2009. To appear.
9. Carsten Fuhs, Rafael Navarro-Marset, Carsten Otto, Jürgen Giesl, Salvador Lucas, and Peter Schneider-Kamp. Search techniques for rational polynomial orders. In *Proceedings of the 9th International Conference on Artificial Intelligence and Symbolic Computation (AISC 2008)*, volume 5144 of *LNAI*, pages 109–124, Birmingham, UK, 2008.
10. Jürgen Giesl, Peter Schneider-Kamp, and René Thiemann. AProVE 1.2: Automatic termination proofs in the dependency pair framework. In *Proceedings of the 3rd International Joint Conference on Automated Reasoning (IJCAR 2006)*, volume

4130 of *LNAI*, pages 281–286, Seattle, WA, USA, 2006. See also http://aprove.informatik.rwth-aachen.de.

11. Jürgen Giesl, Stephan Swiderski, Peter Schneider-Kamp, and René Thiemann. Automated termination analysis for Haskell: From term rewriting to programming languages. In *Proceedings of the 17th International Conference on Rewriting Techniques and Applications (RTA 2006)*, volume 4098 of *LNCS*, pages 297–312, Seattle, WA, USA, 2006.
12. Jürgen Giesl, René Thiemann, and Peter Schneider-Kamp. The DP framework: Combining techniques for automated termination proofs. In *Proceedings of the 11th International Conference on Logic for Programming, Artificial Intelligence and Reasoning (LPAR 2004)*, volume 3452 of *LNAI*, pages 301–331, Montevideo, Uruguay, 2005.
13. Jürgen Giesl, René Thiemann, Peter Schneider-Kamp, and Stephan Falke. Mechanizing and improving dependency pairs. *Journal of Automated Reasoning*, 37(3):155–203, 2006.
14. Jürgen Giesl, René Thiemann, Stephan Swiderski, and Peter Schneider-Kamp. Proving termination by bounded increase. In *Proceedings of the 21st International Conference on Automated Deduction (CADE 2007)*, volume 4603 of *LNAI*, pages 443–459, Bremen, Germany, 2007.
15. Nao Hirokawa and Aart Middeldorp. Automating the dependency pair method. *Information and Computation*, 199(1-2):172–199, 2005.
16. Dieter Hofbauer and Johannes Waldmann. Termination of string rewriting with matrix interpretations. In *Proceedings of the 17th International Conference on Rewriting Techniques and Applications (RTA 2006)*, volume 4098 of *LNCS*, pages 328–342, Seattle, WA, USA, 2006.
17. Adam Koprowski and Johannes Waldmann. Arctic termination ... below zero. In *Proceedings of the 19th International Conference on Rewriting Techniques and Applications (RTA 2008)*, volume 5117 of *LNCS*, pages 202–216, Hagenberg, Austria, 2008.
18. Daniel Le Berre and Anne Parrain. SAT4J: The Java SAT Library. http://www.sat4j.org, 2009.
19. Enno Ohlebusch. Termination of logic programs: Transformational methods revisited. *Applicable Algebra in Engineering, Communication and Computing*, 12(1-2):73–116, 2001.
20. Peter Schneider-Kamp, Jürgen Giesl, Alexander Serebrenik, and René Thiemann. Automated termination proofs for logic programs by term rewriting. *ACM Transactions on Computational Logic*. To appear.
21. Peter Schneider-Kamp, René Thiemann, Elena Annov, Michael Codish, and Jürgen Giesl. Proving termination using recursive path orders and SAT solving. In *Proceedings of the 6th International Symposium on Frontiers of Combining Systems (FroCoS 2007)*, volume 4720 of *LNAI*, pages 267–282, Liverpool, UK, 2007.
22. Gregory Tseitin. On the complexity of derivation in propositional calculus. In *Studies in Constructive Mathematics and Mathematical Logic*, pages 115–125. 1968. Reprinted in *Automation of Reasoning*, volume 2, pages 466–483, Springer, 1983.
23. Harald Zankl and Aart Middeldorp. Satisfying KBO constraints. In *Proceedings of the 18th International Conference on Rewriting Techniques and Applications (RTA 2007)*, volume 5117 of *LNCS*, pages 389–403, Paris, France, 2007.