Proving Termination Properties with MU-TERM*

Beatriz Alarcón, Raúl Gutiérrez, Salvador Lucas, and Rafael Navarro-Marset

ELP group, DSIC, Universidad Politécnica de Valencia
Camino de Vera s/n, E-46022 Valencia, Spain

Abstract. MU-TERM is a tool which can be used to verify a number of termination properties of (variants of) Term Rewriting Systems (TRSs): termination of rewriting, termination of innermost rewriting, termination of order-sorted rewriting, termination of context-sensitive rewriting, termination of innermost context-sensitive rewriting and termination of rewriting modulo specific axioms. Such termination properties are essential to prove termination of programs in sophisticated rewriting-based programming languages. Specific methods have been developed and implemented in MU-TERM in order to efficiently deal with most of them. In this paper, we report on these new features of the tool.

1 Introduction

Handling typical programming language features such as sort/types and subtypes, evaluation modes (eager/lazy), programmable strategies for controlling the execution, rewriting modulo axioms and so on is outside the scope of many termination tools. However, such features can be very important to determine the termination behavior of programs. For instance, in Figure 1 we show a Maude [10] program encoding an *order-sorted* TRS which is terminating when the sorting information is taken into account but which is *nonterminating* as a TRS (i.e., disregarding sort information) [18]. The predicate `is-even` tests whether an integer number is even. When disregarding any information about sorts, the program `EVEN` is not terminating due to the last rule for `is-even`, which specifies a recursive call to `is-even`. However, when sorts are considered and the hierarchy among them is taken into account, such recursive call is not longer possible due to the need of binding variable `Y` of sort `NzNeg` to an expression `opposite(Y)` of sort `NzPos`, which is not possible in the (sub)sort hierarchy given by `EVEN`.

The notions coming from the already quite mature theory of termination of TRSs (orderings, reduction pairs, dependency pairs, semantic path orderings, etc.) provide a basic collection of abstractions for treating termination problems. For real programming languages, though, having appropriate adaptations, methods, and techniques for specific termination problems is essential. Giving support to *multiple* extensions of such *classical* termination notions is one of the main goals for developing a new version of our tool, MU-TERM 5.0:

`http://zenon.dsic.upv.es/muterm`

* Partially supported by EU (FEDER) and MICINN grant TIN 2007-68093-C02-02.

```
fmod EVEN is
  sorts Zero NzNeg Neg NzPos Pos Int Bool .
  subsorts Zero < Neg < Int .
  subsorts Zero < Pos < Int .
  op 0 : -> Zero .
  op s : Pos -> NzPos .
  op p : Neg -> NzNeg .
  ops true false : -> Bool .
  var X : Pos .
  eq opposite(p(0)) = s(0) .
  eq opposite(p(Y)) = s(opposite(Y)) .
  eq is-even(0) = true .
  eq is-even(s(0)) = false .
  eq is-even(s(s(X))) = is-even(X) .
  eq is-even(Y) = is-even(opposite(Y)) .
  subsorts NzNeg < Neg .
  subsorts NzPos < Pos .
  op is-even : Int -> Bool .
  op is-even : NzPos -> Bool .
  op is-even : NzNeg -> Bool .
  op opposite : NzNeg -> NzPos .
  var Y : NzNeg .
endfm
```

Fig. 1. Maude program

MU-TERM [23, 2] was originally designed to prove termination of *Context-Sensitive Rewriting* (CSR, [21]), where reductions are allowed only for specific arguments $\mu(f) \subseteq \{1, \ldots, k\}$ of the k-ary function symbols f in the TRS. In this paper we report on the new features included in MU-TERM 5.0, not only to improve its ability to prove termination of CSR but also to verify a number of *other* termination properties of (variants of) TRSs.

In contrast to *transformational* approaches which translate termination problems into a classical termination problem for TRSs, we have developed specific techniques to deal with termination of CSR, innermost CSR, order-sorted rewriting and rewriting modulo specific axioms (associative or commutative) by using dependency pairs (DPs, [7]). Our benchmarks show that direct methods lead to simpler, faster and more successful proofs. Moreover, MU-TERM 5.0 has been rewritten to embrace the dependency pair framework [17], a recent formulation of the dependency pair approach which is specially well-suited for mechanizing proofs of termination.

2 Structure and Functionality of MU-TERM 5.0

MU-TERM 5.0 consists of 47 Haskell modules with more than 19000 lines of code. A web-based interface and compiled versions in several platforms are available at the MU-TERM 5.0 web site. In the following, we describe its new functionalities.

2.1 Proving Termination of Context-Sensitive Rewriting

As in the unrestricted case [7], the *context-sensitive dependency pairs* (CSDPs, [3]) are intended to capture all possible function calls in infinite μ-rewrite sequences. In [2], even though our quite 'immature' CSDP approach was one of

our major assets, MU-TERM still used *transformations* [15, 25] and the *context-sensitive recursive path ordering* (CSRPO, [9]) in many termination proofs. Since the developments in [2], many improvements and refinements have been made when dealing with termination proofs of CSR. The most important one has been the development of the *context-sensitive dependency pair framework* (CSDP framework, [3, 20]), for mechanizing proofs of termination of CSR. The central notion regarding termination proofs is that of *CS problem*; regarding mechanization of the proofs is that of *CS processor*. Most processors in the standard DP-framework [17] have been adapted to CSR and many specific ones have been developed (see [3, 20]). Furthermore, on the basis of the results in [28] we have implemented specific processors to prove the infiniteness of CS problems. Therefore, MU-TERM 5.0 is the first version of MU-TERM which is also able to disprove termination of CSR. In the following table, we compare the performance of MU-TERM 5.0 and the last reported version of the tool (MU-TERM 4.3 [2]) regarding its ability to prove termination of CSR over the context-sensitive category of the *Termination Problem Data Base*[1] (TPDB) which contains 109 examples[2]. The results show the power of the new CSDP framework in MU-TERM 5.0, not

Termination Tool	Total	Yes	No	CSDPs	CSRPO	Transf.	Average (sec)
MU-TERM **5.0**	99/109	95	4	99	0	0	0.95s
MU-TERM **4.3**	64/109	64	0	54	7	3	3.44s

Table 1. MU-TERM 4.3 compared to MU-TERM 5.0 in proving termination of CSR.

only by solving more examples in less time, but also disregarding the need of using transformations or CSRPO for solving them.

2.2 Proving Termination of Innermost CSR

Termination of *innermost* CSR (i.e., the variant of CSR where only the deepest μ-replacing redexes are contracted) has been proved useful for proving termination of programs in *eager* programming languages like Maude and OBJ* which permit to control the program execution by means of context-sensitive annotations. Techniques for proving termination of innermost CSR were first investigated in [14, 22]. In these papers, though, the original CS-TRS $(\mathcal{R}, \mu)$ is *transformed* into a TRS whose *innermost* termination implies the innermost termination for $(\mathcal{R}, \mu)$. In [4], the *dependency pair method* [7] has been adapted to deal with termination proofs of innermost CSR. This is the first proposal of a *direct* method for proving termination of innermost CSR and MU-TERM was the first termination tool able to deal with it. Our experimental evaluation shows that the use of *innermost context-sensitive dependency pairs* (ICSDPs) highly improves over the performance of transformational methods for proving termination of innermost CSR: innermost termination of 95 of the 109 considered CS-TRSs could be

[1] See `http://termination-portal.org/wiki/TPDB`
[2] We have used version 7.0.2 of the TPDB.

proved by using ICSDPs; in contrast, only 60 of the 109 could be proved by using (a combination of) transformations and then using AProVE [16] for proving the innermost termination of the obtained TRS. Another important aspect of innermost CSR is its use for proving termination of CSR as part of the CSDP framework [1]. Under some conditions, termination of CSR and termination of innermost CSR coincide [14, 19]. We then switch from termination of CSR to termination of innermost CSR, for which we can apply the existing processors more successfully (see Section 2.6). Actually, we proceed like that in $30-50\%$ of the CSR termination problems which are proved by MU-TERM 5.0 (depending on the particular benchmarks).

2.3 Proving Termination of Order-Sorted Rewriting

In *order-sorted rewriting*, sort information is taken into account to specify the kind of terms that function symbols can take as arguments. Recently, the *order-sorted dependency pairs* have been introduced and proved useful for proving termination of order-sorted TRSs [26]. As a remarkable difference w.r.t. the standard approach, we can mention the notion of *applicable rules* which are those rules which can eventually be used to rewrite terms of a given sort. Another important point is the use of order-sorted matching and unification. To our knowledge, MU-TERM 5.0 is the only tool which implements specific methods for proving termination of OS-TRSs[3]. Our benchmarks over the examples in the literature (there is no order-sorted category in the TPDB yet) show that the new techniques perform quite well. For instance, we can prove termination of the OS-TRS `EVEN` in Figure 1 automatically.

2.4 Proving Termination of $A \vee C$-Rewriting

Recently, we have developed a suitable dependency pair framework for proving termination of $A \vee C$-rewrite theories [5]. An $A \vee C$-rewrite theory is a tuple $\mathcal{R} = (\Sigma, E, R)$ where E is a set containing associative or commutative axioms associated to function symbols of the signature Σ. We have implemented the techniques described in [5] in MU-TERM. Even with only a few processors implemented, MU-TERM behaves well in the equational category of the TPDB, solving 39 examples out of 71. Obviously, we plan to investigate and implement more processors in this field. This is not the first attempt to prove termination of rewriting modulo axioms: C*i*ME [11] is able to prove AC-termination of TRSs, and AProVE is able to deal with termination of rewriting modulo equations satisfying some restrictions.

2.5 Use of Rational Polynomials and Matrix Interpretations

Proofs of termination with MU-TERM 5.0 heavily rely on the generation of polynomial orderings using polynomial interpretations with rational coefficients [24].

[3] The Maude Termination Tool [12] implements a number of *transformations* from OS-TRSs into TRSs which can also be used for this purpose.

In this sense, recent improvements which are new with respect to the previous versions of MU-TERM reported in [2, 23] are the use of an autonomous SMT-based constraint-solver for rational numbers [8] and the use of matrix interpretations *over the reals* [6]. Our benchmarks show that polynomials over the rationals are used in around 25% of the examples where a polynomial interpretation is required during the successful proof. Matrix interpretations are used in less than 4% of the proofs.

2.6 Termination Expert

In the (CS)DP framework, a strategy is applied to an initial (CSR, innermost CSR, ...) problem and returns a proof tree. This proof tree is later evaluated following a tree evaluation strategy (normally, breadth-first search).

With small differences depending on the particular kind of problem, we do the following:

1. We check the system for extra variables (at active positions) in the right-hand side of the rules.
2. We check whether the system is innermost equivalent (see Section 2.2). If it is true, then we transform the problem into an innermost one.
3. Then, we obtain the corresponding dependency pairs, obtaining a (CS)DP problem. And now, recursively:
   (a) Decision point between infinite processors and the *strongly connected component* (SCC) processor.
   (b) Subterm criterion processor.
   (c) Reduction triple (RT) processor with linear polynomials (LPoly) and coefficients in $N_2 = \{0, 1, 2\}$.
   (d) RT processor with LPoly and coefficients in $Q_2 = \{0, 1, 2, \frac{1}{2}\}$ and $Q_4 = \{0, 1, 2, 3, 4, \frac{1}{2}, \frac{1}{4}\}$ (in this order).
   (e) RT processor with simple mixed polynomials (SMPoly) and coefficients in N_2.
   (f) RT processor with SMPoly and rational coefficients in Q_2.
   (g) RT processor with 2-square matrices with entries in N_2 and Q_2.
   (h) Transformation processors (only twice to avoid nontermination of the strategy): instantiation, forward instantiation, and narrowing.
4. If the techniques above fail, then we use (CS)RPO.

The explanation of each processor can be found in [3, 20]. Note also that all processors are new with respect to MU-TERM 4.3 [2].

2.7 External use of MU-TERM

The Maude Termination Tool[4] (MTT [12]), which transforms proofs of termination of Maude programs into proofs of termination of CSR, use MU-TERM's expert as an external tool to obtain the proofs. The context-sensitive and order-sorted

[4] `http://www.lcc.uma.es/~duran/MTT`

features developed as part of MU-TERM 5.0 are essential to successfully handling Maude programs in MTT. The Knuth-Bendix completion tool $\mathsf{mkb}_{\mathrm{TT}}$ [29] is a modern completion tool that combines multi-completion with the use of termination tools. In the web version of the tool, the option to use MU-TERM as the external termination tool is available.

3 Conclusions

We have described MU-TERM 5.0, a new version of MU-TERM with new features for proving different termination properties like termination of *innermost* CSR, termination of *order-sorted rewriting* and termination of rewriting *modulo (associative or commutative) axioms*. Apart from that, a complete implementation of the *CSDP framework* [20] has been included in MU-TERM 5.0, leading to a much more powerful tool for proving termination of CSR. While transformations were used in MU-TERM 4.3, in MU-TERM 5.0 they are not used anymore. The research in the field has increased the number of examples which could be handled with CSDPs in 35 (see Table 1). Regarding proofs of *termination of rewriting*, from a collection of 1468 examples from the TPDB 7.0.2, MU-TERM 5.0 is able to prove (or disprove) termination of 835 of them. In contrast, MU-TERM 4.3 was able to deal with 503 only.

More details about these experimental results in all considered termination properties discussed in the previous sections can be found here:

`http://zenon.dsic.upv.es/muterm/benchmarks/index.html`

Thanks to the new developments reported in this paper, MU-TERM 5.0 has proven to be the most powerful tool for proving termination of CSR in the *context-sensitive* subcategory of the 2007, 2009, and 2010 editions of the International Competition of Termination Tools[5]. Moreover, in the *standard* subcategory, we have obtained quite good results in the 2009 and 2010 editions, being the third tool (among five) in solving more examples. We have also participated in the innermost category in the 2009 and 2010 editions and in the equational category in 2010.

Note also that MU-TERM 5.0 has a web interface that allows inexpert users to prove automatically termination by means of the 'automatic' option. This is very convenient for teaching purposes, for instance. And, apart from MTT, it is the only termination tool that accepts programs in OBJ/Maude syntax.

Therefore, MU-TERM 5.0 is no more a tool for proving termination of CSR only: We can say now that it has evolved to become a powerful termination tool which is able to prove termination of a wide range of interesting properties

[5] See `http://www.lri.fr/~marche/termination-competition/2007/`, where only AProVE and MU-TERM participated, and `http://termcomp.uibk.ac.at/termcomp/` where there were three more tools in the competition: AProVE, Jambox [13] (only in the 2009 edition), and VMTL [27]. AProVE and MU-TERM solved the same number of examples but MU-TERM was much faster. The 2008 edition had only one participant: AProVE.

of rewriting with important applications to prove termination of programs in sophisticated rewriting-based programming languages like Maude or OBJ*.

References

1. Alarcón, B.: Innermost Termination of Context-Sensitive Rewriting. Master's thesis, Departamento de Sistemas Informáticos y Computación, Universidad Politécnica de Valencia, Valencia, Spain (2008)
2. Alarcón, B., Gutiérrez, R., Iborra, J., Lucas, S.: Proving Termination of Context-Sensitive Rewriting with MU-TERM. Electronic Notes in Theoretical Computer Science 188, 105–115 (2007)
3. Alarcón, B., Gutiérrez, R., Lucas, S.: Context-Sensitive Dependency Pairs. Information and Computation 208, 922–968 (2010)
4. Alarcón, B., Lucas, S.: Termination of Innermost Context-Sensitive Rewriting Using Dependency Pairs. In: Wolter, F. (ed.) Proc. of the 6th International Symposium on Frontiers of Combining Systems, FroCoS'07. LNAI, vol. 4720, pp. 73–87. Springer-Verlag (2007)
5. Alarcón, B., Lucas, S., Meseguer, J.: A Dependency Pair Framework for $A \vee C$-Termination. In: Ölveczky, P. (ed.) Proc. of the 8th International Workshop on Rewriting Logic and its Applications, WRLA'10. LNCS, Springer-Verlag, to appear (2010)
6. Alarcón, B., Lucas, S., Navarro-Marset, R.: Proving Termination with Matrix Interpretations over the Reals. In: Geser, A., Waldmann, J. (eds.) Proc. of the 10th International Workshop on Termination, WST'09. pp. 12–15 (2009)
7. Arts, T., Giesl, J.: Termination of Term Rewriting Using Dependency Pairs. Theoretical Computer Science 236(1–2), 133–178 (2000)
8. Borralleras, C., Lucas, S., Navarro-Marset, R., Rodríguez-Carbonell, E., Rubio, A.: Solving Non-linear Polynomial Arithmetic via SAT Modulo Linear Arithmetic. In: Schmidt, R.A. (ed.) Proc. of the 22th Conference on Automated Deduction, CADE'09. LNAI, vol. 5663, pp. 294–305. Springer-Verlag (2009)
9. Borralleras, C., Lucas, S., Rubio, A.: Recursive Path Orderings can be Context-Sensitive. In: Voronkov, A. (ed.) Proc. of the 18th Conference on Automated Deduction, CADE'02. LNAI, vol. 2392, pp. 314–331. Springer-Verlag (2002)
10. Clavel, M., Durán, F., Eker, S., Lincoln, P., Martí-Oliet, N., Meseguer, J., Talcott, C.: All About Maude – A High-Performance Logical Framework, LNCS, vol. 4350. Springer-Verlag (2007)
11. Contejean, E., Marché, C., Monate, B., Urbain, X.: Proving Termination of Rewriting with CiME. In: Rubio, A. (ed.) Proc. of the 6th International Workshop on Termination, WST'03. pp. 71–73 (2003)
12. Durán, F., Lucas, S., Meseguer, J.: MTT: The Maude Termination Tool (System Description). In: Armando, A., Baumgartner, P., Dowek, G. (eds.) Proc. of the 4th International Joint Conference on Automated Reasoning, IJCAR'08. LNCS, vol. 5195, pp. 313–319. Springer-Verlag (2008)
13. Endrullis, J.: Jambox, Automated Termination Proofs For String and Term Rewriting, available at `http://joerg.endrullis.de/jambox.html` (2009)
14. Giesl, J., Middeldorp, A.: Innermost Termination of Context-Sensitive Rewriting. In: Ito, M., Toyama, M. (eds.) Proc. of the 6th International Conference on Developments in Language Theory, DLT'02. LNAI, vol. 2450, pp. 231–244. Springer-Verlag (2003)

15. Giesl, J., Middeldorp, A.: Transformation Techniques for Context-Sensitive Rewrite Systems. Journal of Functional Programming 14(4), 379–427 (2004)
16. Giesl, J., Schneider-Kamp, P., Thiemann, R.: AProVE 1.2: Automatic Termination Proofs in the Dependency Pair Framework. In: Furbach, U., Shankar, N. (eds.) Proc. of the 3rd International Joint Conference on Automated Reasoning, IJCAR'06. LNAI, vol. 4130, pp. 281–286. Springer-Verlag, available at `http://www-i2.informatik.rwth-aachen.de/AProVE` (2006)
17. Giesl, J., Thiemann, R., Schneider-Kamp, P., Falke, S.: Mechanizing and Improving Dependency Pairs. Journal of Automatic Reasoning 37(3), 155–203 (2006)
18. Gnaedig, I.: Termination of Order-sorted Rewriting. In: Kirchner, H., Levi, G. (eds.) Proc. of the 3rd International Conference on Algebraic and Logic Programming, ALP'92. LNAI, vol. 632, pp. 37–52. Springer-Verlag (1992)
19. Gramlich, B., Lucas, S.: Modular Termination of Context-Sensitive Rewriting. In: Proc. of the 4th ACM SIGPLAN International Conference on Principles and Practice of Declarative Programming, PPDP'02. pp. 50–61. ACM Press (2002)
20. Gutiérrez, R., Lucas, S.: Proving Termination in the Context-Sensitive Dependency Pair Framework. In: Ölveczky, P. (ed.) Proc. of the 8th International Workshop on Rewriting Logic and its Applications, WRLA'10. LNCS, Springer-Verlag, to appear (2010)
21. Lucas, S.: Context-Sensitive Computations in Functional and Functional Logic Programs. Journal of Functional and Logic Programming 1998(1), 1–61 (1998)
22. Lucas, S.: Termination of Rewriting With Strategy Annotations. In: Nieuwenhuis, R., Voronkov, A. (eds.) Proc. of the 8th International Conference on Logic for Programming, Artificial Intelligence and Reasoning, LPAR'01. LNAI, vol. 2250, pp. 666–680. Springer-Verlag (2001)
23. Lucas, S.: MU-TERM: A Tool for Proving Termination of Context-Sensitive Rewriting. In: van Oostrom, V. (ed.) Proc. of the 15th International Conference on Rewriting Techniques and Applications, RTA'04. LNCS, vol. 3091, pp. 200–209. Springer-Verlag, available at `http://zenon.dsic.upv.es/muterm/` (2004)
24. Lucas, S.: Polynomials over the Reals in Proofs of Termination: from Theory to Practice. RAIRO Theoretical Informatics and Applications 39(3), 547–586 (2005)
25. Lucas, S.: Proving Termination of Context-Sensitive Rewriting by Transformation. Information and Computation 204(12), 1782–1846 (2006)
26. Lucas, S., Meseguer, J.: Order-Sorted Dependency Pairs. In: Antoy, S., Albert, E. (eds.) Proc. of the 10th International ACM SIGPLAN Sympsium on Principles and Practice of Declarative Programming, PPDP'08. pp. 108–119. ACM Press (2008)
27. Schernhammer, F., Gramlich, B.: VMTL - A Modular Termination Laboratory. In: Treinen, R. (ed.) Proc. of the 20th International Conference on Rewriting Techniques and Applications, RTA'09. LNCS, vol. 5595, pp. 285–294. Springer-Verlag (2009)
28. Thiemann, R., Sternagel, C.: Loops under Strategies. In: Treinen, R. (ed.) Proc. of the 20th International Conference on Rewriting Techniques and Applications, RTA'09. LNCS, vol. 5595, pp. 17–31. Springer-Verlag (2009)
29. Winkler, S., Sato, H., Middeldorp, A., Kurihara, M.: Optimizing $\mathsf{mkb}_{\mathrm{TT}}$. In: Lynch, C. (ed.) Proc. of the 21st International Conference on Rewriting Techniques and Applications, RTA'10. Leibniz International Proceedings in Informatics (LIPIcs), vol. 6, pp. 373–384. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, `http://drops.dagstuhl.de/opus/volltexte/2010/2664` (2010)