

MU-TERM: a tool for proving termination of rewriting with replacement restrictions*

Salvador Lucas¹

DSIC, Universidad Politécnic de Valencia
Camino de Vera s/n, 46022 Valencia, Spain.
slucas@dsic.upv.es

Abstract. This paper describes MU-TERM, a tool which can be used to automatically prove termination of computational restrictions of rewriting such as context-sensitive rewriting and lazy rewriting. The tool can also be used to prove termination of rewriting. In this sense, MU-TERM provides the first implementation of reduction orderings based on polynomial interpretations over the rational numbers.

1 Introduction

Restrictions of rewriting can eventually *achieve* termination of rewriting computations by pruning all infinite rewrite sequences issued from every term. However, such kind of improvements can be difficult to prove. Context-sensitive rewriting (*CSR* [6]) is a restriction of rewriting which is useful for describing semantic aspects of programming languages (e.g., Maude, OBJ2, OBJ3, or CafeOBJ) and analyzing termination of the corresponding programs. In *CSR*, a *replacement map* μ discriminates, for each symbol of the signature, the argument positions $\mu(f)$ on which replacements are allowed. Although several methods have been developed for proving termination of *CSR*, no tool for doing it has been reported to date. Our tool, MU-TERM, is intended to fill this gap. MU-TERM is written in Haskell¹ and makes use of the graphical library wxHaskell². The MU-TERM system is available at

<http://www.dsic.upv.es/~slucas/csr/termination/muterm>

There are two main approaches to prove termination of *CSR*: *direct proofs* use adapted versions of simplification orderings such as RPOs and polynomial orderings to compare the left- and right-hand sides of the rules [4]; and *transformations* which convert the problem of proving termination of *CSR* into a proof of termination of rewriting [5]. Our tool implements both approaches. The modular analysis of termination of *CSR* described in [3] has also been implemented. We have used MU-TERM for developing the experiments reported in

<http://www.dsic.upv.es/~slucas/csr/termination/examples>

where we collect almost all published examples of TRSs which can be proved μ -terminating for concrete replacement maps μ .

The tool can also be used for proving *termination* of rewriting. Polynomials over the rationals [8] are used to generate appropriate reduction orderings (nowadays MU-TERM is the only tool which uses such kind of polynomials). On the other hand, μ -reduction orderings [10] based on such polynomial interpretations are also used together with the dependency pairs approach [1] to prove termination of rewriting.

* Work partially supported by MCyT project TIC2001-2705-C03-01, MCyT Acción Integrada HU 2003-0003 and AVCyT grant GR03/025.

¹ See <http://haskell.org/>.

² See <http://wxhaskell.sourceforge.net>.

2 Interface and functionality

The tool has a quite intuitive graphical user interface. TRSs can be loaded from files containing the rules in the ‘simple format’ $l \rightarrow r$, where l and r are terms in the usual prefix syntax. The system is also able to deal with modules following a subset of the full OBJ / Maude grammar. The advantage is that, in contrast to the simple format, we are able to directly specify the *replacement map* by means of the OBJ / Maude *strategy annotations*. For instance, the following module

```
obj ExNatsOddsPairs is
  sort S .
  op 0 : -> S .
  op s : S -> S .
  op nil : -> S .
  op cons : S S -> S [strat (1 0)] .
  op nats : -> S .
  op pairs : -> S .
  op odds : -> S .
  op incr : S -> S .
  op tail : S -> S .
  vars X XS : S .
  eq nats = cons(0,incr(nats)) .
  eq pairs = cons(0,incr(odds)) .
  eq odds = incr(pairs) .
  eq incr(cons(X,XS)) = cons(s(X),incr(XS)) .
  eq tail(cons(X,XS)) = XS .
endo
```

describes the generation of some infinite lists of natural numbers. The strategy annotation (1 0) for `cons` is interpreted by MU-TERM as follows: $\mu(\text{cons}) = \{1\}$. The very recent TPDB (Termination Problems Data Base) format³ can also be used to fully specify a TRS together with a replacement map.

TRSs are introduced in the system from text files via menu **File**; after successfully reading the file, the TRS becomes the *current TRS*. MU-TERM uses the *current TRS* to perform most actions selected by the user: prove termination, transform, etc.

Panel Termination of CSR (direct proof). The tool implements the techniques described in [8]. A proof of μ -termination of a TRS is transformed into the problem of solving a set of constraints over the coefficients of a polynomial interpretation for the symbols of the TRS. An interesting feature of our technique is that we generate polynomial interpretations with *rational* coefficients. We use CiME [2] as an *external* tool to solve the constraints generated by the system. If CiME is directly available (on the path of the OS), then the constraints are automatically generated and sent to CiME; the answer is automatically received and processed to show the corresponding polynomial interpretation. For instance, for ExNatsOddsPairs we get:

$$\begin{array}{lll} [0] = 0 & [\text{nats}] = 1 & [\text{cons}](X_1, X_2) = X_1 + 1/5 \cdot X_2 \\ [s](X) = X & [\text{pairs}] = 1 & [\text{incr}](X) = X + 1 \\ [\text{nil}] = 0 & [\text{odds}] = 3 & [\text{tail}](X) = 5 \cdot X + 1 \end{array}$$

which proves termination of *CSR* for the system. In this case, the use of rational numbers below one (e.g., $\frac{1}{5}$ in the polynomial interpreting `cons`) plays a crucial role for achieving the proof of termination (see [8, Section 4] for a deeper discussion in this respect). If the

³ See <http://www.lri.fr/~marche/wst2004-competition/format.html>.

modular proofs are activated, then MU-TERM computes a maximal *safe* decomposition of the current system (according to the results in [3]) and separately proves termination of each module.

Panel Transformations. This option permits to apply different transformations for proving termination of *CSR* (see [5] for an overview). Each of them transforms the *current* TRS which remains unchanged (the transformed system is added to the internal list of TRSs). Again, if the *modular proofs* are activated, then MU-TERM uses the computed maximal decomposition of the current system to separately apply the transformations. Eventually, the transformed systems will also be separately proved terminating.

We also include a transformation from CS-TRSs into CS-TRSs (rather than TRSs) that permits to prove termination of *lazy rewriting* as termination of *CSR* (see [7]). However, since the results in [3] only concern *CSR*, no modular treatment is given with this transformation.

Panel Termination of rewriting. Term rewriting is a particular case of *CSR* where the replacement map $\mu_{\top}(f) = \{1, \dots, ar(f)\}$, for all $f \in \mathcal{F}$ is used. Thus, polynomial orderings described above can also be used in proofs of termination of rewriting. On the other hand, Arts and Giesl discuss the use of weakly monotonic and non-monotonic orderings for proving termination of TRSs in combination with the dependency pairs approach [1]. We have implemented the use of polynomial interpretations over the rationals to generate such orderings and we use them together with the dependency pairs approach for proving termination of rewriting (see [9] for further details). In particular, we can use this technique, for instance, to prove termination of the TRSs which are obtained from the previous transformations.

References

1. T. Arts and J. Giesl. Termination of Term Rewriting Using Dependency Pairs *Theoretical Computer Science*, 236:133-178, 2000.
2. E. Contejean and C. Marché. CiME: Completion Modulo E. In *Proc. of RTA '96*, LNCS 1103:416-419, Springer-Verlag, Berlin, 1996.
3. B. Gramlich and S. Lucas. Modular termination of context-sensitive rewriting. In *Proc. of PPDP'02*, pages 50-61, ACM Press, New York, 2002.
4. B. Gramlich and S. Lucas. Simple termination of context-sensitive rewriting. In *Proc. of RULE'02*, pages 29-41, ACM Press, New York, 2002.
5. J. Giesl and A. Middeldorp. Transformation Techniques for Context-Sensitive Rewrite Systems. *Journal of Functional Programming*, to appear, 2004.
6. S. Lucas. Context-sensitive rewriting strategies. *Information and Computation*, 178(1):293-343, 2002.
7. S. Lucas. Lazy Rewriting and Context-Sensitive Rewriting. *Electronic Notes in Theoretical Computer Science*, volume 64. Elsevier Sciences, 2002.
8. S. Lucas. Polynomials for proving termination of context-sensitive rewriting. In *Proc. of FOSSACS'04*, LNCS 2987:318-332, Springer-Verlag, Berlin, 2004.
9. S. Lucas. Polynomials over the rationals in proofs of termination. In *Proc. of WST'04*, this volume, 2004.
10. H. Zantema. Termination of Context-Sensitive Rewriting. In *Proc. of RTA '97*, LNCS 1232:172-186, Springer-Verlag, Berlin, 1997.