African Journal of Mathematics and Computer Science Research Vol. 5(9), pp. 153-157, July 2012
Available online at http://www.academicjournals.org/AJMCSR
DOI: 10.5897/AJMCSR12.017
ISSN 2006-9731© 2012 Academic Journals

Review

An overview of term rewriting systems

Dasharath Singh[1], Ali Maianguwa Shuaibu[2]* and Adeku Musa Ibrahim[1]

[1]Department of Mathematics, Ahmadu Bello University, Zaria-Nigeria.
[2]Department of Mathematics, Statistics and Computer Science, Kaduna Polytechnic, Kaduna, Nigeria.

Accepted 19 March, 2012

It is well-known that termination of finite term of rewriting systems is generally undecidable. Notwithstanding, a remarkable result is that, rewriting systems are *Turing complete*. A number of methods have been developed to establish termination for certain term of rewriting systems, particularly occurring in practical situations. In this paper, we present an overview of the existing methods used for termination proofs. We also outline areas of applications of term rewriting systems along with recent developments in regard to automated termination proofs.

Key words: Confluence, rewriting, term, termination, turing complete.

INTRODUCTION

Rewriting is a very powerful method for dealing computationally with equations. However, applying effectively this approach is in general not straightforward. Alternatively, oriented equations, called rewrite rules, are used to replace equals by equals, but only in one direction. A rewrite system is a set of rules used to compute, by repeatedly replacing parts of a given formula with equal ones until the simplest possible form, called normal form, is obtained. For instance, an equation $2+3=5$ can be interpreted as "5 is the result of computing $2+3$", but not vice versa. This directional replacement is expressed by $2+3 \longrightarrow 5$ which reads "$2+3$ reduces to 5".

This computational aspect of equations naturally leads to term rewriting systems (TRSs, for short). Depending on the kinds of objects that are rewritten, there are different kinds of rewrite systems such as string rewrite (Thue or semi-Thue) systems, TRSs and graph rewriting to mention a few (Baader and Nipkow, 1998; Terese, 2003; Dershowitz, 2005).

The formal study of rewriting and its properties began in 1910 with a paper by Axel Thue (Book, 1987). Significantly, most early models of computation were based on notions of rewriting strings or terms. The emergence of Thue systems; Alonzo Church's lambda Calculus; Andrei Markov's normal algorithms, just to mention a few, led to sustained study of rewriting in the context of programming language semantics (Book, 1987). To be more specific, the study of TRSs originated in combinatory logic (CL) and lambda calculus (Curry and Feys, 1958) developed and deeply analyzed half a century ago to investigate the foundation of functions. CL is actually a TRS. One could say that the paradigmatic example of a TRS is the system of CL. As a matter of fact, the roots of the very notion of term rewriting and much of its theory can be found in the system of CL.

In the recent years, a strong impulse for the study of TRSs (including extensions of the usual rewriting format) is given by the design of functional languages such as Miranda (Terese, 2003). Another strong impulse is given by efforts of many researchers to combine logic programming with functional programming (Toyama, 1990; Marché and Zantema, 2007). In this direction, Toyama (1990) proposed and applied Knuth-Bendix completion algorithm for a better performance. The compiling technique proposed in this algorithm is dynamic in the sense that, rewriting rules are repeatedly compiled in the completion process. The execution time of the completion with dynamic compiling is ten or more times faster than the one obtained with a traditional TRS interpreter.

Two of the most central properties of TRSs are

*Corresponding author. E-mail: shuaibuali16@gmail.com. Tel: +23408069628448.

confluence (the Church-Rosser property) and termination (strong normalization). A confluent and terminating system is called convergent (or complete or canonical) and it defines exactly one normal form for each input term.

Termination proofs play a fundamental role in many applications and the challenges in this area are both practical and theoretical (Marché and Zantema, 2007). From a practical point of view, proving termination is vital issue in software development and formal methods for termination analysis are essential for program verification. From a theoretical point of view, termination is closely connected to mathematical logic and ordinal theory.

The central aspect of attaining the aforesaid goals lies in showing that there is no infinite sequence $u_1, u_2, u_3, \ldots$ such that for all i, u_{i+1} can be obtained from u_i by a replacement using a term rewriting rule. This process is called termination.

Termination, in general, is an undecidable property of TRSs (Terese, 2003). Nevertheless, TRSs possess a very significant property that they are Turing complete; that is, every computable process can be delineated by a rewriting system. Thus, all endeavours made in this regard are intended to discover competing methods that quasi-generally work in cases of practical interest. Most of such methods in vogue are based on well-founded orderings.

Summarily, a TRS is a binary relation over the set of terms of a given signature (or alphabet). The pairs of the relation are used for computing by replacements until an irreducible term is eventually reached. This is how the absence of infinite sequences of replacements grants termination. A TRS is terminating if all rewrite sequences are finite. Rules of a terminating system are called reduction or rewrite rules.

Before we endeavour to present an overview of the researches undertaken in this area, we briefly make clear some elementary illustrations, the notion of a rewrite rule and its action.

Example 1

Consider the following rewrite rules:

$$R_1: A(x, S(y)) \longrightarrow S(A(x, y)),$$

$$R_2: A(x, 0) \longrightarrow x.$$

To simplify $A\left(0, S\left(A(S(0), 0)\right)\right)$, we have

$$A\left(0, S\left(A(S(0), 0)\right)\right) \longrightarrow S\left(A\left(0, A(S(0), 0)\right)\right)$$
$$\longrightarrow S\left(A\left(0, S(A(0,0))\right)\right)$$
$$\longrightarrow S\left(S(A(0,0))\right)$$
$$\longrightarrow S(S(0)).$$

This is terminating. Note that the first rule makes $'S$ move upwards while the second rule makes terms smaller.

Example 2

Using the rewrite rule $f(g(x)) \longrightarrow g(f(x))$, we have

$$f(f\left(g\left(f(g(x))\right)\right)) \longrightarrow f(f\left(g\left(g(f(x))\right)\right))$$
$$\longrightarrow f(g\left(f\left(g(f(x))\right)\right))$$
$$\longrightarrow g(f\left(f\left(g(f(x))\right)\right))$$
$$\longrightarrow g(f\left(g\left(f(f(x))\right)\right))$$
$$\longrightarrow g(g\left(f\left(f(f(x))\right)\right)).$$

The fs moves to the right while the gs moves to the left.

Example 3

$$f(g(x)) \longrightarrow g(g\left(f(f(x))\right))$$

looks terminating with the fs moving to the right and the gs to the left. But it gives rise to an infinite rewrite sequence:

$$f(g(g(x)) \longrightarrow g(g\left(f\left(f(g(x))\right)\right))$$
$$\longrightarrow g(g\left(f\left(g\left(g\left(f(f(x))\right)\right)\right)\right)) \longrightarrow \ldots$$

Remark

We reemphasize that the termination of such derivations is crucial using rewriting in proofs and computations. The difficulty in proving the termination of a system, such as those in the previous examples, stems from the fact that while some rules may decrease the size of a term, other rules may increase its size and duplicate occurrences of subterms. If $x \longrightarrow y$ is a reduction then y is somehow simpler or smaller than x. If it is generative then y is

generally more complex or larger than x. Any proof of termination must take into consideration the different possible rewrite sequences generated by the nondeterministic choice of rules.

AN OUTLINE OF TERM REWRITING SYSTEMS

As mentioned earlier, one major property which a TRS needs to satisfy is termination. Generally, termination or halting computing processes explicitly uses dominance orderings in addressing problems relating to termination proofs in theoretical computer science. This aspect of termination started receiving attention in the 1970s. Knuth (1973) applied dominance ordering for demonstrating termination of a sorting algorithm.

Methods of termination

Several methods for proving termination of TRSs have been developed. Most of these methods are based on reduction orderings which are well-founded, compatible with the structure of terms and stable with respect to substitutions. Examples of these methods include, Knuth-Bendix order, polynomial interpretations, multiset order, lexicographic path order, recursive decomposition order, multiset path order, semantic path order, transformation order, forward closures, semantic interpretations, dummy elimination and distribution elimination (Dershowitz, 1987; Hirokawa, 2006; Marché and Zantema, 2007). Proving termination using one of these particular methods, in general, proves more than just the absence of infinite derivation sequences. It turns out that such a proof in many cases implies an upper bound on derivation height, expressed as the function Dh_R on terms (Hofbauer and Lautemann, 1989). Thus, the rate of growth of Dh_R can be used for measuring the strength of termination proof methods.

The use of rewrite systems as termination functions and the formulation of abstract monotonicity conditions are explored in (Bachmair and Dershowitz, 1986). Gorn (1973) uses a stepped lexicographic ordering (under which longer sequences are larger) to prove termination of differentiation. Exponential interpretation method of termination is exploited in (Iturriaga, 1967). The cases where Iturriaga's method works are those which the operators are partially ordered so that the outermost (virtual) operators of the left-hand side of the rules are greater than any other operators.

Singh and Singh (2009) outline an alternative proof of the well-foundedness of the nested multiset ordering. It is shown that the set of nested multisets over a given set forms a cumulative type structure. Also, by exploiting the notion of sets bounded in rank, a necessary and sufficient condition for the well-foundedness of the nested multiset ordering is outlined.

Knuth and Bendix (1970) devised a recursive ordering that combines the notion of precedence with a simple linear weight. The weight is the sum of the weights of all the symbols which are non-negative integers. Furthermore, they point out that their method is applicable to handle duplicating systems (one that has more occurrences of a variable on the right than on the left). Lankford (1979) however suggests a way of extending the method of Knuth and Bendix by using integer polynomial weights (with positive coefficients to guarantee that terms are greater than subterms).

The recursive path ordering (RPO) is shown to be a quasi well-ordering in case of a finite signature and hence well-founded (Toyama, 1990; Dershowitz, 1987). Moreover, it is known to be well-founded for infinite signature and also in the case of the precedence relation, is well-ordering. The RPO has also been adapted to handle associative-commutative operators by flattening and transforming terms (distributing large operators over small ones) before comparing them. The difficulty encountered is that of ensuring monotonicity, since flattening alone may not ensure monotonicity (Bachmair and Dershowitz, 1986; Dershowitz, 1987).

In the work of Bergstra and Klop (1985), an alternative definition of RPO was put forward and proved to be an iterative path order (IPO). The exact relationship between the recursive and iterative approaches to path orders was investigated. It was shown that both approaches coincide in the case of transitive relations (orders). Klop et al. (2006) employed a proof technique due to Buchholz (1995), provides a direct proof of termination for the IPO starting from an arbitrary terminating relation on the signature. Both the proofs essentially rely on a natural number labeled variant ℓex^{ω} of the auxiliary TRS ℓex.

Jouannaud et al. (1982) use the recursive decomposition ordering to prove termination by comparing two terms and the comparison may only stop where two decompositions have incomparable symbols as their first components. The path of subterms ordering is explicitly considered in Dershowitz (1987) to extend the RPO. Also, semantic path ordering (SPO) of Kamin and Lévy (1980) is used to prove termination where terms are compared lexicographically. Furthermore, it was shown that the use of SPO in a termination proof necessarily requires the monotonicity condition to hold.

Another approach, called simple path ordering, defined on multisets for proving termination of differentiation was devised by Plaisted (1983). Here, terms are mapped into multisets of sequences of function symbols: $[\![t]\!] = \{\text{paths in } t\}$, where a path is a sequence of operators, beginning with the outermost one of the whole terms and taking subterms until a constant is reached.

The use of monotonic polynomial interpretations for termination proofs was suggested by Bachmair and Dershowitz (1986) and Dershowitz (1987). Using this

method, an integer polynomial of degree n is associated with each n –ary operator f. The choice of coefficients must ensure monotonicity and that the terms are mapped into nonnegative integers only. This is the case if all coefficients are positive. Zantema (1992) provides a classification of termination of TRSs based on types of orderings. The strongest type of termination he considers was polynomial termination. Polynomial terminations are those that can be proved by a polynomial interpretation (PI). Also, Zantema considers the equivalence relation on terms generated by permuting arguments of operation symbols of multiset status. The main difficulty with proving termination using PI technique is that, polynomial terminating TRSs are double-exponentially bounded in the size of the initial term (Hofbauer and Lautemann, 1989).

Toyama (1990) develops a simple method for proving the equivalence of two given TRSs without the explicit use of induction, and demonstrates that the method can be effectively applied to deriving a new TRS from a given one by using equivalence transformation rules. Cropper and Martin (2001) provide a classification of polynomial orderings on monadic terms where they investigate polynomial orderings which are reduction orderings on term algebras determined by the polynomial interpretations of the function symbols. Such orderings offer an apparent flexibility in allowing the choice of polynomials with arbitrarily high degree and large number of coefficients. It also provides the opportunity to use well-understood decision procedures for real arithmetic to verify the polynomial inequalities needed to prove termination.

In automation, we often encounter large search spaces for parameters required by termination criteria and there is a natural trade-off between power and efficiency. In order to optimize the said trade-off, developing efficient search techniques is of cardinal importance. In this regard, increasing emphasis has shifted towards transformation methods like the dependency pair method or semantic labeling. These techniques have significantly increased the possibility of proving termination automatically. Arts and Giesl (2000) while elaborating on termination of TRSs, demonstrate that the application of dependency pairs does exclude dependency inequalities for the right-hand subterms, which also appear on the left. Hirokawa (2006) develops automated termination methods based on the dependency pair technique. These methods are intended to make termination tools more powerful and efficient. Avanzini and Moser (2010) develop techniques to automatically classify the complexity of TRSs and introduce polynomial path orders (POP^*, for short) and its *extensions*. POP^* is a syntactic restriction of the multiset path order on terms, and whenever compatibility of a TRS $\mathcal{R}$ with POP^* can be established, the innermost runtime complexity of $\mathcal{R}$ is polynomially bounded. The runtime complexity of a TRS is a measure of the maximal number of rewrite steps as a function of the size of the initial term, where the initial (or basic) terms are restricted argument normalized terms.

The construction of a PI and the embedding of the rewrite relation into the multiset path order (MPO) or the lexicographic path order (LPO) or the Knuth-Bendix path order (KBO) are among the most prominent methods for proving termination of a (finite) TRS. A termination proof for a TRS using one of these methods is essential for obtaining an upper bound on the derivation length function. This function by mapping a natural number n to the length of a longest derivation, starting with a term of size bounded by n, provides a natural measure for the strength of a termination proof method. Lepper (2001) lists bounds for these methods along with some restrictions pertaining to PI and KBO in a chronological order.

Complexity characterizations for TRSs have been studied by some researchers (Hofbauer, 1992 and Weiermann, 1995). It was shown in Hofbauer (1992) that termination proofs using MPOs imply primitive recursive derivation lengths; and in Weiermann (1995) work that termination proofs using LPOs imply multiply recursive derivation lengths.

In view of the fact that termination of all derivations initiated by a given term is undecidable and termination for all terms is not even partially decidable, all one can hope for, is to develop competing computing termination tools.

The inherent lack of direction in TRSs, computation of a logic program involving TRSs, virtually entails that any non-trivial program would terminate only for certain classes of inputs. Thus, termination analysis in logic programming turns out to be of utmost significance. In recent years, the subject has been widely studied and significant advances have been made. Currently, there exist a number of fully-automated tools to prove termination of a given logic program with respect to a given class of inputs (Hirokawa, 2006). MPO and LPO have been implemented in *REVE* and *RRL* (Dershowitz, 1987). *REVE* and *RRL* are preferred automated termination provers for TRSs in vogue. There are also a number of other tools that has attempt proving termination automatically. These include, *AProVE, Cariboo, CiME, Jambox, Termptation, NTI, Torpa, Matchbox, Mu Term, TPA,* T_TT*, VMTL* etc. (Marché and Zantema, 2007, for details).

CONCLUSIONS

TRSs are closely related to theorem proving and declarative programs. Termination of a TRS is a desired property proposed to ensure that all computation paths end. The current direction of research is largely towards automation of termination analysis for TRSs. These automated methods are intended to develop competitive

termination tools.

The increasing interest in automated termination analysis of TRSs has led to an annual International Competition of Termination Tools initiated in 2004 (Marché and Zantema, 2007). It aims at identifying most talented competitors who could obtain an assigned task by applying appropriate choices of termination proving techniques within a time limit of 60 s.

REFERENCES

Arts T, Giesl J (2000).Termination of term rewriting using dependency pairs. Theor. Comp. Sci. 236:133-178.

Avanzini M, Moser G (2010).Complexity Analysis by Graph Rewriting Revisited. Institute of Computer science, University of Innsbruck, Austria, Technical Report.

Baader F, Nipkow T (1998). Term Rewriting and All That. Cambridge University Press. pp. 61-132.

Bachmair L, Dershowitz N (1986). Communication, transformation and termination. 8th CADE, LNCS 230, J. org. H. Siekmann (ed.), Springer-Verlag. pp. 5-10.

Bergstra J, Klop J (1985). Algebra for communicating process. TCS 37(1):171-199.

Book RV (1987). Thue systems as rewriting systems. J. Symbolic Computat. 3 (1&2):39-68

Buchholz W (1995). Proof-theoretic analysis of termination proofs. APAL 75(1-2):57-65.

Cropper N, Martin U (2001). The classification of Polynomial Orderings on Monadic terms. Applicable Algebra in Engineering communication Comp. 12(3):197-226.

Curry HB, Feys R (1958). Combinatory Logic, Vol.1, North- Holland.

Dershowitz N (1987). Termination of Rewriting. J. Symbolic Comput. 3(1&2):69-115.

Dershowitz N (2005). Open. Closed. Open: Proceedings of Conference on Rewriting Techniques and Applications (RTA). pp. 376 - 393.

Gorn S (1973). On the conclusive validation of symbol manipulation processes (How do you know it has to work?). J. Franklin Institute 296:6.

Hofbauer D, Lautemann C (1989). Termination proofs and the length of derivations: proceedings of the 3rd International Conference on Rewriting Techniques and Applications.LNCS 355. Springer Verlag. pp. 167 - 177.

Hirokawa N (2006). Automated Termination Analysis for Term Rewriting. Dissertation, Faculty of mathematics, computer science and physics, University of Innsbruck.

Hofbauer D (1992). Termination Proofs with Multiset Path Orderings imply Primitive Recursive Derivation Lengths. Theor. Comp. Sci. 105(1):129-140.

Iturriaga R (1967). Contributions to Mechanical Mathematics. Ph.D. Thesis, Department of mathematics, Carnegie - Mellon University, Pittsburgh, Pennsylvania.

Jouannaud JP, Lescanne P, Reinig F (1982).On Multiset Orderings. Information Processing Letters. 15:57 - 63.

Kamin S, Levy JJ (1980). Two Generalizations of the Recursive Path Ordering. Unpublished note, Department of Computer Science, University of Illinois, Urbana, IL.

Klop JW, Van Oostrom V, de Vrijer R (2006). Iterative Lexicographic Path Orders. LNCS 4060, Kafutatsugi et al (eds.), Amsterdam. pp. 541 – 554.

Knuth DE, Bendix PB (1970). Simple Word Problems in Universal Algebras, in Computational Problems in Abstract Algebra, J. Leech (ed.), Pergamon Press.

Knuth DE (1973). The Art of Computer Programming: Fundamental Algorithms, Addison-Wesley 2:2.

Lankford DS (1979). On Proving Term Rewriting Systems are Noetherian. Memo MTP-3, Mathematics Department, Louisiana Tech. University, U.S .A.

Lepper I (2001). Simplification Orders in Term Rewriting: Derivation Lengths, Order Types and Computability, Inaugural-Dissertation universität Münster, Germany.

Marché C, Zantema H (2007). The Termination Competition. Franz Baader (ed.): Proceedings of the 18th International Conference on Rewriting Techniques and Applications, LNCS 4533, Springer Verlag. pp. 303-313.

Plaisted DA (1983). An Associative Path Ordering. Proceedings of NSF Workshop on Rewrite Rule Laboratory, U.S.A. pp. 123-136.

Singh D, Singh JN (2009). An alternative proof of the well-foundedness of the nested multiset ordering. Int. Math. Forum 4(8):359-362.

TERESE (2003). Term Rewriting Systems. Marc Bezem *et al*. (eds.), Vol.55 of Cambridge Tracts in Theoretical Computer Science, Cambridge University Press.

Toyama Y (1990). Term rewriting and the Church-Rosser property. Ph.D Dissertation, Tohoku University.

Weiermann A (1995). Termination proofs for TRS with LPO imply multiply recursive derivation lengths. TCS 139:355-362.

Zantema H (1992). Termination of Term Rewriting by Interpretation. RUU-CS 9214, Department of Computer Science, Utrecht University, Netherlands.