Termination analysis of rewriting systems based on ligaturization and stringification

Kento Yamazaki (1410043)

School of Information Science,
Japan Advanced Institute of Science and Technology

February 10, 2016

Keywords: string rewriting, term rewriting, termination, automation.

In this paper, we conduct a survey on termination of string rewriting systems. The aim of this study is to develop a simple method to prove termination of string rewriting systems which is suitable for automation, and also to show the fact that this method is applicable to termination analysis of term rewriting systems. We propose a ligature order as a new reduction order and combine it with dependency pairs. We use argument filtering for designing reduction pairs when dependency pair method is employed. In addition, we show that this technique is also available for proving termination of term rewriting systems through stringification. We evaluate these methods with an actual implementation.

1 Background

From social infrastructure, such as nuclear power plants, communication networks, and transportation, to everyday items like mobile phones and cars, software plays a critical role in the modern society. In many cases, a non-terminating program execution means that the software runs out of control. Therefore, techniques to prove termination are essential to verification of safe software. Term rewriting is a computational model in which an equation can be regarded as a rewrite rule whose direction is from the left-hand side to the right-hand side. In addition, rewriting is a computational model which is used for several declarative programming languages such as CafeOBJ and OCaml. String rewriting is a term rewriting in which objects to be computed are restricted to strings. Term rewriting systems and string rewriting systems are terminating if they do not have an infinite rewrite sequence. Termination is an important property since a terminating rewriting system always returns an swers. However, termination of rewriting systems are generally undecidable. Therefore, many studies have been carried out in order to determine the conditions under which rewriting systems are terminating. In this field of study, seminal studies are often conducted using string rewriting systems before constructing a rewriting system which is suitable for application to an initial target. In fact, Matrix interpretation and match bound method are typical examples of such cases. A classical method to prove termination is by showing all rewrite rules are in a well-founded order called a reduction order. If all rewrite rules are in a reduction order,

$\mathcal{R}$ is terminating. There are various reduction orders such as the lexicographic path order (Kamin and Levy,1980), the Knuth-Bendix order (Knuth and Bendix, 1970), and a polynomial interpretation (Lankford, 1979). As a rule of thumb, it is known that the first two methods are more suitable for automated termination provers. In 2000, the method using dependency pairs was introduced (Arts and Gisel, 2000). In this method, a pair of a well-founded order and a preoder called reduction pair is used in order to prove termination. Reduction pairs are constructed based on a reduction order, and it is known that polynomial interpretations are very efficient when dependency pairs are used.

2 Achievements

In this paper, we propose a ligature order and a stringification order as methods for termination analysis. Using a length-lexicographic ligature order, we can decide termination of string rewriting systems which is not decidable with the length-lexicographic order. Using the ligature order, we also can decide termination of problems in braid theory which is not decidable with the length-lexicographic order.

Suppose the following set of rewrite rules which constitute a problem in braid theory.

$$\mathcal{R} = \left\{ \begin{array}{rcll} aaba & \to & abab & (1) \\ bbab & \to & baba & (2) \\ ca & \to & ac & (3) \\ caba & \to & bacb & (4) \end{array} \right\}$$

The length-lexicographic order is an order based on the lengths and the letters of words on both sides in an equation. Assuming a has a higher priority to b, as same as dictionaries, the left-hand side is greater in (1) while the right-hand side is greater in (2). Then we combine the length-lexicographic order with ligaturization which we propose. Ligaturization is a concatenation of every adjoint string. We have the following string rewriting system by ligaturization, and its rewrite rules are corresponding to (1) and (2) in $\mathcal{R}$, respectively.

$$\underline{aa}\ ab\ ba \to \underline{ab}\ ba\ ab$$
$$\underline{bb}\ ba\ ab \to \underline{ba}\ ab\ ba$$

In this string rewriting system, assuming aa and ab have a higher priority to bb and ba, respectively, the left-hand sides become greater due to underlined parts. In (3) and (4), however, simple ligaturization does not result in a reduction order. This is caused because this order is not closed under context. We discuss how to fix this problem.

Termination is decidable with the length-lexicographic order only when the left-hand side is greater than or equal to the right-hand side in every rewrite rule. Therefore, we introduce argument filtering and show that termination is decidable in some cases even where the right-hand side is longer than the left-hand side in a rewrite rule. The system $\mathcal{R} = \{ab \to ba, ba \to acb\}$ is one of such examples. By ignoring the letters after c in the second rewrite rule, termination is decidable using the length -lexicographic ligature order.

We show that this method for termination analysis of string rewriting systems is also applicable to termination analysis of TRSs. We evaluated the performance of our method

using the termination problems from TPDB (Termination Problems Data Base). Problems which are not solvable by known termination provers, such as AProVE or TTT2, are also not solvable using stringification order. However, we succeeded in a novel attempt to apply our techniques for termination analysis of string rewriting systems to TRSs. The following TRS is one of such examples. After stringification, the termination of the system is provable using the length-lexicographic order.

$$\begin{aligned} 1\colon &\quad \mathsf{len}(\mathsf{nil}) \to \mathsf{0} \\ 2\colon &\quad \mathsf{len}(\mathsf{cons}(x,y)) \to \mathsf{s}(\mathsf{len}(y)) \end{aligned}$$

By the dependency pair method, the termination of the TRS can be concluded from the following order constraints.

$$3\colon \quad \mathsf{len}^\sharp(\mathsf{cons}(x,y)) \to \mathsf{len}^\sharp(y)$$

$$\begin{aligned} \left\{\begin{array}{c}\mathsf{len}\\ \mathsf{len}_1\mathsf{nil}\end{array}\right\} &\succsim^{\mathsf{H}} \varnothing \\ \left\{\begin{array}{c}\mathsf{len}\\ \mathsf{len}_1\mathsf{cons}\end{array}\right\} &\succsim^{\mathsf{H}} \left\{\begin{array}{c}\mathsf{s}\\ \mathsf{s}_1\mathsf{len}\end{array}\right\} \quad \{\mathsf{len}_1\mathsf{cons}_2\} \succsim^{\mathsf{H}} \{\mathsf{s}_1\mathsf{len}_1\} \\ \left\{\begin{array}{c}\mathsf{len}^\sharp\\ \mathsf{len}_1^\sharp\mathsf{cons}\end{array}\right\} &\succ^{\mathsf{H}} \{\mathsf{len}^\sharp\} \qquad \{\mathsf{len}_1^\sharp\mathsf{cons}_2\} \succ^{\mathsf{H}} \{\mathsf{len}_1^\sharp\} \end{aligned}$$

Here $\succ^{\mathsf{H}}$ and $\succsim^{\mathsf{H}}$ are Hoare extension of the corresponding length-lexicographic order with $\mathsf{len} > \mathsf{s}_1$.

These proposed techniques on ligaturization and stringification have been implemented as an automatic termination tool and evaluated on a large collection of termination problem.