Inferring RPO Symbol Ordering

Wei Du, Paliath Narendran, Michael Rusinowitch

Inferring RPO Symbol Ordering

(work in progress)

Wei Du[1], Paliath Narendran[2], and Michael Rusinowitch[3]

[1] University at Albany, SUNY
wdu2@albany.edu
[2] University at Albany, SUNY
pnarendran@albany.edu
[3] LORIA-INRIA Nancy-Grand Est
rusi@loria.fr

Abstract

We study the recursive path ordering (RPO) in the context of string-rewriting systems. We are interested in finding a symbol ordering in RPO such that for every rule in the rewriting system, the left hand side is higher than the right hand side. We show that this SYMBOL-ORDER problem is NP-complete by a reduction from the 2-3-SAT problem. We also work on the one-rule case and show a polynomial time algorithm for such systems.

Keywords: recursive path ordering, string-rewriting systems, NP-complete

1 Introduction

Termination orderings for term rewriting systems is an important topic. A major breakthrough in this came when Nachum Dershowitz introduced *simplification orderings* and, in particular, the *recursive path ordering* (RPO) scheme [2]. The elegance of this scheme is that it lifts an ordering on the function symbols to an ordering on terms.

On strings, RPO can be defined in two ways: Let Σ be an alphabet and $\succ$ be an ordering on Σ. Then $x >_{rpo} y$ if and only if one of the following conditions hold:

(1) $y = \varepsilon$ and $|x| > 0$.

(2) $x = au$, $y = av$, and $u >_{rpo} v$.

(3) $x = au$, $y = bv$, and either

(3a) $u \geq_{rpo} y$, or

(3b) $a \succ b$ and $x >_{rpo} v$.

We use $x \overset{\exists}{>}_{rpo} y$ to denote "there is an ordering $\succ$ on Σ such that $x >_{rpo} y$". Note that we can assume that the ordering is total, since any partial ordering can be extended to a total ordering without affecting

[1] Some of the results in this paper will likely be in the first author's dissertation.

the string comparison.

In the case where the symbol ordering $\succ$ is total, an alternative characterization was introduced in [5]. Let $max(w,\Sigma)$ stand for the *maximal* symbol of Σ that occurs in w and let $mul(w,\Sigma)$ be the number of times *this* symbol occurs in w, i.e., $\#_{max(w,\Sigma)}(w)$. Now $w >_{rpo} w'$ iff one of the following holds:

1. $max(w,\Sigma) \succ max(w',\Sigma)$
2. $max(w,\Sigma) = max(w',\Sigma)$ and $mul(w,\Sigma) > mul(w',\Sigma)$
3. $\boldsymbol{a} = max(w,\Sigma) = max(w',\Sigma)$, $mul(w,\Sigma) = mul(w',\Sigma)$,

$$w = w_0\boldsymbol{a}w_1\boldsymbol{a}w_2\ldots\boldsymbol{a}w_k$$

$$w' = u_0\boldsymbol{a}u_1\boldsymbol{a}u_2\ldots\boldsymbol{a}u_k$$

and there exists $0 \le i \le k$ such that $w_i >_{rpo} u_i$ and for all $j > i$ we have $w_j = u_j$.

The *Parikh vector* of a string w over an (ordered) alphabet $\{a_1,\ldots,a_n\}$ is the n-tuple

$$\pi(w) \;=\; \Big(\#_{a_1}(w),\ldots,\#_{a_n}(w)\Big)$$

The partial order $\le_v$ on integer vectors is defined as usual: $(c_1,\ldots,c_n) \;\le_v\; (b_1,\ldots,b_n)$ if and only if $c_i \le b_i$ for all i. Let $<_v$ be the strict ordering associated with $\le_v$.

For a string w, let w_i denote the suffix of length i. For a symbol $a \in \Sigma$, let $\mu(a,w)$ be the longest suffix of w that does not contain a.

1.1 Some properties of *rpo* on strings

Lemma 1. *For all x,y,z, $x >_{rpo} y$ if and only if $xz >_{rpo} yz$.*

Lemma 2. *For all x,y,z, $x >_{rpo} y$ if and only if $zx >_{rpo} zy$.*

Lemma 3. *If $x_1 >_{rpo} y_1$ and $x_2 >_{rpo} y_2$ then $x_1x_2 >_{rpo} y_1y_2$.*

In the light of Lemma 1, we can prove the following for two strings *that share no common suffix*:

Lemma 4. *Let w and w' be two strings that do not share a common suffix. Then $w >_{rpo} w'$ iff one of the following holds:*

1. $max(w,\Sigma) \succ max(w',\Sigma)$
2. $max(w,\Sigma) = max(w',\Sigma)$ and $mul(w,\Sigma) > mul(w',\Sigma)$
3. $\boldsymbol{a} = max(w,\Sigma) = max(w',\Sigma)$, $mul(w,\Sigma) = mul(w',\Sigma)$, and $\mu(\boldsymbol{a},w) >_{rpo} \mu(\boldsymbol{a},w')$.

2 NP-completeness

We prove that the following problem which we call the SYMBOL-ORDER problem, is NP-complete:

Input: A string-rewriting system $\{l_i \rightarrow r_i \mid 1 \leq i \leq n\}$

Question: Is there a symbol ordering $\succ$ such that $l_j >_{rpo} r_j$ for all j?

The reduction is from the following NP-complete problem which we call "2-3-SAT":

Input: A set S consisting of 2-clauses and 3-clauses where each 2-clause only has negative literals and each 3-clause only has positive literals.

Question: Is S satisfiable?

Note that this is a small variant of the well-known *monotone 3SAT* problem.

Lemma 5. *The 2-3-SAT problem is NP-complete.*

Proof. We reduce the 3-SAT problem. Given a set of 3-clauses we can replace every negative literal $\neg a$ by a new positive literal a' and add the 2-clauses: $\neg a \vee \neg a'$ and $a \vee a'$. Then we introduce new positive literals z_1, z_2 and replace $a \vee a'$ by the clauses $a \vee a' \vee z_1$, $a \vee a' \vee z_2$ and $\neg z_1 \vee \neg z_2$. The resulting set of clauses can be obtained in polynomial time from the initial 3-SAT problem. The 3-SAT problem is satisfiable iff the resulting set is satisfiable. □

Lemma 6. *The SYMBOL-ORDER problem is NP-complete.*

Proof. We reduce the 2-3-SAT problem to the SYMBOL-ORDER problem. Let Φ be any CNF formula consisting of 2-clauses and 3-clauses as in the statement of the 2-3-SAT problem. For each variable x_i in Φ, we introduce a symbol a_i. We also introduce a symbol d to simulate truth and falsehood of a variable: x_i is true if and only if $a_i \succ d$, and x_i is false if and only if $a_i \prec d$. For each 3-clause $(x_i \vee x_j \vee x_k)$ we introduce the rule

$$a_i a_j a_k \rightarrow d$$

and for each 2-clause $(\neg x_m \vee \neg x_n)$ we add the rules

$$\begin{array}{lcll} d a_m a_n d & \rightarrow & a_n a_m & \text{and} \\ d a_n a_m d & \rightarrow & a_m a_n & \end{array}$$

From 2-3-SAT to the SYMBOL-ORDER problem: Suppose $(x_i \vee x_j \vee x_k)$ is true, then at least one of these three variables must be true. Let x_i be true, then $a_i \succ d$. Based on the definition of RPO, clearly $a_i a_j a_k >_{rpo} d$. When x_j or x_k is true, we can get the same ordering by the definition.

Suppose $(\neg x_m \vee \neg x_n)$ is true, then at least one of these two variables must be false. If x_m is false, then $a_m \prec d$. By the definition of RPO, we can determine that $d a_m a_n d >_{rpo} a_n a_m$ and $d a_n a_m d >_{rpo} a_m a_n$.

The step is similar when x_n is false.

From the SYMBOL-ORDER problem to 2-3-SAT: Suppose $a_i a_j a_k >_{rpo} d$, then at least one of these three symbols is higher than d. We can prove this by contradiction: assume none of these three symbols is higher than d. Then $d >_{rpo} a_i a_j a_k$ by the definition of RPO. If any symbol is higher than d, we give the corresponding variable a truth value of true. Thus if $a_i \succ d$, then assign true to the variable x_i and the clause is satisfied.

Suppose $da_m a_n d >_{rpo} a_n a_m$ and $da_n a_m d >_{rpo} a_m a_n$, then at least one of these two symbols must be lower than d. This can also be proved by contradiction, if $a_m \succ a_n \succ d$, then the second rule will not be oriented correctly, and if $a_n \succ a_m \succ d$, then the first rule will not be oriented correctly. □

3 The one-rule case

Let Σ be an alphabet and $\Delta \subseteq \Sigma$. A Δ-skeleton of a string w over Σ^* is defined as the homomorphic image $h(w)$ where h is defined as

$$\begin{aligned} h(a) &= a \quad \text{if } a \in \Delta \\ &= \varepsilon \quad \text{otherwise} \end{aligned}$$

In other words, we erase every symbol in w that is not in Δ.

Lemma 7. *Let x and y be distinct strings.*

(1) *If $\#_a(x) > \#_a(y)$ for some $a \in \Sigma$, then $x \overset{\exists}{>}_{rpo} y$.*

(2) *If $\#_a(y) > \#_a(x)$ for all $a \in \Sigma$, then $x \overset{\exists}{\not>}_{rpo} y$.*

Proof. For (1), let a be the max symbol in the ordering, then according to Lemma 4 of [5], whichever string has more copies of the max symbol will be higher. For (2), let a be the max symbol of some ordering, as y has more copies of a, y will be higher. This case is true for any ordering because y has more copies of every symbol. Therefore, x can't be higher. □

Lemma 8. *Let x and y be two* distinct *strings whose Parikh vectors are the same. Then $x \overset{\exists}{>}_{rpo} y$.*

Proof. We can assume without loss of generality that x and y do not share a common suffix. Thus the rightmost symbols in x and y will be different. Let $x = x'a$ and $y = y'b$. Choose $\succ$ such that b is the max symbol. Then $\mu(b,y) = \varepsilon$ and $\mu(b,x) \neq \varepsilon$. □

Corollary 8.1. *Let x and y be two* distinct *strings of the same length. Then $x \overset{\exists}{>}_{rpo} y$.*

For two strings x, y, we define

$$\Delta_{x,y} = \{ a \in \Sigma \mid \#_a(x) = \#_a(y) \}.$$

We omit the subscript when the strings under consideration are obvious from the context.

Lemma 9. *Let x, y be two strings with no common suffix. Then $x \overset{\exists}{>}_{rpo} y$ iff either* (a) *there exists $\boldsymbol{a}$ such that $\#_{\boldsymbol{a}}(x) > \#_{\boldsymbol{a}}(y)$ or* (b) *there exists a in $\Delta_{x,y}$ and $\mu(a,x) \overset{\exists}{>}_{rpo} \mu(a,y)$.*

The following lemma establishes a base case for our algorithm.

Lemma 10. *Let $y \in \Sigma^*$, $a \in \Sigma$ and $m > 0$. Then $a^m \overset{\exists}{>}_{rpo} y$ iff $\#_a(y) < m$.*

Sketch of the algorithm:

As a preprocessing step remove common suffixes first. Then compute Parikh vectors of all suffixes of x and y.

Let \$ be a new symbol that occurs only once at the beginning of x and y. The algorithm builds a list L of length $\leq |\Sigma|+1$ containing pairs of suffixes (x_i, y_j) such that $x_i \overset{\exists}{>}_{rpo} y_j$.

- Initialize list L to empty;
- For $i = 1$ to $|x|$:
  1. If $x_i = \mu(a,x)$ for some a, then continue else go to the next i.
  2. Let $y_j = \mu(a,y)$.
  3. If $\#_b(x_i) > \#_b(y_j)$ for any symbol b, then add (x_i, y_j) to L.
  4. Otherwise compute Δ_{x_i,y_j} and if $(\mu(c,x_i), \mu(c,y_j)) \in L$ for any symbol $c \in \Delta_{x_i,y_j}$ then add (x_i, y_j) to L.
  5. If $(\mu(\$,x), \mu(\$,y)) \in L$ then **True** else **False**

The algorithm runs in quadratic time: The sets of vectors $\pi(x_i)$ and $\pi(y_i)$ can be computed in quadratic time. Then we can compute a table $(i,j) \mapsto \Delta_{x_i,y_j}$ in quadratic time. Therefore we can conclude:

Theorem 1. *There is a polynomial time algorithm solving the SYMBOL-ORDER problem for one-rule string-rewriting systems.*

We add a few more results which may help in improving the performance of the algorithm.

Lemma 11. *Let x and y be two strings. If the Δ-skeletons of x and y are different, then $x \overset{\exists}{>}_{rpo} y$.*

Proof. Let x' and y' be the Δ-skeletons of x and y respectively. Since x' and y' are different but have same Parikh vectors, there is a symbol ordering $\succ$ on Δ such that $x' >_{\text{rpo}} y'$. Let us extend the symbol ordering on Δ to Σ in such a way that for all $a \in \Delta$ and all $b \in \Sigma \setminus \Delta$ we have $b \prec a$.

Claim: If $u', v', z \in \Delta^*$, $w' \in (\Sigma \setminus \Delta)^*$ and $z >_{rpo} u'v'$ then $z >_{rpo} u'w'v'$.

We can write $x = w_0u_1w_1 \ldots u_nw_n$ (resp., $y = w'_0u'_1w'_1 \ldots u'_nw'_n$) where $u_1, u'_1, u_2, u'_2, \ldots, u_n, u'_n \in \Delta$ and $w_0, w'_0, w_1, w'_1, \ldots, w_n, w'_n \in (\Sigma \setminus \Delta)^*$. Note that $x >_{rpo} x'$ since $\succ_{\text{rpo}}$ contains the embedding relation [2]. Then $x >_{\text{rpo}} y$ by iterated applications of the claim to x, y'. □

Lemma 12. *Let x and y be two* distinct *strings such that $\#_a(x) \leq \#_a(y)$ for all $a \in \Sigma$. If the Δ-skeletons of x and y are the same, then $x \overset{\exists}{>}_{rpo} y$ if and only if there is a symbol $c \in \Delta$ such that $\mu(c,x) >_{rpo} \mu(c,y)$ for some symbol ordering on $\Sigma \setminus \{c\}$.*

Lemma 13. *Let x and y be two* distinct *non-suffix-sharing strings such that* (a) *$\#_a(x) \leq \#_a(y)$ for all $a \in \Sigma$,* (b) *$\Delta \neq \emptyset$, and* (c) *the Δ-skeletons of x and y are the same. Then $x \overset{\exists}{>}_{rpo} y$* if *$\pi(\mu(c,x) \not\leq_V \pi(\mu(c,y)$ for some $c \in \Delta$.*

This condition is the same as $\exists c \in \Delta\, \exists a \in (\Sigma \setminus \Delta) : \#_a(\mu(c,x)) > \#_a(\mu(c,y))$.

Note that the other direction does not work: consider $x = cad$ and $y = adcda$. We get $x >_{rpo} y$ if we choose $c \succ a \succ d$. But $\pi(\mu(c,x)) = (1,1) \leq (1,1) = \pi(\mu(c,y))$.

References

[1] F, Baader and T. Nipkow. *Term rewriting and all that*. Cambridge University Press, 1998.

[2] N. Dershowitz. Orderings for term-rewriting systems. *Theoretical Computer Science* 17(3): 279–301. 1982.

[3] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, first edition, 1979.

[4] M. S. Krishnamoorthy and P. Narendran. On recursive path ordering. *Theoretical Computer Science,* 40:323–328, 1985.

[5] P. Narendran and M. Rusinowitch. The theory of total unary RPO is decidable. *Computational Logic – CL 2000, First International Conference, London, UK, 24-28 July, 2000, Proceedings*, volume 1861 of *Lecture Notes in Computer Science*, pages 660–672. Springer, 2000.

[6] P. Schneider-Kamp, R. Thiemann, E. Annov, M. Codish, J. Giesl: Proving Termination Using Recursive Path Orders and SAT Solving. Frontiers of Combining Systems, 6th International Symposium, FroCoS 2007, volume 4720 of *Lecture Notes in Computer Science*, pages 267-282. Springer 2007.

[7] M. Sipser. *Introduction to the Theory of Computation*. Course Technology, Boston, MA, Third edition, 2013.