Inferring RPO Symbol Ordering

Wei Du [1] Paliath Narendran [1] Michael Rusinowitch [2]

[1]University at Albany–SUNY, Albany, NY, USA
e-mail: {wdu2, pnarendran}@albany.edu

[2]LORIA-INRIA Nancy-Grand Est
e-mail: rusi@loria.fr

Outline

- Introduction
- NP-Completeness
- The One Rule Case

Background

- Termination is a property of term rewriting system (TRS).
- Term orderings are important to show termination of TRS.
- Dershowitz introduced recursive path ordering (RPO).

We study RPO in the context of string rewriting systems (SRS).

RPO Definition on Strings

Definition 1

Let Σ be an alphabet and $\succ$ be an ordering on Σ.
Then $x >_{rpo} y$ iff any of these is true:

(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 $x >_{rpo} v$.

Example: $BAB >_{rpo} ACC$ with the ordering $A \succ B \succ C$
Hint: first use $(3a)$, then (2), $(3b)$, $(3b)$, finally (1).

RPO Characterization when $\succ$ is Total

Definition 2

Let $max(w, \Sigma)$ denote the *maximal* symbol of Σ that occurs in w.
$mul(w, \Sigma)$ be the number of times *this* max symbol occurs in w.
$w >_{rpo} w'$ iff any 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. $a = max(w, \Sigma) = max(w', \Sigma)$, $mul(w, \Sigma) = mul(w', \Sigma)$,

$$w = w_0 a w_1 a w_2 \ldots a w_k$$

$$w' = u_0 a u_1 a u_2 \ldots a u_k$$

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

RPO Property on Strings

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. $a = max(w, \Sigma) = max(w', \Sigma)$, $mul(w, \Sigma) = mul(w', \Sigma)$, and $\mu(a, w) >_{rpo} \mu(a, w')$ where $\mu(a, w)$ is the longest suffix of w that does not contain a.

Example: $BAB >_{rpo} ACC$ with the ordering $A \succ B \succ C$
Hint: first use case 3 where $\mu(A, BAB) = B$ and $\mu(A, ACC) = CC$, then use case 1 where $B \succ C$.

The SYMBOL-ORDER Problem

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 $l_i >_{rpo} r_i$ for all i?

For arbitary TRSs, this is known to be NP complete (M.S. Krishnamoorthy and P. Narendran).

How about for SRS?

2-3-SAT Problem is NP-Complete

- 2-3-SAT: A set contains 2-clauses (with only negative literals) and 3-clauses (with only positive literals).
- 2-3-SAT problem is NP-complete by a reduction from 3-SAT:
  Let L be the set of literals and let C be the set of clauses.
  1. Replace all negative literals in the 3-SAT as shown below:
     $\forall \neg a \in L$: let $L \leftarrow (L \setminus \neg a) \cup a'$ and $C \leftarrow C \cup \{\neg a \vee \neg a', a \vee a'\}$
  2. Replace the clause $(a \vee a')$ above by clauses with new literals:
     Let $L \leftarrow L \cup \{z_1, z_2\}$ and
     $C \leftarrow (C \setminus a \vee a') \cup \{a \vee a' \vee z_1, a \vee a' \vee z_2, \neg z_1 \vee \neg z_2\}$

SYMBOL-ORDER Problem is NP-Complete

Reduce the 2-3-SAT problem to the SYMBOL-ORDER problem:

- Let Φ be any CNF formula as stated in the 2-3-SAT problem.
- For each variable x_i in Φ, we introduce a symbol a_i.
- We then use symbol d to simulate truth and falsehood of a variable: x_i is true iff $a_i \succ d$, and x_i is false iff $a_i \prec d$.
- 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

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

The One Rule Case: Definitions

- We use $x \overset{\exists}{>}_{rpo} y$ to denote "there is an ordering $\succ$ on Σ such that $x >_{rpo} y$".
- For two strings x, y, we define

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

- 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 One Rule Case: Properties

- 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 $x \overset{\exists}{\not>}_{rpo} y$.

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

The One Rule Case: Algorithm

Theorem

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

As a preprocessing step remove common suffixes first.

Then compute Parikh vectors of all suffixes of x and y.

Let \$ be a new symbol occurs only 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 $x_i \overset{\exists}{>}_{rpo} y_j$.

The One Rule Case: Algorithm Continued

- 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.
- If $(\mu(\$, x), \mu(\$, y)) \in L$ then **True** else **False**

The One Rule Case: Algorithm Analysis

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 proved the theorem by construction.

References I

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

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

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.

Questions

Thank you!