Fachbereich Informatik
Universität Kaiserslautern
Postfach 3049
D-6750 Kaiserslautern

SEKI - REPORT

Comparing on Strings: Iterated Syllable Ordering and Recursive Path Ordering

Joachim Steinbach

SEKI Report SR-89-15

Comparing on Strings: Iterated Syllable Ordering and Recursive Path Ordering

Joachim Steinbach
SEKI Report SR-89-15

Comparing on strings : Iterated syllable ordering & Recursive path ordering

Joachim Steinbach

Universität Kaiserslautern
Fachbereich Informatik
Postfach 3049
D - 6750 Kaiserslautern (FRG)

This research was supported by the Deutsche Forschungsgemeinschaft, SFB 314 (D4)

Abstract

The recursive path ordering introduced by Dershowitz can prove the termination of term rewriting systems. On string rewriting systems, the iterated syllable (or collecting) ordering is a total ordering. We will prove that the recursive path ordering on monadic terms is equivalent to the iterated syllable ordering on the reverse words.

1. Motivation and Notation

There has been considerable interest in rewrite systems because they are a useful model for non-deterministic computations with various applications including automatic theorem proving, program verification and synthesis, abstract data type specifications and algebraic simplification. Such systems may take the form of term rewriting systems [cf. [AM89]], string rewriting systems [cf. [Bo87], [KN85]], etc.

A term rewriting system $\mathfrak{R}$ over a set of terms Γ is a finite set of rules, each of the form $l \rightarrow r$, where l and r are terms in Γ. The set Γ of all terms is constructed from elements of a set $\mathfrak{F}$ of operators [or function symbols] and some denumerably infinite set $\mathfrak{V}$ of variables. The leading function symbol and the multiset of the [direct] arguments of a term t are referred to by top[t] and arg[t], respectively.

A Thue system $\mathfrak{T}$ over a set of strings Σ^* is a finite set of rules, each of the form $l \rightarrow r$, where l and r are words in Σ^*. Σ^* is the monoid freely generated by a finite alphabet Σ under the operation of concatenation, i.e. the set of all finite strings over Σ. The empty string ε is the identity in the monoid. The length function on strings, denoted by $|u|$, is defined as usual. Especially, $|u|_a$ is the number of occurrences of the letter a in u. Synonymously to terms, top and arg denote the first letter and the rest of a word, respectively.

A close relation between term rewriting and Thue systems will exist if monadic terms are used only. A monadic term only contains unary function symbols and either a constant or a variable. The subset of the monadic terms without constants can unequivocally be transformed into strings and vice versa: Let be $\mathfrak{F} = \{f_1,...,f_n\}$ and $\Sigma = \{a_1,...,a_n\}$, t and $t' \in \Gamma[\mathfrak{F},\{x\}]$, u and $u' \in \Sigma^*$.

- $\tau_1 : \Gamma \rightarrow \Sigma^*$

$$\tau_1[t] = \begin{cases} \varepsilon & \text{if } t = x \in \mathfrak{V} \\ a_i \cdot \tau_1[t'] & \text{if } t = f_i[t'] \end{cases}$$

For example, $\tau_1[f_1[f_1[f_2[x]]]] = a_1a_1a_2$.

- $\tau_2 : \Sigma^* \rightarrow \Gamma$

$$\tau_2[u] = \begin{cases} f_i[x] & \text{if } u = a_i \\ f_i[\tau_2[u']] & \text{if } u = a_i \cdot u',\ u' \neq \varepsilon \end{cases}$$

For example, $\tau_2[a_1a_2a_1] = f_1[f_2[f_1[x]]]$.

An effective calculation with rewrite systems presumes termination. A great number of orderings have been defined. Most of them are so-called precedence orderings using a special ordering on operators (letters). More precisely, a precedence is a partial ordering $\triangleright$ on $\mathfrak{F}$ (Σ) which is an irreflexive and transitive binary relation.

This report deals with two orderings based on a precedence, the recursive path ordering of Dershowitz ([De82]) on term rewriting systems and the iterated syllable (or collecting) ordering (see for example [Si87], [Wi88], [AW89]) on string rewriting systems. The definitions of these orderings are the main constituents of the next two chapters. The main result of this paper consists of a comparison between these orderings. In chapter 4, we will prove that the two orderings are equal if they are restricted to strings.

2. RPO on strings

The comparison w.r.t. the recursive path ordering (RPO, for short) is based on the following idea: A term is decreased by replacing a subterm with any numbers of smaller terms which are connected by any structure of operators smaller (w.r.t. a precedence $\triangleright$) than the leading function symbol of the replaced subterm. The relationship between these operators w.r.t. $\triangleright$ is responsible for decreasing one of the (or both) terms in the recursive definition of the RPO. If one of the terms is 'empty' (i.e. totally decreased) then the other one is greater.

Definition 2.1 (The recursive path ordering, [De82])

Let be s and t two terms and $\triangleright$ any partial ordering on the operators. Then,

$$
\begin{array}{lll}
s >_{RPO} t & & \\
\text{iff} & \text{i)} \quad top(s) \triangleright top(t) & \wedge \quad \{s\} \gg_{RPO} arg(t) \\
\text{or} & \text{ii)} \quad top(s) = top(t) & \wedge \quad arg(s) \gg_{RPO} arg(t) \\
\text{or} & \text{iii)} \quad arg(s) \geq_{RPO} \{t\} &
\end{array}
$$

□

Two terms are considered equivalent w.r.t. the RPO if they are permutationally congruent, i.e. they are the same except for permutations among subterms (e.g. $f(x,y) =_{RPO} f(y,x)$). $\gg_{RPO}$ is the extension of $>_{RPO}$ to multisets of terms. Multisets are like sets, but allow multiple occurrences of identical terms. The extension of $>_{RPO}$ on multisets is defined as follows: A multiset S is greater than (w.r.t. the RPO) a multiset T ($\neq$ S) over Γ, denoted by

$$S \gg_{RPO} T \quad \text{iff} \quad (\forall t \in T \setminus S)\ (\exists s \in S \setminus T)\ s >_{RPO} t$$

i.e. $S \gg_{RPO} T$ if T can be obtained from S by replacing one or more terms in S by any finite number of terms, each of which is smaller (w.r.t. $>_{RPO}$) than one of the replaced terms.

Example 2.2

We would like to prove that the distributive law $x*(y+z) \rightarrow (x*y)+(x*z)$ terminates. We use the total precedence $* \triangleright +$. Therefore, we must show that $\{s\} \gg_{RPO} arg(t)$, i.e. $\{s\} \gg_{RPO} \{x*y , x*z\}$. The single term on the left side has to be greater than both terms on the right side: s is greater than x*y because we have to remove the leading function symbols and can show that $\{x , y+z\} \gg_{RPO} \{x , y\}$ because (after removing x) $y+z >_{RPO} y$ by using the subterm property of the RPO (2.1 iii). $s >_{RPO} x*z$ is proved in the same way.

□

Note that there are several extensions of this ordering. The reader is referred to [De87] or [St89] for an overview. However, we would like to consider a special version of the RPO: the RPO on strings. Since $\mathfrak{F}$ consists of only unary operators , the multiset extension is superfluous.

Lemma 2.3 [The recursive path ordering on strings]

Let be u,v two words over Σ^* and $\triangleright$ any partial ordering on Σ. Then,

$u >_{RPO} v$
iff i) $top[u] \triangleright top[v] \quad \wedge \quad u >_{RPO} arg[v]$
or ii) $top[u] = top[v] \quad \wedge \quad arg[u] >_{RPO} arg[v]$
or iii) $arg[u] \geq_{RPO} v$

Proof: obvious

□

Example 2.4

The rule abc → cba is terminating since there is an RPO which shows this property. Assuming $b \triangleright c \triangleright a$, $abc >_{RPO} cba$ iff $bc \geq_{RPO} cba$ iff $bc \geq_{RPO} ba$ iff $c \geq_{RPO} a$ which is valid since $c \triangleright a$.

□

3. The iterated syllable ordering

In contrast to term rewriting systems, the uniform termination property of Thue systems is decidable. There exist several total orderings which guarantee this property, e.g.

- The ordering in which words are ordered first by length and, if the lengths are equal, lexicographically according to the precedence.
- The ordering of Knuth and Bendix assigns natural numbers to the elements of Σ and then to words by adding the numbers of the letters (called weight) they contain. Two words are compared by comparing their weights, and if the weights are equal, by comparing the top symbols w.r.t. the precedence.

Another well-known ordering on strings is the so-called iterated syllable (or collecting) ordering (see [Si87], [Wi88], [AW89]). To describe this strategy we need some helpful definitions.

Definition 3.1 (Lexicographic extension, Syllabic decomposition)

- Let be $>$ an ordering on strings and $u_1,\ldots,u_p,v_1,\ldots,v_q$ words over Σ^*. Then,

$$(u_1, u_2, \ldots, u_p) >^{lex} (v_1, v_2, \ldots, v_q)$$
$$\text{if either} \quad p > 0 \wedge q = 0$$
$$\text{or} \quad u_1 > v_1$$
$$\text{or} \quad u_1 = v_1 \wedge (u_2, \ldots, u_p) >^{lex} (v_2, \ldots, v_q)$$

  is the lexicographic extension of $>$ on tuples of words.

- Let be $u = u_0 a u_1 a \ldots a u_k$ a word with $k \geq 0$, $u_i \in \Sigma^*$ and $a \in \Sigma$. Then,

$$dec(u, a) = (u_0, u_1, \ldots, u_k)$$

  is the syllabic decomposition of u w.r.t. the letter a.

□

We will present a slightly modified version of the original ordering contained in [Si87] (cf. [Wi88], [AW89]). Both orderings are known to be equivalent (see [Wi88]).

Definition 3.2 (The iterated syllable ordering)

Let be u, v two words over Σ^*. Furthermore, a total precedence $\triangleright$ is given. Then,

$$u >_{SYL} v \quad \text{iff} \quad |u|_a > |v|_a \quad \text{or} \quad |u|_a = |v|_a \wedge dec(u,a) >^{lex}_{SYL} dec(v,a)$$

such that a is the greatest (w.r.t. $\triangleright$) letter in u or v.

□

This ordering was already used implicitly in collecting algorithms solving the generalized word problem of polycyclic groups. The normal form of a string is exactly the corresponding element being minimal relative to the collecting ordering.

Furthermore, Bauer ([Ba81]) applied the basis (without iteration) of this ordering: An element u is greater than v if $|u|_a > |v|_a$ or $|u|_a = |v|_a \wedge dec(u,a) \succ^{lex} dec(v,a)$ where a is a special letter and $\succ$ is the ordering on the length of elements ($u \succ v$ iff $|u| > |v|$). Note that the iterated syllable ordering will be well-founded if this ordering has this property since a lexicographic ordering of fixed-length tuples will be well-founded if the orderings on components are (see [De83]).

Example 3.3 ([Si87])

We will prove the termination of the rule u = baca → caba = v with the help of the iterated syllable ordering based on the precedence $a \triangleright b \triangleright c$. Since $|u|_a = |v|_a = 2$, the syllabic decompositions w.r.t. a must be compared : $(b , c) >^{lex}_{SYL} (c , b)$ because $|b|_b = 1 > 0 = |c|_b$.
Note that $caba >_{SYL} b...b$ since b...b does not contain any a. This is the basic concept of multiset orderings (see chapter 2).

□

4. Comparison and Conclusion

In this chapter we compare the power of the presented orderings restricted to strings. Note that the RPO as well as the SYL are reduction orderings.

Restricted to strings, the recursive path ordering on reverse words and the iterated syllable ordering are equivalent.

Definition 4.1 [Reversal of a string]

Let be $u = a_1a_2...a_n \in \Sigma^*$:

$$\rho[u] = a_n...a_2a_1$$

is the reversal of u.

□

Theorem 4.2

Let be u, v $\in \Sigma^*$, $\triangleright$ a total precedence:

$$u >_{SYL} v \quad \text{iff} \quad \rho[u] >_{RPO} \rho[v].$$

Proof:

"$\rightsquigarrow$": We will prove this statement by induction on |u|+|v|. Let be a the greatest letter occurring in u or v.

i) $|u|_a > |v|_a$
$\rightsquigarrow$ $u = u_0au_1a...au_m$, $v = v_0av_1a...av_n$ and $m > n$
Let be $m = n + k$, $k > 0$.

Note that $\rho[u_{n+k}]a...a\rho[u_1]a\rho[u_0] >_{RPO} \rho[v_n]a...a\rho[v_1]a\rho[v_0]$
if
$\rho[u_{n+k-1}]a...a\rho[u_1]a\rho[u_0] >_{RPO} \rho[v_{n-1}]a...a\rho[v_1]a\rho[v_0]$
since $\rho[v_n]$ could be eliminated by applying 2.3 i), $\rho[u_{n+k}]$ could be removed by applying 2.3 iii) and both a's can be removed with the help of 2.3 ii).

By induction on the index of u and v, respectively, we have to verify

$$\rho[u_k]a...a\rho[u_0] >_{RPO} \rho[v_0]$$

which is valid since a is greater [w.r.t. $\triangleright$] than all symbols occurring in $\rho[v_0]$.

ii) $|u|_a = |v|_a \quad \wedge \quad dec(u,a) >^{lex}_{SYL} dec(v,a)$
$\rightsquigarrow (u_0, u_1, \ldots, u_n) >^{lex}_{SYL} (v_0, v_1, \ldots, v_n)$

W.l.o.g. let be $u_0=v_0 \ldots u_{i-1}=v_{i-1} \quad \wedge \quad u_i >_{SYL} v_i$. We have to show that

$$\rho(u_n)a\rho(u_{n-1})a\ldots a\rho(u_0) >_{RPO} \rho(v_n)a\rho(v_{n-1})a\ldots a\rho(v_0).$$

This is valid if (see i)
$$\rho(u_i)a\ldots a\rho(u_0) >_{RPO} \rho(v_i)a\ldots a\rho(v_0)$$
which is equivalent to
$$\rho(u_i)w >_{RPO} \rho(v_i)w \qquad , \qquad w \in \Sigma^*$$
since $u_0=v_0 \ldots u_{i-1}=v_{i-1}$.

This is valid since, by induction hypothesis, $\rho(u_i) >_{RPO} \rho(v_i)$ and the 'subterm' property of the RPO.

"$\lessdot\!\sim$": Analogous with the other direction we will prove this assertion by induction on $|u|+|v|$. Since $\rho(\rho(u)) = u$, we have to show that $u >_{RPO} v$ implies $\rho(u) >_{SYL} \rho(v)$.

i) $top(u) \triangleright top(v)$

$\rightsquigarrow u >_{RPO} arg(v)$
by definition of the RPO (lemma 2.3 i)

$\rightsquigarrow \rho(u) >_{SYL} \rho(arg(v))$
by induction hypothesis

Let be $top(u) = a$, $top(v) = b$. Therefore, we have to prove that
$\rho(u) >_{SYL} \rho(arg(v))b$
which is equivalent to
$\rho(arg(u))a >_{SYL} \rho(arg(v))b$.

Let be c the greatest letter occurring in u or v. Furthermore, let be $arg(u) = u_m c\ldots cu_1 cu_0$ and $arg(v) = v_n c\ldots cv_1 cv_0$:

- $|\rho(arg(u))|_c > |\rho(arg(v))|_c$
  $\rightsquigarrow |\rho(u)|_c > |\rho(v)|_c$
  since $|\rho(v)|_c = |\rho(arg(v))|_c$ (because $a \triangleright b$ and therefore, $c \triangleright b$)

- $|\rho(arg(u))|_c = |\rho(arg(v))|_c \quad \wedge \quad \rho(u_i) >_{SYL} \rho(v_i) \quad \wedge \quad i < n$
  $\rightsquigarrow \rho(u) >_{SYL} \rho(v)$
  since $i < n \quad \wedge \quad a \triangleright b$

- $|\rho[arg[u]]|_c = |\rho[arg[v]]|_c \quad \wedge \quad \rho[u_n] >_{SYL} \rho[v_n]$
  $\rightsquigarrow \rho[u_n]a >_{SYL} \rho[v_n]b$
  since $|\rho[u_n]a|_a > |\rho[v_n]b|_a \quad \wedge \quad a \triangleright b$

ii] $top[u] = top[v] =: a$

$\rightsquigarrow arg[u] >_{RPO} arg[v]$
by definition of the RPO [lemma 2.3 ii]

$\rightsquigarrow \rho[arg[u]] >_{SYL} \rho[arg[v]]$
by induction hypothesis

$\rightsquigarrow \rho[arg[u]]a >_{SYL} \rho[arg[v]]a$
by definition of the SYL

iii]$top[u] \triangleleft top[v]$

$\rightsquigarrow arg[u] \geq_{RPO} v$
by definition of the RPO [lemma 2.3 iii]

Let be $a := top[u]$:

- $arg[u] =_{RPO} v$
  $\rightsquigarrow arg[u] = v$
  by definition of the RPO
  $\rightsquigarrow \rho[arg[u]] = \rho[v]$
  by definition of ρ
  $\rightsquigarrow \rho[arg[u]]a >_{SYL} \rho[v]$
  by definition of the SYL
  $\rightsquigarrow \rho[u] >_{SYL} \rho[v]$

- $arg[u] >_{RPO} v$
  $\rightsquigarrow \rho[arg[u]] >_{SYL} \rho[v]$
  by induction hypothesis
  $\rightsquigarrow \rho[arg[u]]a >_{SYL} \rho[v]$
  since $|\rho[arg[u]]a|_a > |\rho[arg[u]]|_a$ and $>$ is a partial ordering

□

Note that the same relationship between the SYL and the RPO will exist if the precedence is quasi-total. As usual, a quasi-ordered set $(\Sigma^*,\geq)$ consists of the set Σ^* and a transitive and reflexive binary relation $\geq$ defined on elements of Σ^*. A quasi-ordering defines an equivalence relation $=$ as both $\geq$ and $\leq$, and a partial ordering $>$ as $\geq$ but not $\leq$.

Lemma 4.3

Let be u, v $\in \Sigma^*$, $\trianglerighteq$ a quasi-total precedence:

$$u >_{SYL} v \quad \text{iff} \quad \rho(u) >_{RPO} \rho(v).$$

Proof: analogous with the proof of theorem 4.2

□

In conclusion, we would like to point out that other well-known path (and decomposition) orderings are also equivalent (in the same sense as the RPO) to the iterated syllable ordering. This assertion is based on the following lemma:

Lemma 4.4 [St89]

Let be s, t monadic terms and $\triangleright$ a total precedence. Then,

$$\begin{aligned} s >_{RPO} t \quad & \text{iff} \quad s >_{PSO} t \\ & \text{iff} \quad s >_{RDO} t \\ & \text{iff} \quad s >_{KNS} t. \end{aligned}$$

□

PSO is the path of subterms ordering of Plaisted, RDO is the abbreviation of 'recursive decomposition ordering' developed by Lescanne and KNS stands for the path ordering of Kapur, Sivakumar and Narendran. All these orderings are described in [De87] and [St89].

Acknowledgement

There remains the pleasant duty to express my appreciation to J. Avenhaus, K. Madlener and I. Sonntag for helping me with this work.

References

[AM89] J. Avenhaus, K. Madlener
Term rewriting and equational reasoning
In: Formal Techniques in Artificial Intelligence - A source book, R.B. Banerji (ed.), Academic Press, 1989

[AW89] J. Avenhaus, D. Wissmann
Using rewriting techniques to solve the generalized word problem in polycyclic groups
Proc. ISSAC, Portland, 1989

[Ba81] G. Bauer
Zur Darstellung von Monoiden durch konfluente Regelsysteme
Dissertation, Fachbereich Informatik, Universität Kaiserslautern, W. Germany, Februar 1981

[Bo87] R.V. Book
Thue systems as rewriting systems
J. Symbolic Computation 3, 1987

[De87] N. Dershowitz
Termination of rewriting
J. Symbolic Computation 3, 1987

[De83] N. Dershowitz
Well-founded orderings
Technical Report ATR-83(8478)-3, Information Sciences Research Office, The Aerospace Corporation, El Segundo, California, May 1983

[De82] N. Dershowitz
Orderings for term rewriting systems
J. Theoretical Computer Science, Vol. 17, No. 3, March 1982

[KN85] D. Kapur, P. Narendran
The Knuth-Bendix completion procedure and Thue systems
SIAM J. Computing, Vol. 14, No. 4, November 1985

[Si87] C.C. Sims
Verifying nilpotence
J. Symbolic Computation 3, 1987

[St89] J. Steinbach
Extensions and comparison of simplification orderings
Proc. 3rd RTA, Chapel Hill, LNCS 355, 1989

[Wi88] D. Wissmann
Applying rewriting techniques to groups with power-commutation presentations
Proc. Int. Symposium on Symbolic and Algebraic Computation, Rome, July 1988