Acta Cybernetica 14 (2000) 461-468.

Unusual Algorithms for Lexicographical Enumeration*

Pál Dömösi †

Abstract

Using well-known results, we consider algorithms for finding minimal words of given length n in regular and context-free languages. We also show algorithms enumerating the words of given length n of regular and context-free languages in lexicographical order.

1 Introduction

E. Mäkinen [8] described algorithms to find the lexicographically minimal words for regular and context-free grammars. Using well-known recent results in [1, 2, 3, 6, 7, 9], we show similar algorithms. E. Mäkinen [8] presents also an algorithm to enumerate the words of a regular language in lexicographical order. We give another algorithm for lexicographical enumeration of regular languages. In addition, using an extension of the well-known CYK-algorithm, we show an algorithm to enumerate the words of length n of a context-free language in lexicographical order. Using the well-known Valiant algorithm, see [11, 5], a little refinement of our solution is attainable.

2 Preliminaries

A *word* (over Σ) is a finite sequence of elements of some finite non-empty set Σ. We call the set Σ an *alphabet,* the elements of Σ *letters.* If u and v are words over an alphabet Σ, then their *catenation* uv is also a word over Σ. Then we also say that u is a *prefix* of uv. In particular, for every word u over Σ, $u\lambda = \lambda u = u$, where λ denotes the *empty word.* Given a word u, we define $u^0 = \lambda$, $u^n = u^{n-1}u$, $n > 0$, $u^* = \{u^n \mid n \geq 0\}$ and $u^+ = u^* \setminus \{\lambda\}$. In addition, we put $\Sigma^n = \{w \in \Sigma \mid |w| = n\}$.

The *length* $|w|$ of a word w is the number of letters in w, where each letter is counted as many times as it occurs. Thus $|\lambda| = 0$. By the *free monoid* Σ^* *generated*

*The research has been supported by the Academy of Finland grant 137358 and the Hungarian National Foundation for Scientific Research grants OTKA-T019392 and OTKA-T030140.

†L. Kossuth University, Institute of Mathematics and Informatics, 4032 Debrecen, Egyetem tér 1, Hungary, domosi@math.klte.hu

by Σ we mean the set of all words (including the *empty word* λ) having catenation as multiplication. We set $\Sigma^+ = \Sigma^* \setminus \{\lambda\}$, where the subsemigroup Σ^+ of Σ^* is said to be *free semigroup generated by* Σ. Subsets of Σ^* are referred to as *languages* over Σ. Given a set Σ, let $card(\Sigma)$ denote the cardinality of Σ. A language $L \subseteq \Sigma^*$ is said to be k-*slender* if $card\{w \in L \mid |w| = n\} \leq k$, for every $n \geq 0$. A language is *slender* if it is k-slender for some positive integer k. A 1-slender language is called *thin* language. A language L is said to be a *union of single loops* (or, in short, USL) if for some positive integer k and words $u_i, v_i, w_i, 1 \leq i \leq k$,

$$(*) \quad L = \bigcup_{i=1}^{k} u_i v_i^* w_i.$$

L is called a *union of paired loops* (or UPL, in short) if for some positive k and words $u_i, v_i, w_i, x_i, y_i, 1 \leq i \leq k$,

$$(**) \quad L = \bigcup_{i=1}^{k} \{u_i v_i^n w_i x_i^n y_i \mid n \geq 0\}.$$

For a USL (or UPL) language L the smallest k such that $(*)$ (or $(**)$) holds is referred to as the USL-index (or UPL-index) of L. A USL language L is said to be a *disjoint union of single loops* (DUSL, in short) if the sets in the union $(*)$ are pairwise disjoint. In this case the smallest k such that $(*)$ holds and the k sets are pairwise disjoint is referred to as the DUSL-index of L. The notions of a *disjoint union of paired loops* (DUPL) and DUPL-index are defined analogously considering $(**)$. We shall use the following well-known results.

Theorem 2.1 *[9] The next conditions, (i)-(iii), are equivalent for a language L.*
(*i*) *L is regular and slender.*
(*ii*) *L is USL.*
(*iii*) *L is DUSL.*

□

Theorem 2.2 *[9] Every UPL language is DUPL, slender, linear and unambiguous.*

□

Theorem 2.3 *[6, 10] Every slender context-free language is UPL.* □

We will use the following extension of Theorem 2.3.

Theorem 2.4 *[7] A given slender context-free language can be effectively written as a disjoint union of (finitely many) paired loops.* □

The next statement is a direct consequence of the constructive proof of Theorem 2.1 in [9].

Theorem 2.5 *A given slender regular language can be effectively written as a disjoint union of (finite many) single loops.* □

3 Finding minimal words of given length and enumeration of regular languages

Given a total order $\prec$ on Σ, a *lexicographical order* on Σ^* is defined as an extension of $\prec$ to Σ^* such that for any pair $u, v \in \Sigma^*$, $u \prec v$ if and only if either $v = uu', u' \in \Sigma^+$ or $u = wxu'$, $v = wyv'$, $x \prec y$ for some $w, u', v' \in \Sigma^*$ and $x, y \in \Sigma$. We will denote by $first(\Sigma)$ the first element of Σ under $\prec$. Moreover, for every $u \in \Sigma^*$ we put $next(u) = \min\{v \mid v \in \Sigma^*, u \prec v\}$. In addition, we put $Pref(L) = \{v \mid \exists u \in L,\ v' \in \Sigma^*, u = vv'\}$. Thus $Pref(L)$ denotes the set of all prefixes of words in L.

Given a language L, the language L_{min} is defined by taking from all words of L of the same length only the first one in lexicographic order. Of course, L_{min} is a thin language. We shall use the following results.

Theorem 3.1 *[1, 4] For every regular language L, the language L_{min} is regular, and a regular grammar for it can be effectively constructed.* □

Theorem 3.2 *[2] For every context-free language L, the language L_{min} is context-free. Moreover, given a context-free grammar generating L, a context-free grammar for L_{min} can be effectively constructed.* □

Using Theorem 3.1 and Theorem 3.2, together with Theorem 2.5 and Theorem 2.4, the following algorithms can be constructed.

Algorithm regmin
Input: A regular grammar $G = (V, \Sigma, P, S)$ and a total order $\prec$ on Σ.
Output: A finite language $L_G = \{u_1, v_1, w_1, \ldots, u_{n(G)}, v_{n(G)}, w_{n(G)}\}$ having

$$L_{min} = \bigcup_{i=1}^{n(G)} \{u_i v_i^n w_i \mid n \geq 0\}.$$

End of algorithm regmin

Algorithm cfmin
Input: A context-free grammar $G = (V, \Sigma, P, S)$ and a total order $\prec$ on Σ.

Output: A finite language $L_G = \{u_1, v_1, w_1, x_1, y_1, \dots, u_{n(G)}, v_{n(G)}, w_{n(G)}, x_{n(G)}, y_{n(G)}\}$ having

$$L_{min} = \bigcup_{i=1}^{n(G)} \{u_i v_i^n w_i x_i^n y_i \mid n \geq 0\}.$$

End of algorithm cfmin

On the basis of the above observations, we now show how to construct the following algorithms.

Algorithm REGMIN

Input: A regular grammar $G = (V, \Sigma, P, S)$, a total order $\prec$ on Σ and a positive integer n.

Output: A finite language $L_G = \{u_1, v_1, w_1, \dots, u_{n(G)}, v_{n(G)}, w_{n(G)}\}$ (having $L_{min} = \bigcup_{i=1}^{n(G)} \{u_i v_i^n w_i \mid n \geq 0\}$) and

- a pair k, ℓ of positive integers such that $1 \leq k \leq n(G)$ if the word of length n of L_{min} exists and it has the form $u_k v_k^\ell w_k$;
- an error message if L_{min} has no word of length n.

Method: Apply the algorithm **regmin**; $k, \ell \leftarrow 0$;
for $i \leftarrow 1 \dots n(G)$ **do**
  if the equation $|u_i w_i| + |v_i|\alpha = n$ has a positive integer solution for α
  then $k \leftarrow i; \ell \leftarrow \alpha$;
od
Output:

- k, ℓ, if $k > 0$;
- an error message if $k = 0$;

End of algorithm REGMIN

Algorithm CFMIN

Input: A context-free grammar $G = (V, \Sigma, P, S)$, a total order $\prec$ on Σ and a positive integer n.

Output: A finite language $L_G = \{u_1, v_1, w_1, x_1, y_1, \dots, u_{n(G)}, v_{n(G)}, w_{n(G)}, x_{n(G)}, y_{n(G)}\}$ (having $L_{min} = \bigcup_{i=1}^{n(G)} \{u_i v_i^n w_i x_i^n y_i \mid n \geq 0\}$) and

- a pair k, ℓ of positive integers such that $1 \leq k \leq n(G)$ if the word of length n of L_{min} exists and it has the form $u_k v_k^\ell w_k x_k^\ell y_k$;
- an error message if L_{min} has no word of length n.

Method:
Apply the algorithm **cfmin**; $k, \ell \leftarrow 0$;
for $i \leftarrow 1 \dots n(G)$ **do**

if the equation $|u_i w_i y_i| + |v_i x_i|\alpha = n$ has a positive integer solution for α
then $k \leftarrow i; \ell \leftarrow \alpha$;
od
Output:

- k, ℓ, if $k > 0$;
- an error message if $k = 0$;

End of algorithm CFMIN

It is well-known that for every pair of regular grammars G_1, G_2, a regular grammar G having $L(G) = L(G_1) \setminus L(G_2)$ can be effectively constructed. Therefore, by Theorem 3.1 and Theorem 2.5, we can consider the following idea for enumerating the words of length n in $L(G)$ in lexicographical order having a regular grammar G. Assume that, using **REGMIN**, we just get either the word of length n of $(L(G))_{min}$ or an error message that there exists no such a word in $(L(G))_{min}$. Having the error message, we are ready. Otherwise, construct a regular grammar G' with $L(G') = (L(G) \setminus (L(G))_{min}$, consider G' instead of G and use the above procedure again.

In more details, we consider the following algorithm.

Algorithm reg-enumerate
Input: A regular grammar $G = (V, \Sigma, P, S)$, a total order $\prec$ on Σ and a positive integer n.
Output:

- $L_{G_j} = \{u_{j,1}, v_{j,1}, w_{j,1} \ldots, u_{j,n(G_j)}, v_{j,n(G_j)}, w_{j,n(G_j)}\}$, k_j, ℓ_j, $j = 1, \ldots, m$ (having $m = card\{p \in L(G) \mid |p| = n\}$, $L_j = \bigcup_{i=1}^{n(G_j)} u_{j,i} v_{j,i}^* w_{j,i}$, $j = 1, \ldots, m$ with $L_0 = L(G)$, $L_1 = L_{min}$, $L_k = L_{k-2} \setminus L_{k-1}$, $k = 2, \ldots, m$, such that $1 \leq k_j \leq n(G_j)$, $|u_{j,k_j} v_{j,k_j}^{\ell_j} w_{j,k_j}| = n$, $j = 1, \ldots, m$, $u_{1,k_1} v_{1,k_1}^{\ell_1} w_{1,k_1} \prec u_{2,k_2} v_{2,k_2}^{\ell_2} w_{2,k_2} \prec \ldots \prec u_{m,k_m} v_{m,k_m}^{\ell_m} w_{m,k_m}$) if $L(G)$ has a word of length n;
- an error message otherwise.

Method:
$P =' no'$;
while REGMIN has no error message **do**
$P =' yes'$;
Apply the algorithm **REGMIN**;
Construct a regular grammar G' having $L(G') = (L(G) \setminus (L(G))_{min}$; $G \leftarrow G'$;
od
if $P =' no'$ **then Output:** an error message;
End of algorithm reg-enumerate

4 Enumeration of context-free languages

In [8] it is conjectured that there exists no efficient enumeration algorithm for the lexicographic enumeration of context-free languages. We can provide an algorithm for enumeration of context-free languages, running in polynomial time and space. First we consider the following modified version of CYK algorithm to decide whether a word is a prefix of a word of given length of the language.

Algorithm MCYK
Input: A context-free grammar $G = (V, \Sigma, P, S)$ given in Chomsky normal form, a word $u = b_1 \dots b_m \in \Sigma^+$ $(b_1, \dots, b_m \in \Sigma)$, and a positive integer n.
Output: a variable P having the value

- $P =$*'yes'*, if u is a prefix of an n-length word in $L(G)$;
- $P =$*'no'*, otherwise.

Method:
if $m > n$ then $P =$'no' **else do**
for $i \leftarrow 1 \dots n$ **do**
  if $i \leq m$
    then $V_{i,1} \leftarrow \{A \mid A \to b_i$ is a production $\}$
    else $V_{i,1} \leftarrow \{A \mid \exists a \in \Sigma$ such that $A \to a$ is a production $\}$
od
for $j \leftarrow 2 \dots n$ **do**
  for $i \leftarrow 1 \dots n - j + 1$ **do**
  $V_{i,j} \leftarrow \emptyset$;
    for $k \leftarrow 1 \dots j - 1$ **do**
    $V_{i,j} \leftarrow V_{i,j} \cup \{A \mid A \to BC$ is a production, B is in $V_{i,k}$ and C is in $V_{i+k,j-k}\}$
    od
  od
  od
if $S \in V_{1,n}$ **then** $P =$'yes'
else $P =$'no';
  od
Output: P;
End of algorithm MCYK

Now we construct an algorithm to enumerate the words of length n in context-free languages. We consider the following idea for such an algorithm. Assume we just output $u = a_1 a_2 \cdots a_n$ and are looking for the next word in lexicographical order of length n in $L(G)$. This word, when it exists, has the form

$$v = a_1 a_2 \cdots a_i b_{i+1} b_{i+2} \cdots b_n,$$

for some $0 \leq i \leq n-1, a_{i+1} \prec b_{i+1}$. Clearly, when v exists, we have

$$i = \max\{j \mid 0 \leq j \leq n-1, a_1 a_2 \cdots a_j \text{ is the prefix of a word } w \in L(G) \text{ such that } |w| = n \text{ and the (j+1)}st \text{ letter of } w \text{ is bigger than } a_{j+1}\},$$

$$b_{i+1} = \min\{b \in \Sigma \mid a_{i+1} \prec b \text{ and } a_1 a_2 \cdots a_i b \in Pref(L(G) \cap \Sigma^n)\},$$

and, for any $2 \leq j \leq n - i$,

$$b_{i+j} = \min\{b \in \Sigma \mid a_1 a_2 \cdots a_i b_{i+1} b_{i+2} \cdots b_{i+j-1} b \in Pref(L(G) \cap \Sigma^n)\}.$$

Now, the algorithm should be clear; find first i and b_{i+1} and then, in order, b_{i+2}, b_{i+3}, ..., b_n. Notice that v exists iff i exists and, when both do, we look for each b_j knowing that there must be one.

Algorithm cf-enumerate
Input: A context-free grammar $G = (V, \Sigma, P, S)$, a total order $\prec$ on Σ and a positive integer n.
Output:

- The words of length n in $L(G)$ in lexicographical order if $L(G)$ has a word of length n;
- an error message otherwise.

Method:
Determine the minimal word $p_{min(G,n)}$ of length n in $L(G)$, if such a word exists (apply either methods in [8] having $O(n^2)$ time complexity or the algorithm **CFMIN**);
if there exists no word of length n in $L(G)$ **then** P ='no';
Output: an error message;
else do $a_1 \ldots a_n \leftarrow p_{min(G,n)}$; P ='yes' **od**
while P ='yes' **do**
  Output: $a_1 \ldots a_n$;
  P ='no'; $m \leftarrow n + 1$;
  while P ='no' **and** $m > 1$ **do**
  $m \leftarrow m - 1$; $b \leftarrow a_m$;
   while P ='no' **and** $next(b) \in \Sigma$ **do**
   $b \leftarrow next(b)$; $b_m \leftarrow b$;
if $m > 1$ **then** apply **MCYK** for the inputs $a_1 \ldots a_{m-1} b_m$ and n;
else apply **MCYK** for the inputs b_1 and n;
   od
  od
  if P ='yes' **then do**
if $m > 1$ **then** $b_1 \ldots b_{m-1} \leftarrow a_1 \ldots a_{m-1}$;
   while $m < n$ **do**
  $m \leftarrow m + 1$
  $b \leftarrow first(\Sigma)$; $b_m \leftarrow b$;
  Apply **MCYK** for the inputs $b_1 \ldots b_m$ and n;
   while P ='no' **and** $next(b) \in \Sigma$ **do**
  $b \leftarrow next(b)$; $b_m \leftarrow b$;
  Apply **MCYK** for the inputs $b_1 \ldots b_m$ and n;

```
        od
      od
    a_1 ... a_n <- b_1 ... b_n;
    od
od
```

End of algorithm cf-enumerate

Acknowledgement. The author expresses his gratitude to Professor Arto Salomaa for his kind invitation and hospitality.

The author thanks the anonymous referee for pointing out the errors in a preliminary version which have improved the clarity and the quality of the paper.

References

[1] M. Andraşiu, G. Păun, J. Dassow, A. Salomaa, Language-theoretic problems arising from Richelieu cryptosystems, *Theoret. Comput. Sci.,* **116** (1993) 339-357.

[2] J. Berstel, L. Boasson, The set of minimal words of a context-free language is context-free, *J. Comput. Syst. Sci.,* **55** (1997) 477-488.

[3] J. Dassow, G. Păun, A. Salomaa, On thinness and slenderness of L languages, *Bull.* EATCS **49** (1993), 152-158.

[4] S. Eilenberg, Automata, languages and machines, Vol A, *Academic Press,* New York, 1974.

[5] K. Hermann, G. Walter, A simple proof of Valiant's lemma, *R.A.I.R.O. Inform. Theor. Appl.* **20** (1986) 183-190.

[6] L. Ilie, On a conjecture about slender context-free languages, *Theoret. Comput. Sci.,* **132** (1994) 427-434.

[7] L. Ilie, On lengths of words in context-free languages, *Theoret. Comput. Sci.,* to appear.

[8] E. Mäkinen, On lexicographic enumeration of regular and context-free languages, *Acta Cybernet.,* **13** (1997) 55-61.

[9] G. Păun, A. Salomaa, Thin and slender languages, *Discrete Appl. Math.* **61** (1995) 257-270.

[10] D. Raz, Length considerations in context-free languages, *Theoret. Comput. Sci.* **183** (1997) 21-32.

[11] L. Valiant, General context-free recognition in less than cubic time, *J. Comput. Syst. Sci.* **10** (1975) 308-315.

Received June, 1998