Acta Cybernetica **13** (1997) 55-61.

On lexicographic enumeration of regular and context-free languages*

Erkki Mäkinen†

Abstract

We show that it is possible to efficiently enumerate the words of a regular language in lexicographic order. The time needed for generating the next word is $O(n)$ when enumerating words of length n. We also define a class of context-free languages for which efficient enumeration is possible.

1 Introduction

In [4] we considered the ranking and unranking algorithms for left Szilard languages of context-free grammars. These algorithms imply similar algorithms for context-free languages generated by arbitrary unambiguous context-free grammars. The present paper concerns a somewhat similar but more difficult problem of enumerating regular and context-free languages in lexicographic order. The widely studied problem of coding binary trees [3, 7] can be considered as a subproblem of our present problem. For example, in Zaks' coding method [7] we label the nodes and the leaves of a binary tree by 1 and 0, respectively. By traversing the tree in preorder we obtain a code word consisting of n (the number of nodes) 1's and $n+1$ 0's. The same set of words is obtained by considering the context-free language generated by productions $S \to 1SS$ and $S \to 0$. However, in the general case there are several nonterminals in the grammar in question. This means that the nodes in the corresponding derivation trees have different labels, and the problem of enumerating the "feasible codewords", i.e. the words in the language generated, is much more difficult.

2 Preliminaries

If not otherwise stated we follow the notations and definitions of [1]. Context-free grammars are denoted by $G = (V, \Sigma, P, S)$, where Σ is the set of terminals and V is the union of Σ and the set N of nonterminals.

*This work was supported by the Academy of Finland

†Department of Computer Science, University of Tampere, P.O. Box 607, FIN-33101 Tampere, Finland

If A is a nonterminal in a context-free grammar $G = (V, \Sigma, P, S)$, then $L(G, A)$ stands for the language derivable from A according to the productions of G. The length of a string β is denoted by $len(\beta)$.

For the sake of notational simplicity, we assume that context-free grammars are in Chomsky normal form (CNF), so that all productions are of the form $A \to BC$ or $A \to a$, where A, B, and C are nonterminals, and a is a terminal. The productions having A in their left hand side are called *A-productions.* We say that a production of the form $A \to a$ is *terminating*; the other productions are *continuing.* In a *regular grammar* [1] continuing productions have the form $A \to aB$.

When considering a lexicographic order in $L(G)$ generated by a context-free grammar $G = (V, \Sigma, P, S)$, we suppose that there is a total order $\prec_G$ defined in Σ which imposes the lexicographic order of the words in $L(G)$.

Throughout the paper, we use the unit-cost model for time and space. Hence, we suppose that normal arithmetic operations for arbitrary integers are possible in constant time and an arbitrary integer can be stored in one memory cell. All time and space bounds are given as functions of the length of words. The numbers of productions and nonterminals are always considered as constants.

3 Finding minimal words of given length

We first consider the problem of finding the lexicographically minimal words of different length in a given language. This problem is somewhat related to a very recently solved problem concerning the closure of context-free languages under min-operation. Namely, given a context-free language L, the language L_{min} is obtained by taking from all words of L of the same length only the first in lexicographic order [5]. Raz [6] has recently shown that L_{min} is context-free for an arbitrary context-free langauge L. Given a context-free grammar $G = (V, \Sigma, P, S)$, a total order $\prec_G$ in Σ, and a natural number n, our task in this section is to determine w such that $len(w) = n$ and $w \in L_{min}$.

In order to efficiently perform this task, we store in $A_{min}[i]$, for each nonterminal A and for each length $i = 1, \ldots, n-1$, the lexicographically minimal terminal string of length n obtainable from A according to the productions of G. Hence, each table entry $A_{min}[i]$ belongs to $L(G, A)_{min}$.

The following algorithm tabulates the A_{min} values for each nonterminal of the grammar in question. To simplify the notations, we suppose that Ω is not in Σ and we define $a \prec_G \Omega$ for all a in Σ. Ω will be used as a null value for undefined table entries. Moreover, we use the notation $conc(u, v)$ to stand for the normal concatenation of strings u and v, i.e. $conc(u, v) = uv$.

Algorithm 3.1 (Min)
Input: *A context-free grammar $G = (V, \Sigma, P, S)$, a total order $\prec_G$ in Σ, and a positive integer n.*
Output: *Table $A_{min}[1..n]$, for each nonterminal $A \in V \setminus \Sigma$; $min = S_{min}[n]$ is the minimal word of length n.*

Method:

for *each nonterminal* A **do**
  if *there is no terminating* A*-productions*
    then $A_{min}[1] \leftarrow \Omega$
    else $A_{min}[1] \leftarrow a$ *where* $a \prec_G b$ *holds for all other terminals* b *appearing in the right hand sides of terminating* A*-production;*
  for $i \leftarrow 2 \ldots n$ **do**
    for *each nonterminal* A **do**
    $min \leftarrow \Omega;$
    for *each continuing* A*-production* $A \to BC$ **do**
      for $j \leftarrow 1 \ldots i-1$ **do**
        if $B_{min}[j] \neq \Omega$ **and** $C_{min}[i-j] \neq \Omega$
        then
          if $conc(B_{min}[j], C_{min}[i-j]) \prec_G min$
          then $min \leftarrow conc(B_{min}[j], C_{min}[i-j])$
        od
      od
      $A_{min}[i] \leftarrow min;$
    od

End of Algorithm

As already mentioned, we consider the size of a grammar (including the numbers of terminals, nonterminals and productions) as a constant. Noticing this assumption it is clear that algorithm Min runs in time $O(n^2)$.

We also consider the total order $\prec_G^{-1}$ defined by letting $a \prec_G^{-1} b$ if and only if $b \prec_G a$. The minimal word in lexicographic order in $L(G)$ according to $\prec_G^{-1}$ is the maximal one according to $\prec_G$. This word is denoted by max (cf. min in Algorithm 3.1).

Theorem 3.1 *Let G be a context-free grammar. The words min and max of length n can be found in time $O(n^2)$ and in space $O(n)$.*

Theorem 3.1 can be sharpened if the input grammar is regular. Also the form of the algorithm changes a bit. Next, we rewrite the whole algorithm for the regular case.

Algorithm 3.2 (Reg-Min)

Input: *A regular grammar* $G = (V, \Sigma, P, S)$, *a total order* $\prec_G$ *in* Σ, *and a positive integer* n.
Output: *Table* $A_{min}[1..n]$, *for each nonterminal* $A \in V \setminus \Sigma$; $min = S_{min}[n]$ *is the minimal word of length* n.
Method:

for *each nonterminal* A **do**
  if *there is no terminating* A*-productions*
    then $A_{min}[1] \leftarrow \Omega$
    else $A_{min}[1] \leftarrow a$ *where* $a \prec_G b$ *holds for all other terminals* b *appearing in the right hand sides of terminating* A*-production;*
  for $i \leftarrow 2 \ldots n$ **do**
    for *each nonterminal* A **do**
    $min \leftarrow \Omega$;
    for *each continuing* A*-production* $A \rightarrow aB$ **do**
      if $B_{min}[i-1] \neq \Omega$
      then
        if $conc(a, B_{min}[i-1]) \prec_G min$
        then $min \leftarrow conc(a, B_{min}[i-1])$
        od
      od
      $A_{min}[i] \leftarrow min$;
    od

End of Algorithm

In Algorithm Reg_Min only a constant number of operations is needed for determining each table entry. Hence, we have the following theorem.

Theorem 3.2 *Let* G *be a regular grammar. The words* min *and* max *of length* n *can be found in* $O(n)$ *time and space.*

4 Enumeration of regular languages

So far, we have been able to find the minimal and maximal words in $L(G)$ of given length in lexicographic order. The algorithm enumerating the words in $L(G)$ of given length can now be given as follows using the words min and max:

Algorithm 4.1 (Enumerate)

Input: *A context-free grammar* $G = (V, \Sigma, P, S)$, *a total order* $\prec_G$ *in* Σ, *and a positive integer* n.
Output: *The words on length* n *in* $L(G)$ *in lexicographic order.*
Method:

present_word $\leftarrow$ *min*;
while *present_word* $\neq$ *max* **do**
find the next word in lexicographic order **od**

End of Algorithm

Obviously, our problem is to specify the step "find the next word in lexicographic order". We first consider the problem in the case of regular languages.

Suppose G is a regular grammar and $a_1a_2 \ldots a_n$ is a word in $L(G)$. We know that there is a deterministic finite automaton accepting $L(G)$ [1]. In terms of grammars this means that there is a regular grammar H such that $L(H) = L(G)$ and, for each nonterminal A, the terminals appearing in the right hand sides of A-productions are all different. Hence, without loss of generality, we can suppose that G has this property. It follows that we can conclude the sequence of nonterminals $S = A_1, A_2, \ldots, A_n$ needed in deriving the word $a_1a_2 \ldots a_n$ from the start symbol S, and further, we can conclude the sequence of productions applied.

We start from the end of $a_1a_2 \ldots a_n$ and look for a position in which we can replace the symbol a_i with a symbol b such that $a_i \prec_G b$.

The last symbol a_n is the only one in $a_1a_2 \ldots a_n$ produced by a terminating production. We first check whether or not there is a symbol b such that $A_n \to b$ is another terminating production and $a \prec_G b$. Provided that b is the first (according to $\prec_G$) such symbol we have found out that $a_1a_2 \ldots a_{n-1}b$ is the successor of $a_1a_2 \ldots a_n$. Otherwise (such b does not exist), we have to proceed further to the left.

Suppose now that a_i, $1 \leq i \leq n-1$, is the first symbol that can be replaced. This means that we have a continuing production $A_i \to bB$ such that $a_i \prec_G b$ (and b is before other such terminals according to $\prec_G$). If now $B_{min}[n-i]$ is defined, we can write the successor of $a_1a_2 \ldots a_n$ as

$$conc(a_1a_2 \ldots a_{i-1}b, B_{min}[n-i]).$$

Hence, when a symbol is changed then all positions in its right get the lowest possible value. If the B_{min} value is undefined for all possible B's appearing in the right hand sides of A_i-productions, we again have to proceed to the left.

If $a_1a_2 \ldots a_n \neq max$ then at least one of the symbols in $a_1a_2 \ldots a_n$ must be changeable. Since the number of productions is considered to be a constant, linear time is sufficient for finding the successor of a given word $a_1a_2 \ldots a_n$. Hence, we have the following theorem.

Theorem 4.1 *Given a regular grammar G, there is an algorithm for enumerating the words in $L(G)$ in lexicographic order such that the time needed for generating the next word is $O(n)$.*

Notice that the time bound of Theorem 4.1 holds also for the first word of the enumeration, i.e. for the minimal word in lexicographic order. This follows from Theorem 3.2.

5 Enumeration of context-free languages

In the previous section we were able to show that regular languages have an efficient enumeration algorithm. Unfortunately, it seems that the same does not hold for context-free langauges.

For the sake of simplicity, we suppose that context-free languages considered in the rest of the paper are generated by unambiguous context-free grammars. Suppose now that we apply the same approach as we used for regular languages. Hence, a word $a_1 a_2 \ldots a_n$ in $L(G)$ is given, and we first find out the sequence of productions used in the leftmost derivation producing the word. A unique derivation is always found because we suppose that G is unambiguous.

Let a_i be the symbol to be replaced with a symbol b having the property $a_i \prec_G b$. We have a leftmost derivation

$$S \Rightarrow \ldots \Rightarrow a_1 \ldots a_{i-1}\alpha \Rightarrow a_1 \ldots a_{i-1}a_i\beta$$

where β is a string of nonterminals such that $1 \leq len(\beta) \leq n-i$. We should now be able to efficiently find the lexicographically minimal word of length $n-i$ derivable from β. As in Algorithm 3.1 we have to check all possible combinations of the A_{min} table entries, for each nonterminal instance A appearing in β. In the general case, there seems to be no efficient solution for this problem.

On the other hand, an inefficient method can be implemented even without the preprocessing phase described in section 3: simply enumerate all the words in Σ^* and delete those not in $L(G)$.

We end this section by defining a subclass of context-free grammars which allow efficient enumeration of words in lexicographic order.

We say that a context-free grammar is *strongly prefix-free* if $L(G, A)$ is prefix-free for each nonterminal A. More formally, G is stronly prefix-free if derivations $A \Rightarrow^+ u$ and $A \Rightarrow^+ v$, where u and v are terminal strings, always imply that both $u = vw$ and $v = uw$ are impossible for all non-empty strings w. The class grammars generating left Szilard languages of context-free grammars [2] is an example of strongly prefix-free grammars.

Moreover, we say that a context-free grammar G is *length complete* if the following condition is fulfilled for each nonterminal A:

- if $w \in L(G, A)$, $len(w) = n$, then, for each i, $i = 1 \ldots n-1$, $L(G, A)$ contains a word of length i.

If G is stronly prefix-free then it is sufficient to maintain the A_{min} table values in lexicographic order and to consider only the minimal values from each table. This follows from the fact that in strongly prefix-free grammars the set of A_{min} values is always prefix-free. A_{min} values can be easily maintained in lexicographic order by using radix sort. Moreover, if G is length complete, then there is no need for backtracking because of lacking words of certain length.

The preprocessing phase (filling in the A_{min} tables) is now (asymptotically) as simple as with regular languages. Similarly, the next word can always be found (asymptotically) as efficient as in the case of regular languages. Hence, we have the following theorem.

Theorem 5.1 *Given a stronly prefix-free, length complete context-free grammar G, there is an algorithm for enumerating the words in $L(G)$ in lexicographic order such that the time needed for generating the next word is $O(n)$.*

References

[1] M.A. Harrison, *Introduction to Formal Language Theory* (Addison-Wesley, 1978).

[2] E. Mäkinen, On context-free derivations. *Acta Universitatis Tamperensis* **197** (1985).

[3] E. Mäkinen, A survey on binary tree codings. *Comput. J.* **34** (1991) 438–443.

[4] E. Mäkinen, Ranking and unranking left Szilard languages. Dept. of Computer Science, University of Tampere. Report **A-1997-2**, January 1997.

[5] G. Păun and A. Salomaa, Closure properties of slender languages. *Theoret. Comput. Sci.* **120** (1993), 293–301.

[6] D. Raz, Context-free languages are closed under *min* operation. Manuscript, submitted for publication.

[7] S. Zaks, Lexicographic generation of ordered trees. *Theor. Comput. Sci.* **10** (1980) 63–82.

Received March, 1997