

Efficient Computation of Throughput Values of Context-Free Languages

Didier Caucal^a Jurek Czyzowicz^b Wojciech Fraczak^b
Wojciech Rytter^c

^a*IGM-CNRS, Marne-la-Vallée, France*

^b*Dépt d'informatique, Université du Québec en Outaouais, Gatineau PQ, Canada*

^c*Inst. of Informatics, Warsaw University, Warsaw, Poland*

Abstract

We give the first deterministic polynomial time algorithm that computes the *throughput* value of a given context-free language L . The language is given by a grammar G of size n , together with a weight function assigning a positive weight to each symbol. The *weight* of a word $w \in L$ is defined as the sum of weights of its symbols (with multiplicities), and the *mean weight* is the weight of w divided by length of w . The *throughput* of L , denoted by $\text{throughput}(L)$, is the smallest real number t , such that the mean value of each word of L is not smaller than t . Our approach, to compute $\text{throughput}(L)$, consists of two phases. In the first one we convert the input grammar G to a grammar G' , generating a finite language L' , such that $\text{throughput}(L) = \text{throughput}(L')$. In the next phase we find a word of the smallest mean weight in a finite language L' . The size of G' is polynomially related to the size of G .

The problem is of practical importance in system-performance analysis, especially in the domain of network packet processing, where one of the important parameters is the “*guaranteed throughput*” of a system for on-line network packet processing.

Key words: context-free grammar, push-down automaton, minimal mean weight, throughput

1 Introduction

In this paper we propose the first polynomial time algorithm computing the throughput of context-free languages. An algorithm computing the approximation of the value of throughput of a context-free grammar was proposed in

[1]. The algorithms are applicable in the context of system-performance analysis, particularly in the context of network packet processing. As described in [2], the essential criterion for the evaluation of a system is the measure of its worst case speed of processing data, i.e., its *throughput*. More precisely, this criterion is the lower bound (the infimum) of the greatest ratio of the length of processed input packet to its processing time, taken over all possible input packets. In some cases, a simple system allows a representation by a regular language (a finite automaton) with each alphabet symbol representing a constant time *task* consuming a constant amount of packet data. As noted in [3], in such a case any standard algorithm for minimum mean cycle calculation, e.g., [4], would do. In practice, however, especially when more complex systems are analyzed and a better accuracy is required, context-free grammars have to be used to adequately describe the behavior of the systems. As a consequence, the practical worst case throughput computation of a system is equivalent to the context-free grammar throughput computation, which is the subject of this paper.

2 Notation

Let Σ be a finite alphabet with a weight function $\rho : \Sigma \mapsto \mathbb{N}$, where \mathbb{N} is the set of positive integers. A word over Σ is any finite sequence of letters. The set of all words is denoted by Σ^* , and the set of all non-empty words by Σ^+ . The length of a word w is denoted by $|w|$ and the empty word is denoted by ε .

The weight function defined over Σ extends onto non-empty words in the following way:

$$\rho(a_1 a_2 \dots a_n) \stackrel{\text{def}}{=} \rho(a_1) + \rho(a_2) + \dots + \rho(a_n).$$

The *mean weight* of a non-empty word w is defined as

$$\bar{\rho}(w) \stackrel{\text{def}}{=} \frac{\rho(w)}{|w|}.$$

Given a non-empty language $L \subseteq \Sigma^+$, we define *throughput* of L as the infimum of the mean weight of all words of L :

$$\text{throughput}(L) \stackrel{\text{def}}{=} \inf \{ \bar{\rho}(w) \mid w \in L \}.$$

In other words, *throughput* of L is a real value $t \in \mathbb{R}$ such that:

- (1) $\forall w \in L, \bar{\rho}(w) \geq t$, and
- (2) $\forall \epsilon > 0, \exists w \in L, \bar{\rho}(w) < t + \epsilon$.

Note that the mean weight of a word w is independent of the order of symbols used in w . Hence, the alphabet commutativity may be used in our approach. Parikh [5] showed that the commutative image of every context-free language is the commutative image of some regular language. Consequently, for every context-free language L we can find a regular language R such that $\text{throughput}(L) = \text{throughput}(R)$. However, a direct transformation of a language given by a context-free grammar G to a commutatively equivalent regular expression (or a finite automaton) may yield the result of an exponential size with respect to the size of G . Also, it is worth noting that the Hopkins-Kozen acceleration methods from [6,7], which are built around the Parikh's theorem, do not apply in the context of throughput calculation.

A context-free grammar $G = (\Sigma, N, P, S)$ is composed of a finite set Σ of *terminals*, a finite set N of *nonterminals* disjoint from Σ , a finite set $P \subseteq N \times (N \cup \Sigma)^*$ of *production rules*, and an axiom $S \in N$. The *size of the grammar* is defined as the sum of the number of terminals, nonterminals, and the lengths of right hand sides of production rules.

A context-free grammar $G = (\Sigma, N, P, S)$ defines the set of *syntax trees*, i.e., ordered rooted trees with inner nodes labeled by nonterminals and leaves labeled by $\Sigma \cup \{\varepsilon\}$. Assume, until the end of the paper, that the language L does not contain the empty word and that the grammar does not use empty rules. For every inner node v labeled by $X \in N$, with k children v_1, \dots, v_k there exists a production $X \rightarrow A_1, \dots, A_k \in P$, with $A_i \in \Sigma \cup N$, such that v_i is labeled by A_i , for $i \in [1, k]$.

Every syntax tree T defines a word $w(T)$ over Σ ; it is the concatenation of its leaf labels (read from left to right).

The set of all words over Σ generated by all syntax trees of a grammar $G = (\Sigma, N, P, S)$ with root labeled by $A \in N \cup \Sigma$, is denoted by $L_G(A)$. E.g., if $A = a \in \Sigma$, $L_G(a) = \{a\}$. The language of G is defined as $L_G(S)$ and denoted by $L(G)$.

Every grammar G of size n may be converted in $O(n)$ time to a *2-reduced* grammar G' of size $O(n)$, such that $L(G) = L(G')$. A grammar is *2-reduced* if it is trimmed (there are no useless nonterminals), and each of its production rules has one or two symbols on the right-hand side. Hence, without loss of generality, we may suppose that all the context-free grammars under consideration are 2-reduced grammars.

3 Throughput of a finite language

In the case of an infinite language it is possible that its throughput is not equal to the mean weight of any of its words, e.g., as it is the case for the regular language ab^* , when $\rho(a) > \rho(b)$.

However, in the case of a finite language L there always exists a word in L , whose mean weight equals the throughput of L .

In this section we give an algorithm which, given a grammar generating a finite language L , finds a word w such that $\bar{\rho}(w) = \text{throughput}(L)$.

Let $L \subset \Sigma^+$ be a finite non-empty language with weight function $\rho : \Sigma \mapsto \mathbb{N}$. Given a positive real value t , we define *throughput balance* of L with respect to t , denoted by $tb(L, t)$, as the following real value:

$$tb(L, t) \stackrel{\text{def}}{=} \min \{(\rho(w) - |w|t) \mid w \in L\}$$

Intuitively, $tb(L, t)$ can be seen as a measurement of the “surplus/deficit” of L with respect to a given throughput t ; If $tb(L, t) > 0$ (resp., $tb(L, t) < 0$) then language L has a “surplus” (resp., “deficit”) in achieving throughput t .

Note 1 *The real value $tb(L, t)$, which may be negative, corresponds to the minimal weight of a word in L with respect to the modified weight function $\rho_t : \Sigma \mapsto \mathbb{R}$ defined as $\rho_t(a) \stackrel{\text{def}}{=} \rho(a) - t$.*

Lemma 1 *Let L be a finite language over a weighted alphabet and t be a positive real value. We have:*

$$tb(L, t) \geq 0 \Leftrightarrow \text{throughput}(L) \geq t$$

PROOF. L is finite, hence $\text{throughput}(L) = \min_{w \in L} \bar{\rho}(w)$.

$$\min_{w \in L} \frac{\rho(w)}{|w|} \geq t \Leftrightarrow \min_{w \in L} \left(\frac{\rho(w)}{|w|} - t \right) \geq 0 \Leftrightarrow \min_{w \in L} \frac{\rho(w) - |w|t}{|w|} \geq 0$$

consequently we have that for all $w \in L$, $|w| > 0$:

$$\min_{w \in L} \frac{\rho(w) - |w|t}{|w|} \geq 0 \Leftrightarrow tb(L, t) \geq 0.$$

□

Lemma 2 *Let $G = (\Sigma, N, P, S)$ be a grammar of size n generating a non-empty finite language $L \subset \Sigma^+$ with weight function $\rho : \Sigma \rightarrow \mathbb{N}$. Given a positive real value t , we can decide in $O(n)$ time whether $\text{throughput}(L) \geq t$.*

PROOF. By Note 1 and Lemma 1, deciding whether $\text{throughput}(L) \geq t$ can be done by finding the minimal weight of a word in L with respect to the modified weight function ρ_t .

For all $a \in \Sigma$, we have $tb(L_G(a), t) = \rho_t(a) = \rho(a) - t$.

L is finite, consequently there exists a partial ordering of the nonterminals of G , such that for any $X, Y \in N$ we have $X < Y$ if there exists a syntax tree of some word of L in which X is a descendant of Y . We topologically sort, in $O(n)$ time, the nonterminals of G , which gives such an ordering.

Then, for every $X \in N$, in the increasing order, we compute $tb(L_G(X), t)$.

More precisely, $tb(L_G(X), t)$ is computed as the minimum, over all rules with X on the left side, of the sums of the throughput balances of the right-hand side symbols. The value $tb(L_G(X), t)$ is stored in an array requiring $O(|N|)$ memory space.

Each production rule is taken into consideration once only, consequently the overall cost is linear in the size of the input grammar. \square

The following lemma shows that we can bound the density of the set of mean weights of words of a finite language.

Lemma 3 *Let $L \subset \Sigma^+$ be a finite language, $\rho : \Sigma \mapsto \mathbb{N}$ a weight function, and m the maximum length of a word of L , i.e., $m = \max\{|w| \mid w \in L\}$. The minimum difference between mean weight of two words of L is not smaller than $\frac{1}{m^2}$. I.e., for every $w_1, w_2 \in L$:*

$$\bar{\rho}(w_1) > \bar{\rho}(w_2) \Rightarrow \bar{\rho}(w_1) - \bar{\rho}(w_2) \geq \frac{1}{m^2} .$$

PROOF. We have:

$$\Delta = \frac{\rho(w_1)}{|w_1|} - \frac{\rho(w_2)}{|w_2|} > 0, \quad |w_1||w_2|\Delta = |w_2|\rho(w_1) - |w_1|\rho(w_2) > 0.$$

Since $|w_2|\rho(w_1) - |w_1|\rho(w_2)$ is an integer, $|w_1||w_2|\Delta \geq 1$. i.e., $\Delta \geq \frac{1}{m^2}$. \square

Theorem 4 *Let G be a grammar of size n defining a finite language L with weight function ρ over alphabet Σ such that $\max_{a \in \Sigma} \rho(a) - \min_{a \in \Sigma} \rho(a) =$*

d. There exists an $O(n \log md)$ time algorithm that computes $\text{throughput}(L)$, where m is the maximum length word of L .

PROOF. The throughput of L belongs to the interval $[\min_{a \in \Sigma} \rho(a), \max_{a \in \Sigma} \rho(a)]$. Using Lemma 2, we can perform a binary search in this interval to determine the sub-interval $[r - \frac{1}{m^2}, r]$, for some real value r , which must contain the throughput of L , i.e., $r - \frac{1}{m^2} \leq \text{throughput}(L) \leq r$ or equivalently, by Lemma 1,

$$tb(L, r - \frac{1}{m^2}) \geq 0 \geq tb(L, r).$$

Let w_r be that word from L for which $\rho_r(w_r) = tb(L, r)$. One can show that $\bar{\rho}(w_r)$ and $\text{throughput}(L)$ are both in the interval. Consequently, by Lemma 3, $\text{throughput}(L) = \bar{\rho}(w_r)$.

The binary search reducing an interval of size d to a size not bigger than $\frac{1}{m^2}$ takes $O(\log md)$ iterations and, by Lemma 2, each iteration works in $O(n)$ time. Finding word w_r and then computing its mean weight $\bar{\rho}(w_r)$ can be done in $O(n)$ time. For that, during the last iteration the procedure described in the proof of Lemma 2 has to be slightly extended; for every $X \in N$, we need to store a production rule corresponding to the throughput balance $tb(L_G(X), r)$, i.e., for which the sum of the throughput balance of the right-hand side symbols equals $tb(L_G(X), r)$. The set of the stored production rules defines a one-word grammar corresponding to w_r , from which $\bar{\rho}(w_r)$ can be easily calculated in $O(|N|)$ time.

Thus, the overall time complexity of finding $\text{throughput}(L)$ is $O(n \log md)$. \square

4 Throughput invariant grammar transformation

In this section we show how to convert any context-free grammar $G = (\Sigma, N, P, S)$ into another grammar $G' = (\Sigma, N', P', S')$ that generates a finite language, such that $\text{throughput}(L(G)) = \text{throughput}(L(G'))$. The main idea behind the transformation is the observation that the throughput of $L(G)$ is either equal to the mean weight of some word $w \in L(G)$, whose syntax tree is at most of depth $|N|$, or it is equal to the mean weight of some word $w_1 w_2 \in \Sigma^+$, not necessarily in $L(G)$, such that there exists in G a syntax tree T_X of type as shown in Figure 1, for some $X \in N$.

Let $G = (\Sigma, N, P, S)$ be a 2-reduced grammar of size n . We define the following grammar $\text{Fin}(G) \stackrel{\text{def}}{=} G' = (\Sigma, N', P', S')$ generating a finite language as

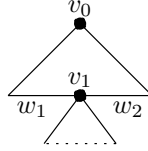


Fig. 1. Syntax tree T_X : nodes v_0 and v_1 are labeled by the same nonterminal X .

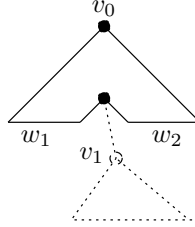


Fig. 2. Every syntax tree in G' (solid lines) corresponds to an initial part of many syntax trees in G (solid and dashed lines). The root v_0 is labeled by $X_{|N|}^X$ in G' and by X in G . Every inner node on the path from v_0 to v_1 is labeled by Y_k^X in G' , for some $Y \in N$ and $k \in \{1, \dots, |N|\}$, and by Y in G . All other inner nodes of solid-line tree are labeled by Y_k in G' , for some $Y \in N$ and $k \in \{1, \dots, |N|\}$, and by Y in G . The subtree rooted in v_1 (dashed lines) is any syntax tree in G with v_1 labeled by X .

follows:

- The set of nonterminals N' is defined as the union $N' = N'_f \cup N'_r \cup \{S'\}$, where:
 - S' is the new axiom symbol,
 - For every $X \in N$ and $k \in \{1, \dots, |N|\}$ there is a nonterminal X_k in N'_f .
Intuitively, for every nonterminal $X \in N$, we create $|N|$ nonterminals $X_1, \dots, X_{|N|}$ in N'_f ; $L_{G'}(X_k)$ will correspond to the finite subset of $L_G(X)$ with syntax trees not higher than k . In particular, if $L_G(X)$ is finite then $L_G(X) = L_{G'}(X_{|N|})$.
 - For every $X, Y \in N$ and $k \in \{1, \dots, |N|\}$, there is X_k^Y in N'_r .
We say that $X \in N$ is *recursive* if there is a syntax tree in G with two nodes both labeled by X and such that one node is a proper ancestor of the other. For every recursive nonterminal $X \in N$ and a nonterminal $Y \in N$, we create $|N|$ nonterminals $X_1^Y, \dots, X_{|N|}^Y$ in N'_r . For every syntax tree in G' with its root labeled by $X_{|N|}^X$ there will exist an infinite number of syntax trees in G as depicted in Figure 2.
- The set of production rules is the union $P' = P_f \cup P_r \cup P_i$.

In order to define those production rules we will introduce the following

partial mappings:

$\psi : \mathbb{N} \times (N \cup \Sigma) \mapsto (N'_f \cup \Sigma)^*$ — is defined as

$$\psi(k, A) \stackrel{\text{def}}{=} \begin{cases} A & \text{if } A \in \Sigma \\ A_{k-1} & \text{if } A \in N, k > 1 \\ \text{undefined} & \text{otherwise} \end{cases}$$

$\phi : \mathbb{N} \times N \times (N \cup \Sigma) \mapsto (N'_r \cup \Sigma)^*$ — is defined as

$$\phi(k, X, A) \stackrel{\text{def}}{=} \begin{cases} \varepsilon & \text{if } A = X \\ A_{k-1}^X & \text{if } A \in N, A \neq X, k > 1 \\ \text{undefined} & \text{otherwise} \end{cases}$$

- For every $X \rightarrow A_1 \dots A_j \in P$ and $k \in \{1, \dots, |N|\}$ such that $\psi(k, A_i)$ is defined for all $i \in \{1, \dots, j\}$, there is a rule

$$X_k \rightarrow \psi(k, A_1) \dots \psi(k, A_j)$$

in P_f .

- For every $X \rightarrow A_1 \dots A_j \in P$, $Y \in N$, $k \in \{1, \dots, |N|\}$, and $l \in \{1, \dots, j\}$, such that $\phi(k, Y, A_l)$ and $\psi(k, A_i)$ are defined for all $i \in \{1, \dots, l-1, l+1, \dots, j\}$, there is a rule

$$X_k^Y \rightarrow \psi(k, A_1) \dots \psi(k, A_{l-1}) \phi(k, Y, A_l) \psi(k, A_{l+1}) \dots \psi(k, A_j)$$

in P_r .

- $P_i = \{S' \rightarrow X_{|N|}^X \mid X \in N\} \cup \{S' \rightarrow S_{|N|}\}$

The new grammar G' generates a finite number of syntax trees not higher than $|N| + 1$. G' is of size $O(n^3)$, where n is the size of G .

Example 5 Consider the following context-free grammar $G = (\Sigma, N, P, S)$ where

- $\Sigma = \{a, b, c\}$,
- $N = \{X, Y\}$,
- $P = \{X \rightarrow b, X \rightarrow aY, Y \rightarrow Xc\}$,
- $S = X$.

The corresponding finite language grammar $\text{Fin}(G) = G' = (\Sigma, N', P', S')$ is:

- $N' = N'_f \cup N'_r \cup \{S'\}$, where:
 - $N'_f = \{X_1, X_2, Y_1, Y_2\}$
 - $N'_r = \{X_1^X, X_2^X, Y_1^X, Y_2^X, X_1^Y, X_2^Y, Y_1^Y, Y_2^Y\}$

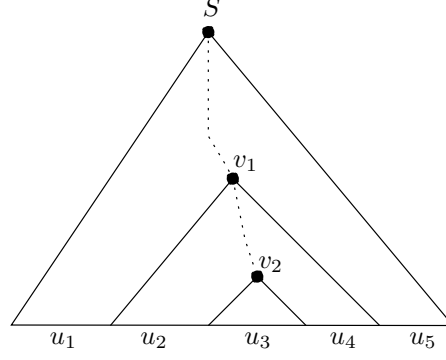


Fig. 3. A syntax tree T with $\langle T \rangle > 0$. Nodes v_1 and v_2 are labeled by the same nonterminal.

- $P' = P_f \cup P_r \cup P_i$ where:

$$P_f = \{X_2 \rightarrow b, X_1 \rightarrow b, X_2 \rightarrow aY_1, Y_2 \rightarrow X_1c\}$$

$$P_r = \{X_2^X \rightarrow aY_1^X, Y_2^X \rightarrow c, Y_1^X \rightarrow c, X_2^Y \rightarrow a, X_1^Y \rightarrow a, Y_2^Y \rightarrow X_1^Y c\}$$

$$P_i = \{S' \rightarrow X_2, S' \rightarrow X_2^X, S' \rightarrow Y_2^Y\}$$

After trimming, the grammar has the following set of productions:

$$\begin{aligned} X_2 &\rightarrow b, X_2^X \rightarrow aY_1^X, Y_1^X \rightarrow c, X_1^Y \rightarrow a, Y_2^Y \rightarrow X_1^Y c, \\ S' &\rightarrow X_2, S' \rightarrow X_2^X, S' \rightarrow Y_2^Y. \end{aligned}$$

with axiom S' . The language of the grammar is $\{b, ac\}$.

In order to prove that $\text{throughput}(L(G)) = \text{throughput}(L(G'))$ we need some auxiliary results (the proof is omitted).

Lemma 6 For all $w, u \in \Sigma^+$ we have:

$$\bar{\rho}(w) \leq \bar{\rho}(u) \Rightarrow \bar{\rho}(w) \leq \bar{\rho}(wu) = \bar{\rho}(uw) \leq \bar{\rho}(u).$$

Let T be a syntax tree of G . By $\langle T \rangle$ we denote the number of different pairs of nodes (v_1, v_2) of T such that v_1 and v_2 both carry the same label, and v_1 is a proper ancestor of v_2 .

Lemma 7 Let $G = (\Sigma, N, P, S)$ and $G' = (\Sigma, N', P', S')$ be context-free grammars such that $G' = \text{Fin}(G)$. For every syntax tree T of G either:

- (1) there exists a word $w' \in L(G')$ such that $\bar{\rho}(w') \leq \bar{\rho}(w(T))$; or
- (2) there exists a syntax tree T_0 of G such that $\langle T_0 \rangle < \langle T \rangle$ and $\bar{\rho}(w(T_0)) \leq \bar{\rho}(w(T))$.

PROOF. If $\langle T \rangle = 0$ then by construction of G' , $w(T) \in L_{G'}(S_{|N|}) \subseteq L(G')$ and the first statement of the lemma holds.

Otherwise, consider Figure 3, where $w(T) = u_1u_2u_3u_4u_5$. Let (v_1, v_2) denote a pair of occurrences of the same nonterminal $X \in N$ in the syntax tree T , such that the level of v_1 is minimal, i.e., there is no pair of nodes (p_1, p_2) in the syntax tree T and that p_1 is a proper descendant of v_1 and p_1, p_2 have the same labels. Therefore, in the syntax tree T the distance between v_1 and v_2 , and between v_1 and all the leaves of v_1 which are not leaves of v_2 (i.e., u_2, u_4) is at most $|N|$.

Thus, u_2u_4 is a word in $L_{G'}(X_{|N|}^X)$, i.e., in $L_{G'}(S')$, and the tree T_0 obtained from T by replacing sub-tree v_1 by v_2 , is the syntax tree of G such that $\langle T_0 \rangle < \langle T \rangle$ and $w(T_0) = u_1u_3u_5$. By Lemma 6, either $\bar{\rho}(u_2u_4) \leq \bar{\rho}(w(T))$ or $\bar{\rho}(u_1u_3u_5) \leq \bar{\rho}(w(T))$. \square

Lemma 8 *For each word $w \in L(G)$, there exists a word $w' \in L(G')$, such that $\bar{\rho}(w') \leq \bar{\rho}(w)$.*

PROOF. By induction on $\langle T \rangle$ for a syntax tree for w in G using Lemma 7. \square

Lemma 9 *For any $\epsilon > 0$ there exists a word $w \in L(G)$, such that*

$$\bar{\rho}(w) < \text{throughput}(L(G')) + \epsilon .$$

PROOF. G' generates a finite language, consequently there exists a non-empty word $w_0 \in L(G')$, such that $\bar{\rho}(w_0) = \text{throughput}(L(G'))$.

If w_0 is generated by a syntax tree with its root labeled by a nonterminal $X_{|N|} \in N'_f$, by construction of G' , we have $w_0 \in L(G)$, and the statement of the lemma is obviously true.

Otherwise, w_0 is generated by a syntax tree T' of G' with its root labeled by $X_{|N|}^X \in N'_r$, as depicted in Figure 2 (solid lines), with $w_0 = w_1w_2$.

For every $k > 0$ there exists a syntax tree T_k of G , as depicted in Figure 4, which generates $u_1w_1^k u w_2^k u_2$ for some $u_1, u_2, u \in \Sigma^*$.

For any $\epsilon > 0$ and any $y \in \Sigma^*$, there exists a sufficiently large k such that

$$\bar{\rho}(w_0^k y) < \bar{\rho}(w_0) + \epsilon$$

In particular, for $y = u_1 u u_2$, we have

$$\bar{\rho}(w(T_k)) = \bar{\rho}(u_1 w_1^k u w_2^k u_2) = \bar{\rho}(w_0^k u_1 u u_2) < \bar{\rho}(w_0) + \epsilon,$$

which proves the statement of the lemma. \square

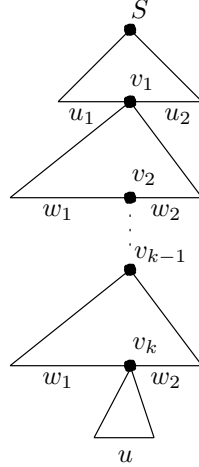


Fig. 4. A syntax tree T_k of $G = (\Sigma, N, P, S)$ generating word $u_1 w_1^k u w_2^k u_2$. Nodes v_1, \dots, v_k are all labeled by the same $X \in N$.

Lemmas 8 and 9 imply directly that our transformation of a context-free grammar does not change the throughput of the language. This can be formulated formally as follows:

Lemma 10 *For every context-free grammar G , we have:*

$$\text{throughput}(L(G)) = \text{throughput}(L(\text{Fin}(G))) .$$

5 Polynomial-time algorithm for throughput computation of a context-free language

Results from two previous sections, namely Theorem 4 and Lemma 10 lead us to the following procedure calculating throughput of a language given by a context-free grammar G .

Algorithm Throughput-Calculation;

Phase 1: Compute grammar $G' = \text{Fin}(G)$ as described
in Section 4;

Phase 2: Find the throughput of $L(G')$ and report
it as the throughput of $L(G)$.

Theorem 11 *Let $G = (\Sigma, N, P, S)$ be a context-free grammar of size n , and $\rho : \Sigma \mapsto \mathbb{N}$ a weigh function such that $\max_{a \in \Sigma} \rho(a) - \min_{a \in \Sigma} \rho(a) = d$. There exists an $O(n^4 + n^3 \log d)$ time algorithm finding throughput($L(G)$).*

PROOF. We have already proved that the two step procedure for calculating throughput of $L(G)$ gives the correct result.

By construction, transformation $Fin(G)$ yields a context free grammar G' of size $O(n^3)$ with all syntactic trees no higher than n . Thus, the maximum length m of a word from $L(G')$ is in $O(2^n)$.

Finally, by Theorem 4, finding the throughput of $L(G')$ takes $O((n^3) \log md)$ time, i.e., $O(n^4 + n^3 \log d)$ since m is in $O(2^n)$. \square

6 Conclusions

We presented the first polynomial-time algorithm computing the throughput of context-free languages. The only previously known solution to this problem was an approximate approach presented in [1]. Our solution may be viewed as a generalization of the technique of Karp [4], working for finite digraphs, to the case of the class of graphs generated by context-free grammars. However, the approach presented here is different from that given by Karp.

Unfortunately the complexity of our approach is substantially higher. An open problem is then to improve the proposed time complexity, possibly by using a completely different approach. In particular, one can try to exploit explicitly the commutative property of the given grammar, i.e., the fact that permuting the symbols of a word or permuting the symbols on the right-hand side of production rules does not affect the throughput of the generated language. We implicitly used (to some extent) the commutation property in our paper while we constructed the transformation Fin of a given grammar to one generating a finite language. The resulting language, though finite, could be of doubly exponential size. Fortunately it was possible to overcome the doubly exponential barrier.

More explicit, direct application of the commutation property may lead to better algorithmic bounds.

Our interest in this problem was directly fuelled by its application in system-performance analysis and, more precisely, in the performance measurement of network packet processing engines. We believe that other applications, in particular in string processing or some optimization problems are also possible.

References

- [1] J. Czyzowicz, W. Fraczak, M. Yazdani, Throughput of high-performance concatenation state machines, in: Proceedings of the sixteenth australasian workshop on combinatorial algorithms (AWOCA2005), 2005, pp. 85–94.
- [2] M. Yazdani, W. Fraczak, F. Welfeld, I. Lambadaris, A criterion for speed evaluation of content inspection engines, in: Fifth International Conference on Networking (ICN 2006), IEEE Computer Society, 2006, pp. 19–24.
- [3] A. Dasdan, R. Gupta, Faster maximum and minimum mean cycle algorithms for system-performance analysis, *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 17 (10) (1998) 889–899.
- [4] R. Karp, A characterization of the minimum cycle mean in a digraph, *Discrete Mathematics* 23 (1978) 309–311.
- [5] R. J. Parikh, On context-free languages, *J. ACM* 13 (4) (1966) 570–581.
- [6] M. W. Hopkins, D. Kozen, Parikh’s theorem in commutative kleene algebra, in: *Logic in Computer Science, LICS 1999*, 1999, pp. 394–401.
- [7] J. Esparza, S. Kiefer, M. Luttenberger, On fixed point equations over commutative semirings, in: *24th International Symposium on Theoretical Aspects of Computer Science, STACS 2007*, Vol. 4393 of LNCS, Springer, 2007, pp. 296–307.