On the Computational Complexity of Infinite Words

Research supported by DAAD and Academy of Finland
under a common grant No. 864524.

Pavol Ďuriš

Dept. of Computer Science I (Algorithms and Complexity),
Univ. of Technology, Ahornstrasse 55, Aachen, Germany
(duris@dcs.fmph.uniba.sk)

Ján Maňuch

TUCS and Dept. of Math., Univ. of Turku,
Datacity, Lemminkäisenkatu 14A, Turku, Finland
(manuch@cs.utu.fi)

Turku Centre for Computer Science
TUCS Technical Report No 423
October 2001

ISBN 952-12-0892-9
ISSN 1239-1891

Abstract

This paper contains answers to several problems in the theory of the computational complexity of infinite words. We show that the problem whether all infinite words generated by iterating dgsm's have logarithmic space complexity is equivalent to the open problem asking whether the unary classes of languages in P and in DLOG are equivalent. Similarly, the problem to find a concrete infinite word which cannot be generated in logarithmic space is equivalent to the problem to find a concrete language which does not belong to DSPACE(n). Finally, we separate classes of infinite words generated by double and triple D0L TAG systems.

TUCS Research Group

Mathematical Structures in Computer Science

Introduction

The study of infinite words is an important research topic in Combinatorics on words. The main stream of research focused on combinatorial properties, cf. [Lo], and the descriptional complexity of infinite words, i.e., the measure how complicated simple mechanisms are needed to generate particular infinite words. An extensive study of different machineries for infinite word generation can be found in [CK].

In [HKL] a new area of investigation was introduced, the computational complexity of words, i.e., the measure how much resources (such as time and space) are needed to generate a certain infinite word. The paper concentrates on relations between the two mentioned complexities: descriptional and computational. Further results in this area can be found in [HK].

In [CK], [HKL] and [HK] several interesting problems are proposed. In this paper we will show that even some of the simplest problems proposed are equivalent to well-known hard open problems in complexity theory of Turing machines.

The paper is organized as follows:
In Section 1 we fix our terminology. In Section 1.1 we recall the definition of the computational complexity, while in Section 1.2 we define several simple methods for generating infinite words:

- iterating morphism, the most commonly used method introduced already in [Th];
- iterating a deterministic generalized sequential machine (a dgsm), i.e. a deterministic finite state transducer;
- double and triple D0L TAG systems.

In Section 2.1 we consider the open problem, proposed in [HKL], namely whether all infinite words generated by iterating dgsm's have logarithmic space complexity. This problem was attacked in [Le], claiming that the answer to the problem is affirmative. On the other hand, here we show that the problem is equivalent to an other hard open problem asking whether unary classes of languages P and DLOG (denoted u-P and u-DLOG, respectively) are equivalent. One can easily observe that u-P = u-DLOG if and only if $\cup_{c>0}\,\mathrm{DTIME}(c^n) = \mathrm{DSPACE}(n)$.

Section 2.1 contains two other small results concerning the growth of dgsm's, It was shown in [HKL] that the greatest possible growth of a dgsm is exponential and that infinite words generated by such dgsm's have logarithmic space complexity. We show that the smallest non-trivial growth is $\Theta(n \log n)$ and that, similarly, dgsm's with such growth generate infinite words with logarithmic space complexity.

In [HKL] an another interesting problem is proposed: to find a concrete infinite word which cannot be generated in the logarithmic space. It is mentioned already in [HK] that this problem is at least as hard as to prove $L \notin$ DLOG for some $L \in$ NP. In Section 2.2 we show that it is exactly as hard as the problem to find a concrete language, which does not belong to DSPACE(n). Note that even the problem to find a concrete language, which does not belong to DSPACE($\log n$) = DLOG is a hard open problem.

Finally, in Section 2.3 we separate the classes of infinite words generated by double and triple D0L TAG systems as it was conjectured in [CK].

1 Preliminaries

In this section we fix our terminology. Let Σ be a finite alphabet. The sets of all finite and infinite words over Σ are denoted Σ^* and $\Sigma^{\mathbb{N}}$, respectively. In the following sections we define the computational complexity of infinite words and describe several iterative devices generating infinite words.

1.1 The computational complexity of infinite words

The best way how to define the computational complexity of an object is to describe it in the terms of Turing machines. For example, the Kolmogorov complexity of a finite word is the size of the smallest Turing machine generating the word, cf. [Ko]. In the case of infinite words we will use the model of computation based on *k-tape Turing machine*, which consists of

1. a finite state control;
2. k infinite one-way working tapes each containing one two-way read/write head;
3. one infinite output tape containing one one-way write-only head.

We assume that k-tape TM starts in the initial state with all tapes empty and behaves as normal Turing machine. We say that k-tape TM generates an infinite word $w \in \Sigma^{\mathbb{N}}$ if

1. in each step of computation, the content of the output tape is a prefix of w;
2. for each u, prefix of w, there is an integer n such that u is a prefix of the content of the output tape after n steps of computation.

Let M be a k-tape TM generating a word w. The *time* and *space complexities* of M are functions $T_M : \mathbb{N} \to \mathbb{N}$ and $S_M : \mathbb{N} \to \mathbb{N}$ defined as follows:

- $T_M(n)$ is the smallest number of steps of the computation of M when the prefix of w of length n is already written on the output tape;

- $S_M(n)$ is the space complexity of working tapes during first $T_M(n)$ steps of the computation, i.e., the maximum of lengths of words written on working tapes.

Finally, for any integer function $s : \mathbb{N} \to \mathbb{N}$ we define the following complexity classes:

- $\text{GTIME}(s) = \{w \in \Sigma^{\mathbb{N}}$; there exists a k-tape TM M generating w and $T_M(n) \le s(n)$ for all $n \ge 1\}$;

- $\text{GSPACE}(s) = \{w \in \Sigma^{\mathbb{N}}$; there exists a k-tape TM M generating w and $S_M(n) \le s(n)$ for all $n \ge 1\}$;

It follows from the speed-up argument, as in ordinary complexity theory, that functions $s(n)$ and $c.s(n)$, where c is a constant, have the same space computational complexities.

1.2 Iterative devices generating infinite words

In this section we define several simple methods used for generating infinite words. The simplest and most commonly used method is to iterate a morphism $h : \Sigma^* \to \Sigma^*$: if h is nonerasing and for a letter $a \in \Sigma$, a is a prefix of $h(a)$, then there exists the limit

$$w = \lim_{n \to \infty} h^n(a)\,.$$

A natural generalization of this method is to use more powerful mapping in the iteration, a deterministic gsm. As a convention we assume throughout that all dgsm's are non-erasing.

The further generalization leads to *double D0L TAG system* which consist of two infinite one-way tapes each of which containing a one-way read-only head and a one-way write-only head. In each step of the generation both read-only heads read a symbol and move right to the next square while the write-only heads write the corresponding outputs to the first empty squares of these tapes. We assume that the infinite word generated by a double D0L TAG system is written on the first tape. A double D0L TAG system can be specified in the terms of the rewriting rules of the form:

$$\binom{a}{b} \to \binom{\alpha}{\beta}, \text{ where } a, b \in \Sigma,\ \alpha, \beta \in \Sigma^+.$$

Assuming that in each rewriting rule, $|\beta| = 1$, we get a mechanism which iterates a dgsm. Finally, we can define *triple D0L TAG system* by extending the number of tapes to three.

2 The results

2.1 Dgsm's

In [HKL] the following problem is proposed: are all words generated by dgsm's in GSPACE($\log n$)? We prove here that this problem is equivalent to the problem whether classes u-DLOG and u-P are equivalent. Note that this in some sense contradicts the result of [Le] that all infinite words generated by iterating dgsm's have logarithmic space complexity: if the result in [Le] is correct, then, together with the following theorem, we have D-EXPTIME = DSPACE(n), which is unlikely. [Le] gives only the sketch of the proof of the result and, hence, we are unable to check if it is correct.

Theorem 1. *All infinite words generated by iterating dgsm's have logarithmic space complexity if and only if* u-P = u-DLOG.

Proof. First, let us assume that u-P = u-DLOG. Take a dgsm τ over finite alphabet Σ generating an infinite word $w = w_1 w_2 \dots$. We prove that the space complexity of w is $\mathcal{O}(\log n)$. It is obvious that there is a 1-tape Turing machine M generating the word w in quadratic time. Consider the languages $L_c = \{0^n;\ n \geq 1,\ w_n = c\}$ for all $c \in \Sigma$. Note that L_c is an unary language. We can easily construct a Turing machine recognizing L_c in quadratic time using Turing machine M. By the assumption there exist Turing machines M_c recognizing the languages L_c in logarithmic space. Now, consider 3-tape TM, which runs M_c's to generate n-th letter of w by using the third tape as working tape. It stores the binary representation of n on the first tape and the position of the head of TM M_c on the input tape on the second tape. Before each run of any M_c it erases the working tape and writes on the second tape position 1. It runs M_c for each letter c of the alphabet of τ until some M_c accepts and then it writes the letter c on the output tape. In each step of the simulation of any M_c it checks whether the position represented on the second tape is the last one. Clearly such machine generates the word w in logarithmic space.

Second, assume that all words generated by iterating a dgsm have logarithmic space complexity. Take a Turing machine M working in polynomial time, i.e., $T(M) = \mathcal{O}(n^k)$, recognizing a language $L \subseteq 0^*$. We construct 1-tape TM M' for generating an infinite word with the tape divided into three layers. On the first layer it generates unary inputs in increasing order, on the second layer it simulates computations of M on the input stored on the first layer, and on the third layer it writes 1, if computation ends in an accepting state, or 0, if it ends in a rejecting state. Before each simulation it erases the second and third layer of the tape.

Now, consider a dgsm τ which carries out the computations

$$C_i \to C_{i+1}, \text{ for } i \geq 0,$$

where C_i corresponds to the i-th configuration of M'. It also maps the starting letter \$ into the starting configuration of M'. Clearly, the iteration of τ will generate infinite word, the sequence of all configurations of TM M': $W = \$C_0C_1C_2\ldots$. By the assumption the infinite word W has logarithmic space complexity, i.e., there exists a TM M'' generating W in logarithmic space. Finally, we define TM M''' recognizing L, which on the input 0^n runs M'', but instead of writing the bits of W to the output tape, it compares its input with the input on the first layer of each generated configuration, and moreover, it checks the first letter on the third layer of each generated configuration. When the compared inputs coincide and the first letter on the third layer is 0 or 1 then TM M''' halts in the rejecting or in the accepting state, respectively. Otherwise, it continues in generating bits of the next configuration. Clearly, M''' recognize the language L.

Now, it suffices to show that M''' works in logarithmic space. Let C_{i_n} be the configuration in which M' writes 0 or 1 on the first place of the third layer, when the first layer contains 0^n. Hence in the block of configurations $B_n = C_{i_{n-1}+1}\ldots C_{i_n}$, M' erases the second and the third layer of the tape, changes the input on the first layer to 0^n and runs TM M on this input. Since M works in time $\mathcal{O}(n^k)$ the length of any configuration in block B_n is at most $\mathcal{O}(n^k)$ and the number of configurations in B_n is at most $\mathcal{O}(n^k)$. Hence the length of the block B_n is at most $\mathcal{O}(n^{2k})$. Turing machine M''' on the input 0^x generates first x blocks of configurations until it halts. Hence it generates the prefix of the infinite word W of length

$$\sum_{n=0}^{x} |B_n| = \mathcal{O}(x^{2k+1})\,.$$

Since M'' works in logarithmic space, M''' will use space $\mathcal{O}(\log x)$ to carry out the computation on the input 0^x. □

Next, we will show two small results concerning the growth of a dgsm τ, i.e., the integer function $g : \mathbb{N} \to \mathbb{N}$, where $g(n)$ is the length of $\tau^n(a)$. In [HKL] it was proved that the infinite word generated by a dgsm with the exponential growth has logarithmic space complexity. Note that the exponential growth is the greatest possible growth. Next, we show that the smallest non-trivial growth of a dgsm is $\Theta(n \log n)$ and also that dgsm's with the growth $\Theta(n \log n)$ generate infinite words with logarithmic space complexity. More precisely:

Lemma 1. *If a dgsm has the growth $o(n \log n)$, then it generates an ultimately periodic infinite word. Such word can be generated in constant space.*

Proof. Let τ be a dgsm with the growth $o(n \log n)$ generating an infinite word $w = w_1w_2\ldots$, with $w_i \in \Sigma$. Let $\tau_q(z)$ [$\sigma_q(z)$] be the output [the last state] of dgsm τ when reading the input z and starting in the state q.

We define the two sequences of words and one sequence of states of dgsm τ. Let a be the starting symbol, q_0 be the initial state and let $\tau(a) = \tau_{q_0}(a) = av$. Then put:

$$\begin{aligned} u_1 &= a, & v_1 &= v, & q_1 &= \sigma_{q_0}(a) \\ u_n &= u_{n-1}v_{n-1}, & v_n &= \tau_{q_{n-1}}(v_{n-1}), & q_n &= \sigma_{q_{n-1}}(v_{n-1}). \end{aligned}$$

Observe that $\tau^n(a) = u_{n+1} = u_n v_n = \cdots = u_1 v_1 v_2 \ldots v_n$. This implies $1 + \sum_{j=1}^{n} |v_j| = |\tau^n(a)| = o(n \log n)$. Next, we try to estimate the length of the increment $|v_n|$. Since dgsm τ is non-erasing, for all $i < n$ we have $|v_i| < |v_n|$. We have:

$$n|v_n| \leq \sum_{j=n+1}^{2n} |v_j| \leq \sum_{j=1}^{2n} |v_j| = o(2n \log 2n) = o(n \log n)\,.$$

Hence, $|v_n| = o(\log n)$ and for any constant c there exists n such that $c^{|v_n|} < n$. If we take $c = |\Sigma| + 1$, there must be a repetition among the words $v_1, \ldots, v_n$, let say $v_i = v_{i+k}$ for some integers $i, i+k \leq n$ and $k > 0$. Then $v_j = v_{j+k}$ for all $j \geq i$, hence the infinite word $w = u_1 v_1 v_2 v_3 \ldots$ is ultimately periodic and it can be then generated in constant space, cf. [HKL], Lemma 3.1. □

Lemma 2. *An infinite word generated by a dgsm with the growth $\Theta(n \log n)$ has logarithmic space complexity.*

Proof. Let τ be a dgsm with the growth $\Theta(n \log n)$ generating an infinite word $w = w_1 w_2 \ldots$. Consider the sequences $\{u_n\}_{n \geq 0}$, $\{v_n\}_{n \geq 0}$, $\{q_n\}_{n \geq 0}$ from the previous proof.

We construct TM M generating w as follows. In the first step it writes u_1 on the output tape, v_1 on the first tape and it sets to the state q_1. In each following step n it simulates dgsm τ on input written on the first tape starting in the state q_{n-1}. The output of the simulation of τ is written, at the same time, to the output tape and to some temporary tape, so that after the simulation it can be copied back to the first tape. Hence, in the end of step n the first tape contains word v_n. The last state of the simulation of τ is the state q_n, from which the next simulation will start.

The space needed to generate the m-th letter of w is at most $|v_n|$, where n is an integer such that v_n contains the letter w_m, i.e.,

$$\sum_{i=1}^{n-1} |v_i| < m \leq \sum_{i=1}^{n} |v_i|\,.$$

One can show in the same way as in in the proof of Lemma 1 that $|v_n| = \mathcal{O}(\log n)$. Moreover, since $n \leq 1 + \sum_{i=1}^{n-1} |v_i| \leq m$, we have $|v_n| = \mathcal{O}(\log n) = \mathcal{O}(\log m)$. Hence, TM M works in logarithmic space. □

2.2 Logarithmic space complexity

The second part of Problem 5.2 in [HK] asks to find a specific infinite word which cannot be generated in logarithmic space. We show that this problem is as hard as the problem to find a specific language which does not belong to DSPACE(n), and this is a hard open problem.

Notation 1. Denote the n-th binary word in the lexicographical order by $\mathrm{lex}(n)$. Note that for $n \geq 1$, $\mathrm{bin}(n) = 1\,\mathrm{lex}(n)$.

Definition 1. Let w be an infinite binary word and $L \subseteq \{0,1\}^*$ an binary language. We say that w *determines* language L if for every positive integer n: the n-th letter of w is 1 if and only if $\mathrm{lex}(n)$ belongs to L.

Theorem 2. *Let w be an infinite binary word and L the language determined by the word w. Then, the word w is in* GSPACE($\log n$) *if and only if L belongs to* DSPACE(n)*.*

Proof. First, assume that w has logarithmic space complexity. Let M be a Turing machine generating w in logarithmic space. We construct Turing machine M' recognizing language L. Let $\mathrm{lex}(n)$ be the word on the input tape of M', where n is a positive integer. The length of input is $\Theta(\log n)$. M' simulates M in the following way: it remembers only the last letter generated by M and counts the number of them on a special working tape. When this number is equal to n, it stops and accepts the input if and only if the last output letter was 1.

Since M uses space only $\Theta(\log n)$ to generate first n output letters and the same space is needed for counting of the number of output letters, M' works in space $\Theta(\log n)$, which is linear to the length of input. Hence, $L \in$ DSPACE(n).

Next, assume that $L \in$ DSPACE(n). So we have a Turing machine M recognizing L in linear space. Let M' be a Turing machine such that it generates words in lexicographical order on the first working tape and runs M on each generated word. Depending on if the word was accepted or rejected it writes 1 or 0 on the output tape. Clearly, the length of the n-th word on the first working tape is $\Theta(\log n)$. M works in linear space, hence in space $\Theta(\log n)$. Therefore, M' uses logarithmic space to generate n-th letter: $w \in$ GSPACE($\log n$). $\square$

As a consequence we have that if we would be able to show about a specific infinite word that it does not belong to GSPACE($\log n$), then we would have also a specific language which does not belong to DSPACE(n), and vice versa. Note, that even the problem to show that a specific language does not belong to DSPACE($\log n$) is open.

2.3 Separation of double and triple D0L TAG systems

In the Section 6 of [HKL] is mentioned that the generation of infinite words by double D0L TAG systems is very powerful mechanism, and that it is not known any concrete example of an infinite word which cannot be generated by this mechanism, although by a diagonalization argument such words clearly exist. In [CK] (Conjecture 4) is conjectured that there exists an infinite word that can be generated by a triple D0L TAG system, but not by any double D0L TAG system. In what follows we are going to give the whole class of infinite words which cannot be generated by any double D0L TAG system. Combining this result with some results in [CK] we can also give an affirmative answer to Conjecture 4 of [CK]. First, let us fix some notation.

Notation 2. Let $w = c_1 \dots c_n$ be a word with $c_i \in \Sigma$. Then $\mathrm{symb}(w) = \{c_i,\ 1 \le i \le n\}$ denotes the set of all symbols occurring in the word.

Theorem 3. *Let $s : \mathbb{N} \to \mathbb{N}$ be an integer function such that $s(i) \in 2^{\omega(i)}$, i.e., $s(i)$ grows faster than exponentially. Then the infinite word*

$$w = 10^{s(1)}10^{s(2)}10^{s(3)}\dots$$

cannot be generated by any double D0L TAG system.

Proof. Assume that w can be generated by double D0L TAG system, and let τ be such a system. Let M be the set of all symbols occurring on the second tape of τ and B the maximal number of symbols written on any tape in one step, i.e.,

$$B = \max\{\max(|\alpha|, |\beta|),\ \text{where } \begin{pmatrix} a \\ b \end{pmatrix} \to \begin{pmatrix} \alpha \\ \beta \end{pmatrix} \text{ is a rewriting rule of } \tau\}\,.$$

Let $w_i = 10^{s(i)}$. First, we show that for any constant $k \ge 1$ there is a positive integer j such that:

$$s(j) + 1 = |w_j| > k|w_1 \dots w_{j-1}1| + 1\,. \tag{1}$$

Since $s(i) \in 2^{\omega(i)}$, for any number $c > 1$ there is an integer j such that:

$$\begin{aligned} &|w_j| > c^j, \\ &|w_i| \le c^i \text{ for all } 1 \le i \le j-1. \end{aligned}$$

Then we have

$$k|w_1 \dots w_{j-1}1| \le k \sum_{i=0}^{j-1} c^i \le k \cdot \frac{c^j - 1}{c - 1}\,.$$

Taking $c = k + 1$, we get $k|w_1 \dots w_{j-1}1| + 1 \le c^j < |w_j|$.

Consider that we are reading the first 0 of $w_j = 10^{s(j)}$ in the word w. Let u_1 (v_1) be the word written on the first (second) tape between the reading and the writing head. And let, recursively, u_l (resp. v_l) be the word added on the first (resp. second) tape after reading v_{l-1}. Note that for all $l \geq 1$, we have $|v_1 \dots v_{l-1}| < |u_1 \dots u_l|$.

We define also set $M_0 \subseteq M$ as follows. For $x \in M$, let $\binom{0}{x} \to \binom{\alpha}{\beta}$ be a rule of τ. Then, $x \in M_0$ if and only if $\alpha \in 0^+$. Hence, M_0 is the set of all symbols such that when reading 0 on the first tape and such symbol on the second tape, it writes only 0's on the first tape.

Assume that for some $i \geq 1$:

$$\sum_{l=1}^{i} |u_l| \leq s(j)\,. \tag{2}$$

By (2), words $u_1, \dots, u_i$ contain only 0's, hence when reading words $v_1, \dots, v_{i-1}$, only 0's are written on the first tape. Since $|v_1 \dots v_{l-1}| < |u_1 \dots u_l|$, we have also that while reading words $v_1, \dots, v_{i-1}$, only 0's are read from the first tape.

Consider the following oriented graph. The vertices are elements of M. There is an arc $x \to y$, if for rule $\binom{0}{x} \to \binom{\alpha}{\beta}$, $y \in \mathrm{symb}(\beta)$ (see Notation 2). For $X \subseteq M$, let $\mathrm{clos}(X)$ be the set of all vertices of the graph to which we can get from any vertex of X following the arcs. If $\mathrm{clos}(\mathrm{symb}(v_1)) \subseteq M_0$, then since $u_1 \in 0^+$, we get by induction that $u_l \in 0^+$ and $\mathrm{symb}(v_l) \subseteq \mathrm{clos}(\mathrm{symb}(v_1)) \subseteq M_0$ for all $l \geq 1$. This is a contradiction since, there must be u_l containing 1.

Hence, there must be an oriented path starting in a vertex of $\mathrm{symb}(v_1)$ and ending in a vertex of $M - M_0$ of length $i_0 \leq |M_0|$. Let i_0 be the shortest such path. If we prove that (2) holds true for $i = |M| + 1 \geq |M_0| + 2 \geq i_0 + 2$, then since during reading $v_1, \dots, v_{i_0+1}$ only 0's are read and written on the first tape, we have $\mathrm{symb}(v_1), \dots, \mathrm{symb}(v_{i_0}) \subseteq M_0$ and $\mathrm{symb}(v_{i_0+1}) \cap (M - M_0) \neq \emptyset$. This implies that $1 \in \mathrm{symb}(u_{i_0+2})$, which is a contradiction with (2).

Now it suffices to prove that Equation (2) holds for $i = |M| + 1$. After reading one symbol, system τ can write on any tape at most B symbols. Hence, we have for all $l \geq 1$, $|u_{l+1}|, |v_{l+1}| \leq B|v_l|$. We estimate the left hand side of (2):

$$\sum_{l=1}^{|M|+1} |u_l| \leq |u_1| + \sum_{l=2}^{|M|+1} B^{l-1}|v_1| \leq \max(|u_1|, |v_1|).\sum_{l=0}^{|M|} B^l\,. \tag{3}$$

Notice that $T = \sum_{l=0}^{|M|} B^l$ is a constant for τ.

Next, we try to estimate $|u_1|$ and $|v_1|$. Consider the situation when the reading heads are on the $(n+1)$-th symbols of both tapes. Then n symbols have been already read and hence at most Bn symbols written. So, there is at most $(B-1)n$ symbols between writing and reading head on any tape. This implies:

$$\max(|u_1|, |v_1|) \leq (B-1)|w_1 \dots w_{j-1}1| \,. \tag{4}$$

Taking $k = T(B-1)$ we get

$$\sum_{l=1}^{|M|+1} |u_l| \overset{(3)}{\leq} T.\max(|u_1|, |v_1|) \overset{(4)}{\leq} T(B-1)|w_1 \dots w_{j-1}1| \overset{(1)}{<} s(j)$$

as desired. □

Let us recall one lemma proved in Examples 11 and 13 of [CK]:

Lemma 3. *Let $s : \mathbb{N} \to \mathbb{N}$ be an integer function which is computable, i.e., can be computed by a TM. Then there exists $t : \mathbb{N} \to \mathbb{N}$ such that $t(n) \geq s(n)$ for all $n \geq 1$ and the word*

$$w = 10^{t(1)}10^{t(2)}10^{t(3)} \dots$$

can be generated by a triple D0L TAG system. Moreover, such integer function t can be effectively computed.

The proof of the lemma is based on the following idea. Let M be a TM computing unary strings $1^{s(1)}, 1^{s(2)}, \dots$ and let τ be a dgsm generating the sequence of configurations of computation of M. We can easily extend the dgsm τ to a triple D0L TAG system by coding all letters generated by τ to 0, except for the last letters of the strings in the sequence $1^{s(1)}, 1^{s(2)}, \dots$, which are coded to 1. Together with our result we have the following corollary:

Corollary 1. *There exists an infinite word which can be generated by a triple D0L TAG system, but not by any double D0L TAG system.*

Hence, the inclusion "double DOL ⊆ triple D0L" is proper, as conjectured in Conjecture 4 in [CK].

References

[HK] Hromkovič, J., Karhumäki, J., *Two lower bounds on computational complexity of infinite words*, New trends in formal languages, LNCS 1218, 366–376, 1997.

[HKL] Hromkovič, J., Karhumäki, J., Lepistö, A., *Comparing descriptional and computational complexity of infinite words*, Results and trends in theoretical computer science, LNCS 812, 169–182, 1994.

[CK] Culik, K., II, Karhumäki, J., *Iterative devices generating infinite words*, Int. J. Found. Comput. Sci., Vol. 5 No. 1, 69-97, 1994.

[Le] Lepistö, A., *On the computational complexity of infinite words*, In: Developments in Language Theory II (eds.: J. Dassow, G.Rozenberg, A. Salomaa), World-Scientific, Singapore, 350–359, 1996.

[Ko] Kolmogorov, A.N., *Three approaches to the quantitative definition of information*, Problems Inform. Transmission 1, 662–664, 1968.

[Lo] Lothaire, M., *Combinatorics on words*, Encyclopedia of Mathematics and Its Applications, Vol. 17, Reading, Massachusetts, Addison-Wesley, 1983.

[Th] Thue, A., *Über unendliche zeichenreihen*, Norske Vid. Selsk. Skr., I Mat. Nat. KI., Kristiania 7, 1–22, 1906.

Turku Centre for Computer Science
Lemminkäisenkatu 14
FIN-20520 Turku
Finland

http://www.tucs.abo.fi

University of Turku
- **Department of Mathematical Sciences**

Åbo Akademi University
- **Department of Computer Science**
- **Institute for Advanced Management Systems Research**

Turku School of Economics and Business Administration
- **Institute of Information Systems Science**