SCIENTIFIC PAPERS, UNIVERSITY OF LATVIA, 2012. Vol. 787
COMPUTER SCIENCE AND INFORMATION TECHNOLOGIES 125–135 P.

Frequency Computable Relations [⋆]

Madara Augstkalne, Anda Beriņa, Rūsiņš Freivalds,

Institute of Mathematics and Computer Science, University of Latvia,
Raiņa Bulvāris 29, Riga, LV-1459, Latvia

Abstract. A transducer is a finite-state automaton with an input and an output. We compare possibilities of nondeterministic and probabilistic transducers, and prove several theorems which establish an infinite hierarchy of relations computed by these transducers. We consider only left-total relations (where for each input value there is exactly one allowed output value) and Las Vegas probabilistic transducers (for which the probability of any false answer is 0). It may seem that such limitations allow determinization of these transducers. Nonetheless, quite the opposite is proved; we show a relation which can only be computed by probabilistic (but not deterministic) transducers, and one that can only be computed by nondeterministic (but not probabilistic) transducers.
Frequency computation was introduced by Rose and McNaughton in early sixties and developed by Trakhtenbrot, Kinber, Degtev, Wechsung, Hinrichs and others. It turns out that for transducers there is an infinite hierarchy of relations computable by frequency transducers and this hierarchy differs very much from similar hierarchies for frequency computation by a) Turing machines, b) polynomial time Turing machines, c) finite state acceptors.

1 Introduction

Frequency computation was introduced by G. Rose [13] as an attempt to have a deterministic mechanism with properties similar to probabilistic algorithms. The definition was as follows. A function $f\colon w \to w$ is (m, n)-computable, where $1 \leq m \leq n$, iff there exists a recursive function $R\colon w^n \to w^n$ such that, for all n-tuples $(x_1, \cdots, x_n)$ of distinct natural numbers,

$$card\{i : (R(x_1, \cdots, x_n))_i = f(x_i)\} \geq m.$$

R. McNaughton cites in his survey [12] a problem (posed by J. Myhill) whether f has to be recursive if m is close to n. This problem was answered by B.A. Trakhtenbrot [14] by showing that f is recursive whenever $2m > n$. On the other hand, in [14] it was proved that with $2m = n$ nonrecursive functions can be (m, n)-computed. E.B. Kinber extended the research by considering frequency enumeration of sets [9]. The class of (m, n)-computable sets equals the class of recursive sets if and only if $2m > n$.

[⋆] The research was supported by Grant No. 09.1570 from the Latvian Council of Science.

For resource bounded computations, frequency computability behaves differently. For example, it is known that whenever $n' - m' > n - m$, under any reasonable resource bound there are sets which are (m', n')-computable, but not (m, n)-computable. However, for finite automata, an analogue of Trakhtenbrot's result holds: the class of languages (m, n)-recognizable by deterministic finite automata equals the class of regular languages if and only if $2m > n$. Conversely, for $2m \leq n$, the class of languages (m, n)-recognizable by deterministic finite automata is uncountable for a two-letter alphabet [1]. When restricted to a one-letter alphabet, every (m, n)-recognizable language is regular. This was also shown by Kinber.

Frequency computations became increasingly popular when the relation between frequency computation and computation with a small number of queries was discovered [11, 6, 2, 3].

We considered problems similar to those in the classical papers [14, 9, 1] for finite-state transducers. We found the situation to be significantly different.

A finite state transducer is a finite state machine with two tapes: an input tape and an output tape. These tapes are one-way, i.e. the automaton never returns to the symbols once read or written. Transducers compute relations between the input words and output words. A deterministic transducer produces exactly one output word for every input word processed.

In this paper we consider advantages and disadvantages of nondeterministic, deterministic, frequency and probabilistic transducers. Obviously, if a relation is such that several output words are possible for the same input word, then the relation cannot be computed by a deterministic transducer. For this reason, in this paper we restrict ourselves to relations which produce exactly one output word for every input word processed.

Definition 1. *We say that a relation $R(x, y)$ is* left-total*, if for arbitrary x there is exactly one y satisfying $R(x, y)$.*

Probabilistic algorithms may be of several types: those allowing errors of all types, Monte Carlo, Las Vegas, etc. Since our relations produce exactly one output word for every input word processed, it was natural to consider only Las Vegas transducers, i.e. probabilistic transducers for which the probability of every false answer is 0. It follows immediately from the definition that every relation computed by such a probabilistic transducer can be computed by a nondeterministic transducer as well. We prove below that nondeterministic transducers are strictly more powerful than Las Vegas transducers.

The zero probability of all possible false results of a probabilistic transducer has another unexpected consequence. It turns out that only recursive relations of small computational complexity can be computed by probabilistic transducers with a probability, say, $\frac{1}{4}$ or $\frac{1}{7}$. Hence a natural question arises, whether every rational number $0 \leq p \leq 1$ can be the best probability to compute some relation by a probabilistic finite-state transducer. It turns out that it is not the case. We show examples of relations that can be computed by probabilistic finite-state transducers with probabilities $\frac{1}{2}$, $\frac{1}{3}$, $\frac{1}{4}$, etc. and not better. We believe that no other best probabilities can exist for relations of the type considered by us.

This hierarchy of relations turns out to be related to the number of symbols of help needed for deterministic finite-state transducers that take advice to compute these relations.

2 Definitions

We use standard definitions of deterministic, nondeterministic and probabilistic transducers, which are well-established in theoretical computer science literature [5]. Our model is slightly different in the regard that we allow multiple output symbols for each transition, however it can be easily seen that the expressive power of transducer is unaffected.

Definition 2. *A nondeterministic transducer D is a six-tuple $\langle Q, \varSigma, \varDelta, \delta, q_0, F\rangle$, which satisfies the following conditions:*

- *Q is a finite set, whose members are called states of D. $q_0 \in Q$ is called the initial state of D,*
- *$\varSigma$ is a set of input symbols of D and is called the input alphabet,*
- *$\varDelta$ is a set of output symbols of D and is called the output alphabet,*
- *δ is a relation from $Q \times (\varSigma \cup \{\epsilon\})$ to $Q \times \varDelta^*$. Each tuple $((q, \alpha), (p, \beta))$ is called a transition rule of D,*
- *$F \subseteq Q$ is the set of accepting or final states of D*

A transducer operates in discrete time $t = 1, 2, 3 \ldots$ and handles two one-way tapes, where one is read-only input tape and contains a string from the input alphabet $\varSigma$ and second is write-only output tape with finite output alphabet $\varDelta$.

The computation begins at the start state q_0. The computation from state q proceeds by reading symbol α from the input tape, following a suitable transition rule $((q, \alpha), (p, \beta))$ (the new state being p) and writing the corresponding output β to the output tape. The only exception is so called ϵ-transition $((p, \epsilon), (q, \beta))$, where the transducer changes the state and possibly writes an output without reading an input symbol.

We consider all our transducers as machines working infinetely long. At every moment, let x be the word having been read from the input tape up to this moment, and let y be the word written on the output tape up to this moment. Then we say that the pair (x, y) belongs to the relation computed by the transducer.

Definition 3. *A deterministic transducer is a nondeterministic transducer such that transition relation δ is a function, i.e., for each input state and each input symbol the corresponding output state and output symbol is uniquely determined according to the transition table δ.*

Definition 4. *A probabilistic transducer is a transducer of the form $\langle Q, \varSigma, \varDelta, \varPsi, q_0, F\rangle$ where the transition function $\varPsi$ is probabilistic, i.e., for every state q and input symbol α, output state p and output symbol β there is an* *associated probability with which transition occurs. We furthermore stipulate that probabilities of all transition from any fixed pair of input state and input symbol must sum to one.*

Definition 5. *We say that a probabilistic transducer A computes a left-total relation R with probability p if for arbitrary pair $(x, y) \in R$ when A works on input x the probability to go into an accepting state having produced the output y is no less than p.*

Definition 6. *We say that a probabilistic transducer A is a Las Vegas transducer computing a left-total relation R with probability p if for arbitrary pair $(x, y) \in R$ when A works on input x the probability to go into an accepting state having produced the output y is no less than p, and the probability to go into an accepting state having produced the output other than y equals 0.*

Since we consider in our paper only left-total relations and since our probabilistic transducers are Las Vegas, it easily follows that the transducer for arbitrary x and for arbitrary $\alpha \in \{\alpha_1, \alpha_2, \ldots, \alpha_n\}$ either produces a correct output y or reaches a non-accepting state.

Definition 7. *A frequency transducer with parameters (m, n) is a transducer with n input tapes and n output tapes. Every state of the transducer is defined as $(a_1, a_2, \cdots, a_n)$-accepting where each $a_i \in \{$ accepting , nonaccepting $\}$. We say that a left-total relation R is (m, n)-computed by the transducer if for arbitrary n-tuple of pairwise distinct input words $(x_1, x_2, \cdots, x_n)$ there exist at least m distinct values of x_i such that the i-th output y_i satisfies $(x_i, y_i) \in R$.*

Please notice that we do not demand the state of the frequency transducer to be all-accepting after the reading of the input words. This allows us introduce a counterpart of Las Vegas transducer.

Definition 8. *A Las Vegas frequency transducer with parameters (m, n) is a transducer with n input tapes and n output tapes. Every state of the transducer is defined as $(a_1, a_2, \cdots, a_n)$-accepting where each $a_i \in \{$ accepting , nonaccepting $\}$. We say that a left-total relation R is (m, n)-computed by the transducer if for arbitrary n-tuple of pairwise distinct input words $(x_1, x_2, \cdots, x_n)$: 1) there exist at least m distinct values of x_i such that the i-th output y_i satisfies $(x_i, y_i) \in R$, and 2) if a result y_i is produced on any output tape i and the current state of the transducer is $(a_1, a_2, \cdots, a_{i-1}, accepting, a_{i+1}, \cdots, a_n)$-accepting, then $R(x_i, y_i)$.*

We consider frequency transducers like all other transducers as machines working infinitely long. At every moment, let x be the word having been read from the input tape up to this moment, and let y be the word written (and accepted) on the output tape up to this moment. Then we say that the pair (x, y) belongs to the relation computed by the transducer. However, in the case when the input alphabet contains less than n letters, there is a problem how to define (m, n)-computation correctly.

Definition 9. *We say that a frequency transducer is performing a* strong (m, n)*-computation if* at moments when all the input words $(x_1, x_2, \cdots, x_n)$ are distinct *there exist at least* m *distinct values of* x_i *such that the i-th output* y_i *satisfies* $(x_i, y_i) \in R$.

Definition 10. *We say that a frequency transducer is performing a* weak (m, n)*-computation with parameters* $(b_1, b_2, \cdots b_n)$*, where* $b_1, b_2, \cdots b_n$ *are distinct integers, if the transducer is constructed in such a way that in the beginning of the work the transducer reads the first* b_1 *symbols from the input word* x_1*, the first* b_2 *symbols from the input word* b_2*,* $\cdots$ *, the first* b_n *symbols from the input word* x_n *and at all subsequent moments reads exactly 1 new symbol from every input tape. This ensures that at all moments the input words* $(x_1, x_2, \cdots, x_n)$ *are distinct. There is no requirement of the correctness of the results when the length of input words is* $(b_1, b_2, \cdots b_n)$ *but at all moments afterwards there exist at least* m *distinct values of* x_i *such that the i-th output* y_i *satisfies* $(x_i, y_i) \in R$.

3 Frequency transducers

First of all, it should be noted that a frequency transducer does not specify uniquely the relation computed by it. For instance, a transducer with 3 input tapes and 3 output tapes $(2, 3)$-computing the relation

$$R(x,y) = \begin{cases} \text{true, if } x = y \text{ and } x \neq 258714, \\ \text{true, if } y = 0 \text{ and } x = 258714, \\ \text{false, if} \qquad\qquad \text{otherwise.} \end{cases}$$

can output $y = x$ for all possible inputs and, nonetheless, the result is always correct for at least 2 out of 3 inputs since the inputs always distinct. Please notice that the program of the frequency transducer does not contain the "magical" number 258714. Hence the number of states for an equivalent deterministic transducer can be enormously larger.

Theorem 1. *There exists a strong finite-state frequency transducer* $(1, 2)$*-computing a continuum of left-total relations.*

Proof. Consider the following frequency transducer with 2 input tapes and 2 output tapes. If the inputs x_1 and x_2 are such that x_1 is an initial fragment of x_2, then the output y_1 equals x_1, and $y_2 = 0$. If the inputs x_1 and x_2 are such that x_2 is an initial fragment of x_1, then the output y_2 equals x_2, and $y_1 = 0$. In all the other cases $y_1 = y_2 = 0$.

Let R be a relation defined by taking an arbitrary infinite sequence ω of zeros and ones, and defining

$$R(x,y) = \begin{cases} \text{true, if } y = x \text{ and } x \text{ is an initial fragment of } \omega, \\ \text{true, if } y = 0 \text{ and } x \text{ is not an initial fragment of } \omega, \\ \text{false, if} \qquad\qquad \text{otherwise.} \end{cases}$$

There is a continuum of such sequences and a continuum of the corresponding relations. Each of them is $(1, 2)$-computed by the frequency transducer. □

Theorem 2. *If $2m > n$ and there exists a strong finite-state frequency transducer (m, n)-computing a left-total relation R then R can also be computed by a deterministic finite-state transducer.*

Proof. Let R be a left-total relation and A be a strong finite-state frequency transducer (m, n)-computing it. Let Q be (another) left-total relation (m, n)-computed by the same transducer A. Let $x_1, x_2, \cdots, x_k$ be distinct input words such that the pairs $(x_1, y_1'), (x_2, y_2'), \cdots, (x_k, y_k')$ are in R, the pairs (x_1, y_1''), $(x_2, y_2''), \cdots, (x_k, y_k'')$ are in Q, and $y_1' \neq y_1'', y_2' \neq y_2'', \cdots, y_k' \neq y_k''$. What happens if $k \geq n$ and the transducer gets an input-tuple containing only values from $\{x_1, x_2, \cdots, x_k\}$? The transducer has to output at least m correct values for R and at least m correct values for Q which is impossible because $2m > n$. A careful count shows that $k \leq 2n - 2m$.

Hence for arbitrary given relation R (m, n)-computable by A there is a constant k such that any relation Q (m, n)-computable by A differs from R on no more that k_0 pairs (x, y). Let Q_0 be one of the relations where indeed k such pairs exist, and let $(x_1, y_1'), (x_2, y_2'), \cdots, (x_k, y_k')$ and (x_1, y_1''), $(x_2, y_2''), \cdots, (x_k, y_k'')$ be these pairs. Of course, there is no algorithm to construct k and Q_0 effectively, but they cannot fail to exist. For arbitrary x the value of y such that $(x, y) \in R$ can be calculated effectively using n-tuples of input words involving the input words $x_1, x_2, \cdots, x_k$ and remembering that no relation computed by A can differ from R and from Q_0 in more than k values. Since A is a finite automaton, the standard cut-and-paste arguments show that this computation needs only input words which differ from x only on the last d digits where d is a suitable constant. □

Theorem 3. *There exists a left-total relation R $(2, 3)$-computed by a weak finite-state Las Vegas frequency transducer and not computed by any deterministic finite-state transducer.*

Proof. We consider the relation

$$R(x, y) = \begin{cases} \text{true, if } y = 1^{|x|} & \text{and } |\, x \,| \equiv 0 (mod 3), \\ \text{true, if } y = 1^{|x|} & \text{and } |\, x \,| \equiv 1 (mod 3) \\ \text{true, if } y = 1^{|x|-1} & \text{and } |\, x \,| \equiv 2 (mod 3) \\ \text{false, if} & \text{otherwise.} \end{cases}$$

The weak finite-state Las Vegas frequency $(2, 3)$-transducer starts with reading 1 symbol from the first input tape, 2 symbols from the second input tape and 3 symbols from the third input tape and reads exactly 1 symbol from each input tape at any moment. Hence at any moment the transducer has no more than one input word x_i with $|\, x_i \,| \equiv 2 (mod 3)$. To have a correct result for the other two input words, it suffices to keep the length of the output being equal the length of the corresponding input. In case if the length of the input is $|\, x_i \,| \equiv 2 (mod 3)$, the state becomes non accepting.

The relation cannot be computed by a deterministic transducer because the length of the output y decreases when the length of of the input increases from $3k + 1$ to $3k + 2$. □

This proof can easily be extended to prove the following Theorem 4.

Theorem 4. *If $m < n$ then there exists a left-total relation R (m,n)-computed by a weak finite-state Las Vegas frequency transducer and not computed by any deterministic finite-state transducer.*

Theorem 5. *There exists a left-total relation R $(2,4)$-computed by a weak finite-state Las Vegas frequency transducer and not computed by any finite-state $(1,2)$-frequency transducer.*

Proof. We consider the relation

$$R(x,y) = \begin{cases} \text{true, if } y = 1^{|x|} & \text{and } \mid x \mid \equiv 0(mod 9), \\ \text{true, if } y = 1^{|x|} & \text{and } \mid x \mid \equiv 3(mod 9), \\ \text{true, if } y = 1^{|x|} & \text{and } \mid x \mid \equiv 4(mod 9), \\ \text{true, if } y = 1^{|x|} & \text{and } \mid x \mid \equiv 6(mod 9), \\ \text{true, if } y = 1^{|x|} & \text{and } \mid x \mid \equiv 8(mod 9), \\ \text{true, if } y = 0 & \text{and } \mid x \mid \equiv 1(mod 9), \\ \text{true, if } y = 0 & \text{and } \mid x \mid \equiv 2(mod 9), \\ \text{true, if } y = 0 & \text{and } \mid x \mid \equiv 5(mod 9), \\ \text{true, if } y = 0 & \text{and } \mid x \mid \equiv 7(mod 9), \\ \text{false, if} & \text{otherwise.} \end{cases}$$

(1) The weak finite-state Las Vegas frequency $(2,4)$-transducer starts with reading 1 symbol from the first input tape, 2 symbols from the second input tape and 3 symbols from the third input tape, 4 symbols from fourth input tape and reads exactly 1 symbol from each input tape at any moment. The transducer always outputs $y_i = 1^{|x_i|}$ on the i-th output tape. Since the transducer can count the length of the input modulo 9, the false outputs (in cases $\mid x_i \mid$ congruent to 1,2,5 or 7 (mod 9)) are not accepted and the transducer is Las Vegas.

At every moment the lengths of the input words are $k, k+1, k+2, k+3$ for some natural k. At least two of them are congruent to 0,3,4,6 or 8 (mod 9).

(2) Assume that the relation is $(1,2)$-computed by a transducer performing a *weak* $(1,2)$-computation with parameters (b_1, b_2). Whatever the difference $d = b_2 - b_1$, there exists a value of s such that both $s + b_1$ and $s + b_2$ are congruent to 1,2,5 or 7 (mod 9). Hence the transducer produces two wrong results on the pair $1^{s+b_1}, 1^{s+b_2}$ in contradiction with the $(1,2)$-computability. □

4 Nonconstructive methods for frequency transducers

Unfortunately, it is not clear how to generalize the explicit construction of the relation $R(x,y)$ in Theorem 5 to prove distinction between (m,n)-computability and (km,kn)-computability for weak finite-state frequency transducers. Luckily, there is a non-constructive method to do so. This method is based on usage of algorithmic information theory.

Definition 11. *We define a transformation I which takes words $x \in \{0,1\}^*$ into $I(x) \in \{0,1\}^*$ by the following rule. Every symbol* 0 *is replaced by* 1100110100 *and every symbol* 1 *is replaced by* 1011001010.

Definition 12. *We define a transformation J which takes words $x \in \{0,1\}^*$ into $J(x) \in \{0,1\}^*$ by the following rule. Every symbol* 0 *is replaced by* 0100110100 *and every symbol* 1 *is replaced by* 0011001010.

Lemma 1. *For arbitrary $x \in \{0,1\}^*$ the result of the transformation I is a word $I(x)$ such that $\mid I(x) \mid = 10 \mid x \mid$, and $I(x)$ contains equal number of zeros and ones.*

Lemma 2. *For arbitrary $x \in \{0,1\}^*$ the result of the transformation J is a word $J(x)$ such that $\mid J(x) \mid = 10 \mid x \mid$, and every subword y of $J(x)$ such that $\mid J(x) \mid = 10$ contains no more than 5 symbols* 1.

Definition 13. *We define a transformation K which takes words $x \in \{0,1\}^*$ into a 2-dimensional array*

$$K(x) = \begin{pmatrix} K_{11} & K_{12} & \cdots & K_{1n} \\ K_{21} & K_{22} & \cdots & K_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ K_{n1} & K_{n2} & \cdots & K_{nn} \end{pmatrix}$$

of size $10 \mid x \mid \times$ 10 $\mid x \mid$ by the following rule. The first row $\begin{pmatrix} K_{11} & K_{12} & \cdots & K_{1n} \end{pmatrix}$ copies $I(x)$. Every next row is a cyclic copy of the preceding one:

$$\begin{pmatrix} K_{s1} & K_{s2} & \cdots & K_{sn} \end{pmatrix} = \begin{pmatrix} K_{(s-1)2} & K_{(s-1)3} & \cdots & K_{(s-1)1} \end{pmatrix}$$

.

Definition 14. *We define a transformation L which takes words $x \in \{0,1\}^*$ into a 2-dimensional array*

$$L(x) = \begin{pmatrix} L_{11} & L_{12} & \cdots & L_{1n} \\ L_{21} & L_{22} & \cdots & L_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ L_{n1} & L_{n2} & \cdots & L_{nn} \end{pmatrix}$$

of size $10 \mid x \mid \times$ 10 $\mid x \mid$ by the following rule. The first row $\begin{pmatrix} L_{11} & L_{12} & \cdots & L_{1n} \end{pmatrix}$ copies $J(x)$. Every next row is a cyclic copy of the preceding one:

$$\begin{pmatrix} L_{s1} & L_{s2} & \cdots & L_{sn} \end{pmatrix} = \begin{pmatrix} L_{(s-1)2} & L_{(s-1)3} & \cdots & L_{(s-1)1} \end{pmatrix}$$

.

There is a dichotomy: 1) either there exist 4 rows (say, with numbers b_1, b_2, b_3, b_4) in $K(x)$ such that in every column Z such that among the values $K_{(b_1)z}, K_{(b_2)z}$, $K_{(b_3)z}, K_{(b_4)z}$ there are exactly 2 zeros and 2 ones, or 2) for arbitrary 4 rows (with numbers b_1, b_2, b_3, b_4 in $K(x)$) there is a column z such that among the values $K_{(b_1)z}, K_{(b_2)z}, K_{(b_3)z}, K_{(b_4)z}$ there are at least 3 values equal to 1. (Please remember that by Lemma 2 the total number of zeros and ones in every row is the same.) We will prove that if the Kolmogorov complexity of x is maximal and the length of x is sufficiently large then the possibility 1) does not exist.

Lemma 3. *If n is sufficiently large and x is a Kolmogorov-maximal word of length n then for arbitrary 4 rows (with numbers b_1, b_2, b_3, b_4 in $K(x)$) there is a column z such that among the values $K_{(b_1)z}, K_{(b_2)z}, K_{(b_3)z}, K_{(b_4)z}$ there are at least 3 values equal to 1.*

Proof. Assume from the contrary that there exist 4 rows (say, with numbers $b_1, b_2, b_3,\ b_4$) in $K(x)$ such that in every column z such that among the values $K_{(b_1)z}, K_{(b_2)z}, K_{(b_3)z}, K_{(b_4)z}$ there are exactly 2 zeros and 2 ones. By definition of K, $K_{(b_i)z} = K_{1(z+b_i-1)}$. Hence every assertion "among the values $K_{(b_1)z}, K_{(b_2)z}, K_{(b_3)z}, K_{(b_4)z}$ there are exactly 2 zeros and 2 ones" can be written as "among the values $K_{1(c_1)}, K_{1(c_2)}, K_{1(c_3)}, K_{1(c_4)}$ there are exactly 2 zeros and 2 ones" which is equivalent to "among the values $I(d_1), I(d_2), I(d_3), I(d_4)$ there are exactly 2 zeros and 2 ones".

Every value $I(d)$ was obtained from a single letter in the word x. Namely, the letters $I(10j+1), I(10j+2), \cdots, I(10j+10)$ were obtained from the j-th letter$x(j)$ of x. $I(10j+1) = 1$ both for $x(j)$ being a or b. $I(10j+2) = 1$ if $x(j)$ equals a but not b. $I(10j+3) = 1$ if $x(j)$ equals b but not a. Hence we introduce a functional

$$h(x(j)) = \begin{cases} 1 & \text{, if } d \equiv 0(mod 10) \\ \overline{x(j)} & \text{, if } d \equiv 1 \text{ or } 4 \text{ or } 5 \text{ or } 7 (mod 10) \\ x(j) & \text{, if } d \equiv 2 \text{ or } 3 \text{ or } 6 \text{ or } 8 (mod 10) \\ 0 & \text{, if } d \equiv 9 (mod 10) \end{cases}$$

Using this functional we transform every assertion of "among the values $I(d_1), I(d_2), I(d_3), I(d_4)$ there are exactly 2 zeros and 2 ones" type into a Boolean formula "among the values $h(x_{j_1}), h(x_{j_2}), h(x_{j_3}), h(x_{j_4})$ there are exactly 2 zeros and 2 ones".

Let a set S of such Boolean formulas be given. We say that another formula F is independent from the set S if F cannot be proved using formulas from the set S. For instance, if F contains a variable not present in any formula of S then F is independent from S.

Take a large integer n and consider the set T of all binary words from $\{a, b\}^{2n}$. There are 2^{2n} words in T. Let T_1 be the subset of T containing all words with equal number of a's and b's. Cardinality of T_1 equals $2^{2n-o(n)}$. The set S contains $2n$ formulas but not all of them are independent. However, since each formula contains only 4 variables, there are at least $\frac{2n}{4}$ independent formulas in S. Apply one-by-one these independent formulas and removes from T_1 all the words where some formula fails. Notice that application of a new formula independent from the preceding ones remove at least half of the words. Hence after all removals no more than $2^{\frac{3n}{2}-o(n)}$ words remain. Effective enumeration of all the remaining words and usage of Kolmogorov numbering as in Section 4 gives a method to compress each x to a length not exceeding $\frac{3n}{2} - o(n)$. This contradicts non-compressibility of Kolmogorov-maximal words. □

Since independence of formulas in our argument was based only on the used variables the same argument proves the following lemma.

Lemma 4. *If n is sufficiently large and x is a Kolmogorov-maximal word of length n then for arbitrary 4 rows (with numbers b_1, b_2, b_3, b_4 in $L(x)$) there is a column z such that among the values $L_{(b_1)z}, L_{(b_2)z}, L_{(b_3)z}, L_{(b_4)z}$ there are at least 3 values equal to 1.*

We are ready to prove the main theorem of this paper.

Theorem 6. *There exists a left-total relation R $(3,6)$-computed by a weak finite-state frequency transducer and not computed by any finite-state $(2,4)$-frequency transducer.*

Proof. Consider the relation

$$R(x,y) = \begin{cases} \text{true, if } y = 1^{|x|} & \text{and } \mid x \mid \equiv j (modn) \text{ and } L_{1j} = 0 \\ \text{true, if } y = 0 & \text{and } \mid x \mid \equiv j (modn) \text{ and } L_{1j} = 1 \\ \text{false, if} & \text{otherwise.} \end{cases}$$

where $L(x)$ is as described above. □

References

1. Holger Austinat, Volker Diekert, Ulrich Hertrampf, Holger Petersen. Regular frequency computations. *Theoretical Computer Science,* vol. 330 No. 1, pp. 15–20, 2005.
2. Richard Beigel, William I. Gasarch, Efim B. Kinber. Frequency computation and bounded queries. *Theoretical Computer Science,* vol. 163 No. 1/2, pp. 177–192, 1996.
3. John Case, Susanne Kaufmann, Efim B. Kinber, Martin Kummer. Learning recursive functions from approximations. *Journal of Computer and System Sciences,* vol. 55, No. 1, pp. 183–196, 1997.
4. A.N.Degtev. On (m,n)-computable sets. *Algebraic Systems,* Edited by D.I. Moldavanskij, Ivanovo Gos. Universitet, pp. 88–99, 1981.
5. Eitan Gurari. An Introduction to the Theory of Computation. *Computer Science Press, an imprint of E. H. Freeman,* Chapter 2.2, 1989.
6. Valentina Harizanova, Martin Kummer, Jim Owings. Frequency computations and the cardinality theorem. *The Journal of Symbolic Logic,* vol. 57, No. 2, pp. 682–687, 1992.
7. Maren Hinrichs and Gerd Wechsung. Time bounded frequency computations. *Information and Computation,* vol. 139, pp. 234-257, 1997.
8. Richard M. Karp and Richard Lipton. Turing machines that take advice. *L' Enseignement Mathematique,* vol. 28, pp. 191–209, 1982.

9. Efim B. Kinber. Frequency calculations of general recursive predicates and frequency enumeration of sets. *Soviet Mathematics Doklady,* vol. 13, pp. 873–876, 1972.
10. Efim B. Kinber. Frequency computations in finite automata, *Kibernetika,* No. 2, pp. 7–15, 1976(Russian; English translation in Cybernetics 12 (1976) 179-187).
11. Martin Kummer. A proof of Beigel's Cardinality Conjecture. *The Journal of Symbolic Logic,* vol. 57, No. 2, pp. 677–681, 1992.
12. Robert McNaughton. The Theory of Automata, a Survey. *Advances in Computers,* vol. 2, pp. 379–421, 1961.
13. Gene F. Rose. An extended notion of computability. *Abstracts of International Congress for Logic, Methodology and Philosophy of Science,* p.14, 1960.
14. Boris A. Trakhtenbrot. On the frequency computation of functions. *Algebra i Logika,* vol. 2, pp.25–32, 1964 (Russian)