Available online at www.sciencedirect.com
SCIENCE DIRECT®
Theoretical Computer Science 330 (2005) 15–21
Theoretical Computer Science
www.elsevier.com/locate/tcs

Regular frequency computations☆

Holger Austinat, Volker Diekert*, Ulrich Hertrampf, Holger Petersen

FMI, Universität Stuttgart, Universitätsstr. 38, D-70569 Stuttgart, Germany

Abstract

An (m, n)-computation of a function f is given by a deterministic Turing machine which on n pairwise different inputs produces n output values where at least m of the n values are in accordance with f. In such a case, we say that the Turing machine computes f with frequency $\geqslant m/n$. The most prominent result for frequency computations is due to Trakhtenbrot: The class of (m, n)-computable functions equals the class of computable functions if and only if $2m > n$.

Via characteristic functions the definition of (m, n)-computability carries over to sets. Here Trakhtenbrot's result reads as: The class of (m, n)-computable sets equals the class of recursive sets if and only if $2m > n$.

The notion of frequency computation can be extended to other models of computation. For resource bounded computations, the behavior is completely different: for e.g., whenever $n' - m' > n - m$, it is known that under any reasonable resource bound there are sets (m', n')-computable, but not (m, n)-computable.

However, scaling down to finite automata, the analogue of Trakhtenbrot's result holds again: We show here that the class of languages (m, n)-recognizable by deterministic finite automata equals the class of regular languages if and only if $2m > n$. This was originally stated by Kinber, but his proof has a flaw, as pointed out by Tantau.

Conversely, for $2m \leqslant n$, the class of languages (m, n)-recognizable by deterministic finite automata is uncountable for a two-letter alphabet. When restricted to a one-letter alphabet, then every (m, n)-recognizable language is regular. This was also shown by Kinber. We give a new and more direct proof for this result.

Keywords: Frequency computation; Regular languages

☆ A preliminary version of this work appeared in [2].

* Corresponding author.
E-mail address: diekert@informatik.uni-stuttgart.de (V. Diekert).

0304-3975/$ - see front matter © 2004 Elsevier B.V. All rights reserved.
doi:10.1016/j.tcs.2004.09.007

1. Introduction

The notion of frequency computations was introduced in 1960 by Rose [10]: An (m,n)-computation of a function $f : \Sigma^* \to \mathbb{N}$ is given by a deterministic Turing machine M which on n pairwise different inputs produces n output values where at least m of the n values are in accordance with f. In such a case, we say that the Turing machine computes f with frequency $\geqslant m/n$ (This seems to suggest that the fraction m/n exactly corresponds to the power of such computations, but that is wrong, as follows from the fact that e.g. $(1,3)$-computations are provably less powerful than $(2,6)$-computations.)

Quite naturally, Myhill wondered whether f was recursive if m was close to n [9, p. 393]. This question was answered positively by Trakhtenbrot, who showed that (1) an (m,n)-computable function f is recursive if $2m > n$, and (2) for every pair (m,n), such that $2m \leqslant n$, there are uncountably many functions being (m,n)-computable, in particular, there are non-recursive functions of this type [12].

Later Dëgtev, Kummer and Stephan showed that for $2m \leqslant n$ and $2m' \leqslant n'$, the classes of (m,n)- and (m',n')-computable functions differ whenever $m \neq m'$ or $n \neq n'$ [4,8]. The exact inclusions, however, are still unknown (except for a few special cases).

The notion has also been extended to *resource bounded frequency computations*. For example, one may require that the Turing machine which performs the frequency computation works in polynomial time. In this case, the inclusion problem for frequency classes bears a one-to-one correspondence to so-called (m,n)-*admissible* sets, which can be handled by finite combinatorics [3, Theorem 7.3]. Hinrichs and Wechsung showed that $(m+1,n+1)P$ is a proper subset of $(m,n)P$ whenever $m < 2^{n-m}$ [6], where $(m,n)P$ denotes the class of all sets whose characteristic functions are (m,n)-computable in polynomial time. They also showed that for large enough m (at least doubly exponential) the classes $(m,n)P$ and $(m+1,n+1)P$ coincide. Their conjecture was the validity of that equality for all $m \geqslant 2^{n-m}$. While this conjecture is easily seen to hold for $n-m \leqslant 2$, in [5] it could also be shown for $n-m=3$. For larger values of $n-m$ it is still open.

Frequency computations in another setting have been studied by Kinber [7] and subsequently by Austinat et al. [1,2], who considered the case of deterministic finite automata. This leads to *regular frequency computations*. Here, again, formal languages are viewed as characteristic functions. In this framework, Trakhtenbrot's result for functions carries over to regular frequency computations: the class of (m,n)-recognizable languages equals the class of regular languages if and only if $2m > n$. If $2m \leqslant n$, then there are uncountably many (m,n)-recognizable subsets of Σ^* as soon as $|\Sigma| \geqslant 2$.

In [7], Kinber claimed an even stronger result about sets separable by finite automata (Theorem 3) from which the above result would follow as a corollary. However, Theorem 3 turned out to be wrong, as was shown by a counter-example given by Tantau [11].

An interesting special case concerns tally languages: when restricted to a one-letter alphabet, all (m,n)-recognizable languages are regular for $1 \leqslant m \leqslant n$. This was already proven in [7], but we give a new and more direct proof.

2. The classes (m,n)REG

The characteristic function of a formal language $L \subseteq \Sigma^*$ is denoted by $\chi_L : \Sigma^* \to \mathbb{B}$, where Σ is a finite alphabet and $\mathbb{B} = \{0,1\}$ is the set of Boolean values; it is defined as

$\chi_L(w) = 1$ if $w \in L$ and $\chi_L(w) = 0$ otherwise. We extend the notion of a deterministic finite automaton in the following way. Let $A = (Q, \Sigma, \$, \delta, q_0, \tau, n)$, where Q is a finite set of states with initial state q_0, the set Σ is a finite alphabet and \$ is a new symbol, $\$ \notin \Sigma$, the mapping $\delta : Q \times (\Sigma \cup \{\$\})^n \to Q$ is the transition function, the mapping $\tau : Q \to \mathbb{B}^n$ is the type of a state, and n is the number of components. The type of a state is used for the output.

We describe the behavior of such an automaton formally. For an input vector $u = (u_1, \ldots, u_n) \in (\Sigma^*)^n$, define $|u| = \max\{|u_i| \mid 1 \leqslant i \leqslant n\}$, and $q \cdot u = \delta(q, (u_1\$^{\ell_1}, \ldots, u_n\$^{\ell_n}))$, where $\delta : Q \times ((\Sigma \cup \{\$\})^n)^* \to Q$ is the natural extension of δ on n-tuples of words and $\ell_i = |u| - |u_i|$ for $1 \leqslant i \leqslant n$. The output of the automaton is then defined to be the type $\tau(q_0 \cdot u)$. Such an automaton is called an n-DFA.

A language $L \subseteq \Sigma^*$ (m, n)-recognized by an n-DFA A if and only if for each n-tuple $u = (u_1, \ldots, u_n) \in (\Sigma^*)^n$ of pairwise distinct words, the n-tuples $\tau(q_0 \cdot u)$ and $(\chi_L(u_1), \ldots, \chi_L(u_n))$ coincide on at least m components. A language L is called (m, n)-recognizable if and only if there exists an n-DFA A that (m, n)-recognizes L. The class of all (m, n)-recognizable languages is denoted by (m, n)REG. An example of a 2-DFA is shown in Fig. 2 at the end of this section (the boxed pairs are the types).

For the proof of our main result, we need the following lemma.

Lemma 1. *Let $2m > n$ and $L \subseteq \Sigma^*$ be a language in (m, n)REG. Let A be an n-DFA that (m, n)-recognizes L and let $x_1, \ldots, x_{n-1} \in \Sigma^*$ be words. Assume that L were not regular, then the following two assertions would hold*:

(1) *There are words $y_1, \ldots, y_n \in \Sigma^*$, satisfying $|y_i| > |x_j|$ (for all $1 \leqslant i \leqslant n, 1 \leqslant j \leqslant n-1$), such that the automaton A, given as input $(x_1, \ldots, x_{n-1}, y_i)$ $(1 \leqslant i \leqslant n)$, gives the same sequence of $n-1$ answers for $(x_1, \ldots, x_{n-1})$ and the wrong answer on each y_i.*

(2) *Let $y_1, \ldots, y_n \in \Sigma^*$ be any n words. For each input $(x_1, \ldots, x_{n-1}, y_i)$, $1 \leqslant i \leqslant n$, let b_i be the nth component of the output vector, and let $(b'_1, \ldots, b'_n) = \tau(q_0(y_1, \ldots, y_n))$* (see Fig. 1 for an illustration). *If $(b_1, \ldots, b_n)$ and $(b'_1, \ldots, b'_n)$ differ on at least m bits, then there is at least one i such that $b_i \neq \chi_L(y_i)$.*

Proof. For the first claim, let $|x| = \max\{|x_i| \mid 1 \leqslant i \leqslant n-1\}$. Now consider an enumeration $y'_1, y'_2, \ldots$ of $\{y \in \Sigma^* \mid |y| > |x|\}$, and look at the output of A on $(x_1, \ldots, x_{n-1}, y'_i)$. If only finitely many answers to the y'_i were wrong, then L would be regular: We could use the nth component of the output to define a regular language which would be a finite variation of L. Thus, there are infinitely many wrong answers for the last component. Since there are at most 2^{n-1} many answer sequences on $(x_1, \ldots, x_{n-1})$, at least one of these appears infinitely often. This suffices to show the first claim.

For the second claim, consider two vectors $(b_1, \ldots, b_n)$ and $(b'_1, \ldots, b'_n)$ as defined in the lemma, which differ on at least m components. Since $(b'_1, \ldots, b'_n)$ is produced by A, a machine which performs an (m, n)-computation of L, we know that at least m of the b'_i have to satisfy $b'_i = \chi_L(y_i)$. But $2m > n$, and thus less than m of the b'_i can possibly satisfy $b'_i \neq \chi_L(y_i)$. It follows immediately that if all the b_i were correct, at least m of them would coincide with the according b'_i, and consequently less than m would differ, contrary to our assumption. Thus, there is at least one incorrect b_i, i.e. one i with $b_i \neq \chi_L(y_i)$. □

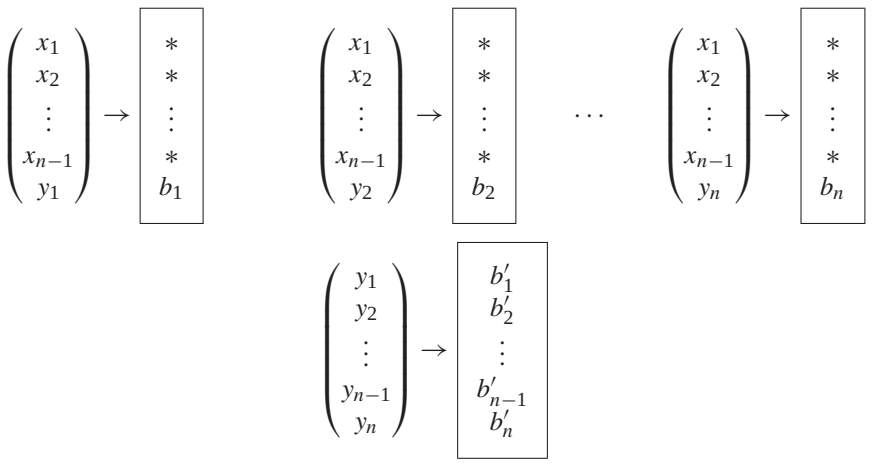

Fig. 1. Illustration of Lemma 1, Part 2. Input vectors are displayed in parentheses, state types are boxed; $*$ marks arbitrary output values.

Theorem 2. *Let $2m > n$ and $L \in (m, n)$*REG*. Then L is regular.*

Proof. We show that the inclusion (m, n)REG $\subseteq (m, n-1)$REG holds for $n > m > n/2$. Then, by induction we obtain (m, n)REG $\subseteq (m, m)$REG. The result follows since (m, m)REG obviously equals the class of regular languages for $m > 0$.

Let $L \in (m, n)$REG via some n-DFA A. If L is regular we are done. Otherwise, by making use of Lemma 1, we reduce the number of input strings by one while preserving the number of correct answers. In the first step of the construction however, we *increase* the number of inputs to $2n-1$. This intermediate automaton A' receives inputs $x_1, \ldots, x_{n-1}$ and $y_1, \ldots, y_n$. In parallel it simulates A on the n-tuples $(x_1, \ldots, x_{n-1}, y_i)$ for $1 \leqslant i \leqslant n$, and on $(y_1, \ldots, y_n)$. It enters a *distinguished state*, if and only if the following two conditions are satisfied.

(1) The output values that are generated by A on the initial $n-1$ components of each n-tuple $(x_1, \ldots, x_{n-1}, y_i)$ are the same for $1 \leqslant i \leqslant n$, i.e., if $(b_1, \ldots, b_n)$ is the output on $(x_1, \ldots, x_{n-1}, y_i)$ and $(b'_1, \ldots, b'_n)$ is the output on $(x_1, \ldots, x_{n-1}, y_k)$, then $b_j = b'_j$ for all j with $1 \leqslant j \leqslant n-1$, but possibly $b_n \neq b'_n$ for $i \neq k$.
(2) $(b_1, \ldots, b_n)$ and $(b'_1, \ldots, b'_n)$ differ on at least m bits, where as in Part 2 of Lemma 1, b_i is the nth component of the output on $(x_1, \ldots, x_{n-1}, y_i)$, and $(b'_1, \ldots, b'_n)$ the output vector for $(y_1, \ldots, y_n)$.

If A' enters a distinguished state, then we define its output. Its output on $x_1, \ldots, x_{n-1}$ is the same as that of A when receiving any $(x_1, \ldots, x_{n-1}, y_i)$, $1 \leqslant i \leqslant n$, and it is arbitrary on $y_1, \ldots, y_n$.

Suppose that A' enters a distinguished state. Then Part 2 of Lemma 1 applies, and the answer A gives for the last component of at least one tuple $(x_1, \ldots, x_{n-1}, y_i)$ is wrong. But

then A and hence A' generate at least m correct output values for the first $n-1$ components and the restriction to these components will be an $(m, n-1)$-recognition.

In the next step of the construction we eliminate the inputs $y_1, \ldots, y_n$ by enumerating all possible input symbols and keeping track of all subsets of states reachable in A', resulting in an automaton A''.

By Part 1 of Lemma 1 for every $x_1, \ldots, x_{n-1}$ there are inputs $y_1, \ldots, y_n$ that lead to a distinguished state. Therefore, based on the current state s of A'', one of the possible extensions of the partially read $y_1, \ldots, y_n$ leading to a distinguished state of A' is chosen and the first $n-1$ components of its output are assigned to s. □

The following proposition follows implicitly from the work of Trakhtenbrot [12]. The construction is rather simple.

Proposition 3. *Let $2m \leqslant n$. Then there are uncountably many languages in the class (m, n)*REG. *In particular, there is a language $L \in (m, n)$*REG *which is not regular* (*in fact, not recursively enumerable*).

Proof. We have to consider $m = 1$ and $n = 2$ only, since it is easy to see that $(1, 2)\text{REG} \subseteq (m, n)\text{REG}$ whenever $2m \leqslant n$. Let $\Sigma = \{0, 1\}$, and let $x \in \mathbb{R}$ be an arbitrary real number from the half open interval $[0, 1)$.

With every word w from Σ^* we associate a value $val(w)$, which is defined informally by $val(w) = 0.w$, or formally by

$$val(w) = \sum_{i=1}^{n} w_i \cdot 2^{-i},$$

where $w = w_1 \ldots w_n$, and $w_i \in \{0, 1\}$ for $i \in \{1, \ldots, n\}$.

We define $L_x := \{w \in \Sigma^* \mid val(w) < x\}$.

For fixed x the $(1, 2)$-automaton recognizing L_x on input (w_1, w_2) just has to determine, whether $val(w_1) < val(w_2)$. If this is the case, the type $(1, 0)$ is assumed, otherwise $(0, 1)$. The automaton is shown in Fig. 2.

We check correctness: If $val(w_1) < val(w_2)$, then the automaton could only be wrong, if $w_1 \notin L_x$, but $w_2 \in L_x$. But then $val(w_1) \geqslant x$ and $val(w_2) < x$, consequently $val(w_2) < val(w_1)$, a contradiction. The case $val(w_1) \geqslant val(w_2)$ is handled analogously.

The result follows since there are uncountably many $x \in [0, 1)$ (and thus languages L_x). □

Remark 4. The construction above was communicated to us by Nickelsen and Tantau. In [7] there is a slightly different construction for the result above: For every infinite word, the language of finite prefixes is $(1, 2)$-recognizable.

Corollary 5. *We have (m, n)*REG = REG *if and only if $2m > n$.*

3. The unary case

The proof above applies to all alphabets with at least two different letters. We will see here that it is in fact a necessary condition. The class of regular languages, which are defined

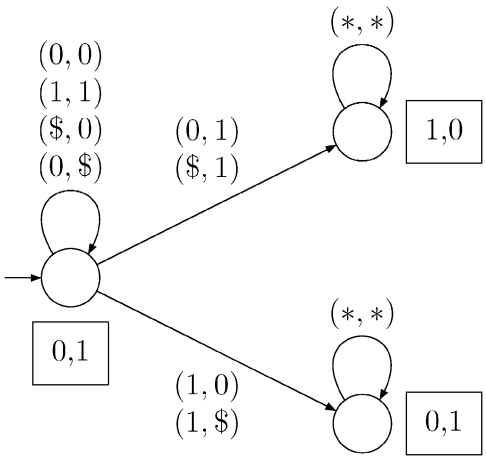

Fig. 2. A $(1, 2)$-automaton for L_x. Transitions are shown in parentheses, state types are boxed; \$ is the padding symbol; $(*, *)$ stands for an arbitrary tuple $\in \Gamma^2 \setminus \{\$\}^2$. Note, that the automaton is independent of the specific value of x.

over a one-letter alphabet, is called UREG. Analogously, we define (m, n)UREG as the class of languages in (m, n)REG which are defined over a one-letter alphabet.

Proposition 6. *For all $m, n \in \mathbb{N}$ such that $1 \leqslant m \leqslant n$, we have*

$$(m, n)\text{UREG} = \text{UREG}.$$

Proof. We prove for all n the inclusion $(1, n)\text{UREG} \subseteq \text{REG}$ by induction on n. The general claim follows, because $(m, n)\text{UREG} \subseteq (1, n)\text{UREG}$ for all $m > 0$.

The induction base is trivial, because $(1, 1)\text{UREG} = \text{UREG}$. So let $n > 1$, and let $(1, n-1)\text{UREG} \subseteq \text{REG}$.

For every language $L \in (1, n)\text{UREG}$, there is some n-DFA A witnessing this fact. Let Q be its finite set of states, $q_0 \in Q$ be its initial state, and let $\{a\}$ be the alphabet. Every word v to be considered has the form $a^{|v|}$, but as defined at the beginning of Section 2, the inputs seen by A will be tuples where the components are of the form $a^{|v|}\$^r$ for some $r \geqslant 0$. Define t and s such that for all $q \in Q$ we have $q \cdot (a^t, \ldots, a^t, \$^t) = q \cdot (a^{t+s}, \ldots, a^{t+s}, \$^{t+s})$; for example we may choose $t = |Q|$ and $s = |Q|!$.

Next we define the following function $g : \mathbb{N} \to Q \times \{0, \ldots, s-1\}$

$$g(x) = (q_0 \cdot (a^{x+s}, \ldots, a^{x+s}, a^x\$^s), \quad x \bmod s).$$

Now we distinguish two cases. The first case is

$$\forall\, x, y \in \mathbb{N} : g(x) = g(y) \Rightarrow (a^x \in L \Leftrightarrow a^y \in L).$$

In this case, every language $L_d := \{w \in L \mid |w| \bmod s = d\}$ $(0 \leqslant d \leqslant s-1)$ is regular, because we can on input a^x simulate A on input $(a^x, \ldots, a^x, a^x)$ and let all states be accepting or rejecting, depending on $g(x)$. But $L = \bigcup_{d=0}^{s-1} L_d$, and so $L \in \text{REG}$.

In the other case there are $x < y$ such that $g(x) = g(y)$, but $\chi_L(a^x) \neq \chi_L(a^y)$. Consider the language L' defined as

$$L' = \{v \in L \mid |v| \geqslant y + t\}.$$

It is enough to define a $(1, n-1)$-automaton A' for L'. Then by induction hypothesis L' is regular and hence, L is regular, because it is a finite variation of L'. Consider any input sequence $(a^{z_1}, \ldots, a^{z_{n-1}})$. As long as some $z_i < y + t$, the output of A' is defined to be $(0, \ldots, 0)$, and thus it gives at least one correct answer.

If $z_i \geqslant y + t$ for all i, then A' simulates A on input $(a^{z_1}, \ldots, a^{z_{n-1}}, a^x)$, and if A outputs $(b_1, \ldots, b_n)$, then the output of A' is defined as the first $(n-1)$ components $(b_1, \ldots, b_{n-1})$. We have to show that at least one of these answers is correct.

After reading $(a^{y+t}, \ldots, a^{y+t}, a^y\$^t)$ the automaton A is in the same state q_x as reading $(a^{x+t}, \ldots, a^{x+t}, a^x\$^t)$, because $g(x) = g(y)$. Then after reading $(a^{y+t}, \ldots, a^{y+t}, a^x\$^{t+(y-x)})$, the automaton is again in state q_x, because $x < y$ and $y - x$ is a multiple of s. Since $z_i \geqslant y + t$ for all i, we obtain that $q_0 \cdot (a^{z_1}, \ldots, a^{z_{n-1}}, a^y) = q_0 \cdot (a^{z_1}, \ldots, a^{z_{n-1}}, a^x)$, hence the same output is produced. But $a^x \in L$ and $a^y \notin L$ (or vice versa), thus, for exactly one of the two inputs given to A the last output bit is wrong, and consequently one of the first $(n-1)$ output bits has to be correct. □

References

[1] H. Austinat, V. Diekert, U. Hertrampf, A structural property of regular frequency computations, Theoret. Comput. Sci. 292 (1) (2003) 33–43.

[2] H. Austinat, V. Diekert, U. Hertrampf, H. Petersen, Regular frequency computations, in: Proc. RIMS Kokyuroku Symp. on Algebraic Systems, Formal Languages and Computations, Vol. 1166, Research Institute for Mathematical Sciences, Kyoto University, Japan, 2000, pp. 35–42.

[3] R. Beigel, M. Kummer, F. Stephan, Quantifying the amount of verboseness, Inform. Comput. 118 (1995) 73–90.

[4] A.N. Dëgtev, On (m, n)-computable sets, in: D.I. Moldavanskij (Ed.), Algebraic Systems, Ivanova Gos. University, 1981, pp. 88–99 (in Russian).

[5] U. Hertrampf, C. Minnameier, Resource bounded frequency computations with three errors, manuscript, 2004.

[6] M. Hinrichs, G. Wechsung, Time bounded frequency computations, Inform. Comput. 139 (2) (1997) 234–257.

[7] E.B. Kinber, Frequency computations in finite automata, Kibernetika 2 (1976) 7–15 (in Russian; Engl. transl. Cybernetics 12 (1976) 179–187).

[8] M. Kummer, F. Stephan, The power of frequency computation, in: H. Reichel (Ed.), in: Proc. 10th Internat. Congr. on Fundamentals of Computation Theory (FCT 1995), Lecture Notes in Computer Science, Vol. 969, 1995, pp. 323–332.

[9] R. McNaughton, The theory of automata, a survey, Adv. Comput. 2 (1961) 19–61/379–421.

[10] G.F. Rose, An extended notion of computability, in: Proc. Internat. Congr. for Logic, Methodology, and Philosophy of Science, Stanford, CA, 1960, p. 14.

[11] T. Tantau, Towards a cardinality theorem for finite automata, in: K. Diks, W. Rytter (Eds.), in: Proc. 27th Internat. Symp. on Mathematical Foundations of Computer Science (MFCS 2002), Warsaw, Poland, Lecture Notes in Computer Science, Vol. 2420, 2002, pp. 625–636.

[12] B.A. Trakhtenbrot, On the frequency computation of functions, Algebra Logika 2 (1963) 25–32 (in Russian).