数理解析研究所講究録
1166 巻 2000 年 35-42

Regular Frequency Computations

Holger Austinat, Volker Diekert, Ulrich Hertrampf, Holger Petersen

Universität Stuttgart
Institut für Informatik
Breitwiesenstr. 20–22

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 $\geq m/n$. 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 $2m > n$.
The notion of frequency computation can be extended to finite automata, and the analogue of Trakhtenbrot's result holds for formal languages: The class of languages (m,n)-recognizable by deterministic finite automata equals the class of regular languages if and only if $2m > n$. For $2m \leq n$, 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. We give new proofs for these results.

1 Introduction

The notion of frequency computations was introduced in 1960 by Rose [7]: 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 $\geq m/n$.

Quite naturally, Myhill wondered whether f was recursive if m was close to n [6, 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 $2m \leq n$ there are uncountably many functions being

(m,n)-computable, in particular, there are non-recursive functions of this type [8].

Later Degtev, Kummer and Stephan showed that for $2m \leq n$ and $2m' \leq n'$, the classes of (m,n)- and (m',n')-computable functions differ whenever $m \neq m'$ or $n \neq n'$ [2,5]. The exact inclusions, however, are still unknown (except for a few special cases).

This notion has also been extended to *time bounded frequency computations*. For example, one may require that the Turing machine which performs the frequency computations 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 [1, Theorem 7.3]. Hinrichs and Wechsung showed that $(m+1,n+1)\mathrm{P}$ is a proper subset of $(m,n)\mathrm{P}$ whenever $m < 2^{n-m}$ [3], where $(m,n)\mathrm{P}$ denotes the class of all sets whose characteristic functions are (m,n)-computable in polynomial time.

The notion of frequency computation has been extended by Kinber [4] to deterministic finite automata, which leads to *regular frequency computations*. 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 $2m \leq n$, then there are uncountably many subsets of Σ^* as soon as $|\Sigma| \geq 2$. However, when restricted to a one-letter alphabet, then all (m,n)-recognizable languages are regular for $1 \leq m \leq n$.

The results above are not new, but were first established in [4]. Here, however, we present new proofs for them. In fact, we were not aware of Kinber's result until the very end of the preparation of the present manuscript.

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 $|u| = \max\{|u_i| \mid 1 \leq i \leq n\}$, 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 $1 \leq i \leq n$. 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 $L \subseteq \Sigma^*$ is (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 first main result, we need the following lemma.

Lemma 1. *Let $2m > n$ and $L \subseteq \Sigma^*$. Let A be an n-DFA that (m,n)-recognizes L and $x_1, \ldots, x_{n-1} \in \Sigma^*$ be fixed. Then L is regular, or the following holds:*

1. *There are words $y_1, \ldots, y_n \in \Sigma^*$ such that the automaton A, given as input $(x_1, \ldots, x_{n-1}, y_i)$ $(1 \leq i \leq n)$, gives the same $n-1$ answers for $(x_1, \ldots, x_{n-1})$ and the wrong answer on each y_i. Moreover, we can assume $|y_i| > |x_j|$ for all $1 \leq i \leq n$, $1 \leq j \leq n-1$.*
2. *Let $y_1, \ldots, y_n \in \Sigma^*$ be any n words. For each input $(x_1, \ldots, x_{n-1}, y_i)$, $1 \leq i \leq n$, let b_i be the n-th component of the output vector, and let $(b'_1, \ldots, b'_n)$ be the output vector for $(y_1, \ldots, y_n)$. If $(b_1, \ldots, b_n)$ and $(b'_1, \ldots, b'_n)$ differ on at least m bits, there is at least one i such that $b_i \neq \chi_L(y_i)$.*

Proof. For the first claim, let $|x| = \max\{|x_i| \mid 1 \leq i \leq n-1\}$. Now consider an enumeration $y'_1, y'_2, \ldots$ of $\{y \in \Sigma^* \mid |y| > |u|\}$, 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 n-th component of the output to define a regular language which would be a finite variation of L. Thus, we can assume that there are infinitely many wrong answers for the last component. Since there are at most 2^{n-1} many answers on $(x_1, \ldots, x_{n-1})$, at least one of these answers appears infinitely often.

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. Assume all n of the b_i are correct, then at least m correct answers in $(b'_1, \ldots, b'_n)$

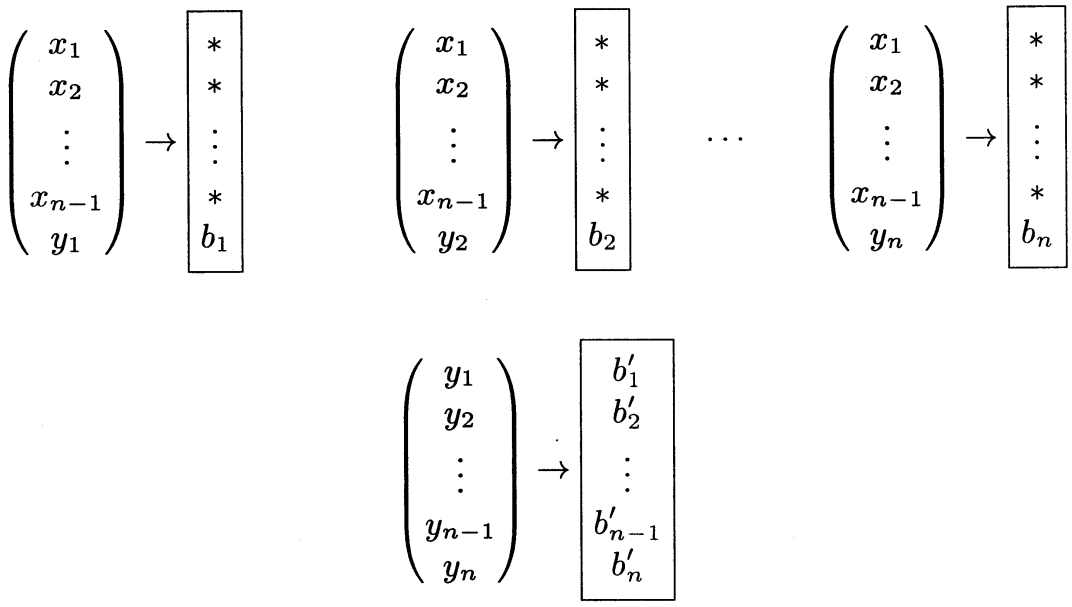

Fig. 1. Illustration of Lemma 1, Part 2. Input vectors are displayed in parentheses, state types are boxed; $*$ marks arbitrary output values.

coincide with the corresponding components in $(b_1, \ldots, b_n)$. (Cf. Fig. 1 for an illustration.) This implies that the number of differences is less than m. □

Proposition 1. *Let $2m > n$ and $L \in (m,n)$REG. Then L is regular.*

Proof. We show that the inclusion $(m,n)\text{REG} \subseteq (m,n-1)\text{REG}$ holds for $n > m > n/2$. Then, by induction we obtain $(m,n)\text{REG} \subseteq (m,m)\text{REG}$. The result follows since (m,m)REG is obviously 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 $1 \leq i \leq n$, 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 A generates on the initial $n-1$ components of each n-tuple $(x_1, \ldots, x_{n-1}, y_i)$ are the same for $1 \leq i \leq n$, 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 $1 \leq j \leq n-1$, but possibly $b_n \neq b'_n$ for $i \neq k$.

2. As in Part 2 of Lemma 1, let b_i be the n-th component of the output on $(x_1, \ldots, x_{n-1}, y_i)$, and let $(b'_1, \ldots, b'_n)$ be the output vector for $(y_1, \ldots, y_n)$. Then $(b_1, \ldots, b_n)$ and $(b'_1, \ldots, b'_n)$ differ on at least m bits.

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 $(x_1, \ldots, x_{n-1}, y_i)$, $1 \leq i \leq n$, 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 [8].

Proposition 2. *Let $2m \leq n$. 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. The construction is surprisingly simple. 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 $2m \leq n$. Let $\Sigma = \{0,1\}$, $x \in \mathbb{R}$ be an arbitrary real number from the half open interval $[0,1)$, and $bin(x)$ be the infinite binary expansion of x (padded with zeros if necessary). Then we define $L_x := \{w \in \Sigma^* \mid 0.w < bin(x)\}$.

For each such x the $(1,2)$-automaton recognizing L_x just has to determine the smaller of the two inputs, say x_1, which is assigned output 1, whereas the greater number x_2 is assigned 0. (If both inputs represent the

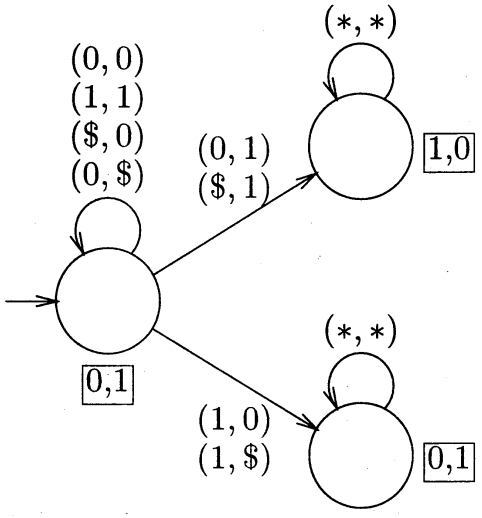

Fig. 2. An $(1,2)$-automaton for L_x. Transitions are shown in parentheses, state types are boxed; \$ is the padding symbol; $(*,*)$ stands for an arbitrary tuple $\in \Gamma^2 \setminus \{\$\}^2$.

same number but differ in trailing zeros, it is safe to output $(0,1)$, as well as it would be to output $(1,0)$.) The automaton is shown in Fig. 2.

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

Remark 1. The construction in the proof above was communicated to us by Arfst Nickelsen and Till Tantau. Kinber [4] gave a slightly different construction for the result above which is at least as nice: For every infinite word, the language of finite prefixes is $(1,2)$-recognizable.

□

Corollary 1. *We have* $(m,n)\mathrm{REG} = \mathrm{REG}$ *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 over a one-letter alphabet, is called UREG. Analogously, we define $(m,n)\mathrm{UREG}$ as the class of languages in $(m,n)\mathrm{REG}$ which are defined over a one-letter alphabet.

Proposition 3. *For all $m, n \in \mathbb{N}$ such that $1 \leq m \leq n$, we have*

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

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

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

For every language $L \in (1,n)\mathrm{UREG}$, 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 $r \geq 0$. 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 $L_d := \{w \in L \mid |w| \bmod s = d\}$ $(0 \leq d \leq s-1)$ 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 $L \in \mathrm{REG}$.

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

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

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 $z_i \geq y + t$ 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 $z_i \geq y + t$ 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. Richard Beigel, Martin Kummer, and Frank Stephan. Quantifying the amount of verboseness. *Information and Computation*, 118:73–90, 1995.
2. A. N. Degtev. On (m, n)–computable sets. In *Algebraic Systems*, pages 88–99. Ivanova Gos. University, 1981. (In Russian).
3. Maren Hinrichs and Gerd Wechsung. Time bounded frequency computations. In *Proceedings of the Twelfth Annual IEEE Conference on Computational Complexity*, pages 185–192, Ulm, Germany, June 24–27 1997.
4. Efim B. Kinber. Frequency computations in finite automata. *Kibernetika*, 2:7–15, March–April 1976. (In Russian; English translation in *Cybernetics*, 12:179–187, 1976).
5. Martin Kummer and Frank Stephan. The power of frequency computation. In *Proceedings of the Tenth International Congress on Fundamentals of Computation Theory–FCT 1995*, LNCS 969, pages 323–332, 1995.
6. Robert McNaughton. The theory of automata, a survey. *Advances in Computers*, 2:379–421, 1961.
7. Gene F. Rose. An extended notion of computability. In *Proceedings of the International Congress for Logic, Methodology, and Philosophy of Science*, Stanford, California, 1960.
8. Boris A. Trakhtenbrot. On the frequency computation of functions. *Algebra i Logika*, 2:25–32, 1963. (In Russian).