Frequency prediction of functions *

Kaspars Balodis, Ilja Kucevalovs, Rūsiņš Freivalds

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

Abstract. Prediction of functions is one of processes considered in inductive inference. There is a "black box" with a given total function f in it. The result of the inductive inference machine $F(< f(0), f(1), \cdots, f(n) >)$ is expected to be $f(n+1)$. Deterministic and probabilistic prediction of functions has been widely studied. Frequency computation is a mechanism used to combine features of deterministic and probabilistic algorithms. Frequency computation has been used for several types of inductive inference, especially, for learning via queries. We study frequency prediction of functions and show that that there exists an interesting hierarchy of predictable classes of functions.

1 Introduction

Physicists are well aware that physical indeterminism is a complicated phenomenon and probabilistical models are merely reasonably good approximations of reality. The problem "What is randomness?" has always been interesting not only for philosophers and physicists but also for computer scientists. The term "nondeterministic algorithm" has been deliberately coined to differ from "indeterminism".

Probabilistic (randomized) algorithms is one of central notions in Theory of Computation. However, since long ago computer scientists have attempted to develop notions and technical implementations of these notions that would be similar to but not equal to randomization.

The notion of frequency computation was introduced by G. Rose [28] as an attempt to have an absolutely deterministic mechanism with properties similar to probabilistic algorithms. The definition was as follows. A function f: $w \to w$ is (m,n)-computable, where $1 \leq m \leq n$, iff there exists a recursive function R: $w^n \to w^n$ 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 [25] 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 [31] by showing that f is recursive whenever $2m > n$. On

* The research was supported by Grant No. 09.1570 from the Latvian Council of Science and by Project 2009/0216/1DP/1.1.1.2.0/09/IPIA/VIA/044 from the European Social Fund.

the other hand, B.A. Trakhtenbrot [31] proved that if $2m = n$ then nonrecursive functions can be (m, n)-computed. E.B. Kinber extended the research by considering frequency enumeration of sets [20]. 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. Frequency computation in polynomial time was discussed in full detail by M. Hinrichs and G. Wechsung [19].

For resource bounded computations, the behavior of frequency computability 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 $2m > n$. Conversely, for $2m > n$, the class of languages (m, n)-recognizable by deterministic finite automata [3] 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 E.B. Kinber.

Frequency computations became increasingly popular when relation between frequency computation and computation with a small number of queries was discovered [23, 18, 6, 7].

Inductive inference is a process to find an algorithm from sample computations. We restrict ourselves to the case when a total function is to be identified.The first paper in this area was [17], yet (sometimes indirectly) the research was influenced by the theory of experiments with finite automata [24].

In the *prediction of functions* we consider a functional F. We say that F predicts a total function $f : N \to N$ correctly if the result $F(< f(0), \cdots, f(n) >)$ always equals $f(n+1)$. A class U of functions is called predictable if there exists a functional F correctly predicting every function $f \in U$.

This definition of predictability of functions is rather weak because only rather non-interesting classes of functions are predictable. All functions in U are distinguishable using only the value $f(0)$. Hence a more complicated definition is used.

Definition 1. *We say that F predicts a total function $f : N \to N$ in the limit if the result $F(< f(0), \cdots, f(n) >)$ equals $f(n+1)$ for all but a finite number of the values of n. It is not even demanded that $F(< f(0), \cdots, f(n) >)$ is defined for all n. A class U of functions is called predictable if there exists a recursive functional F correctly predicting every function $f \in U$.*

Definition 2. *We say that F (m, n)-predicts a class U of total functions $f : N \to N$ if for arbitrary n-tuple of pairwise distinct functions $f_1, f_2, \cdots, f_n$ from the class U the frequency algorithm works on n inputs receiving $F(< f_j(0), \cdots, f_j(n) >)$ on the j-th input (n being the same on all the inputs) and producing (at different moments!) outputs "$f_j(n + 1) = r$ (the computation is infinitely long and for some j the result can never be produced). It is demanded that there are at least m pairwise distinct functions $g_1, g_2, \cdots, g_m$ such*

that $\{g_1, g_2, \cdots, g_m\} \subseteq \{f_1, f_2, \cdots, f_n\}$ *and for all* $i \in \{1, 2, \cdots, m\}$ *the correct result on the corresponding output is produced for all but a finite number of* n.

2 Results

Lemma 1. *If a total function* f *is deterministically predicted in the limit then* f *is recursive.*

Proof. By the definition of predictability, there exists an n_0 such that for all $n > n_0$ the result $F(< f(0), \cdots, f(n) >)$ equals $f(n+1)$. Whatever the values $< f(0), \cdots, f(n_0) >$, the recursivity of f is implied by the recursivity of F. □

Nonetheless there is specifics of the frequency computation.

Theorem 1. *There is a class* U *of total functions containing a non-recursive function* g *such that for arbitrary positive integer* n *there is an algorithm* $(n, n+1)$*-predicting the class* U.

Proof. Let f be a total non-recursive function such that $f(0) = 0$. Let U consist of the function f and all constants c. The frequency algorithm for each f_j predicts the next value as $f_j(0)$. For all the constant functions the prediction is correct. Since the functions are supposed to be distinct, no more than one of the functions is predicted incorrectly. □

How (m, n)-predictability and (m', n')-predictability is related? Some implications are evident.

Lemma 2. *If* U *is* (m, n)*-predictable, then* U *is also* $(m, n+1)$*-predictable.*

Lemma 3. *If* U *is* $(m+1, n+1)$*-predictable, then* U *is also* (m, n)*-predictable.*

Theorem 2. *For arbitrary positive integer* k *there is a class* U_k *of total functions such that:*
1) for arbitrary positive integer n *the class* U_k *is* $(n, n+k)$*-predictable,*
2) for no positive integer n *the class* U_k *is* $(n, n+k-1)$*-predictable.*

Proof. The class U_k consists of all the constants and k distinct non-recursive functions. If at any tuple of distinct target functions at least k errors are allowed, then the frequency algorithm can predict the functions as they were constants. If less than k errors are allowed then existence of a frequency predicting algorithm is supposed to predict in the limit at least one non-recursive function. This contradicts an easy modification of Lemma 1. □

The proofs of Theorems 1 and 2 used essentially the property of the class U to contain non-recursive functions. This raises a natural question: does $(n, n+k)$-predictability depend on the parameters (n, k) if all the functions in U are recursive?

Theorem 3. *For arbitrary positive integer k there is a class U_k of total recursive functions such that:*
1) for arbitrary positive integer n the class U_k is $(n, n+k)$-predictable,
2) for no positive integer n the class U_k is $(n, n+k-1)$-predictable.

The main idea of the proof is to construct U_k as a set of total recursive functions f_{ab} where $a \in N$ and $b \in \{1, 2, \cdots, k\}$. Each $f_{ab}(0)$ contains information about the value of a involved and complete information about the programs for all the functions f_{0b}, f_{1b}, $\cdots$, $f_{(a-1)b}$ but not the information about the programs for any of the functions f_{ab}. This way, if a is the largest first index of the target functions of the frequency algorithm, there is no need for an error on all the target functions with exception of f_{a1}, f_{a2}, $\cdots$, f_{ak}. On the other hand, the functions f_{a1}, f_{a2}, $\cdots$, f_{ak} are constructed to ensure that the frequency algorithm F computed by Turing machine φ_a working on these k functions cannot predict correctly all but a finite number of values $f_{aj}(n)$ for at least one of the functions f_{aj}. (In our paper φ is a Gödel numbering of all one argument partial recursive functions such that φ_0 is the nowhere defined function. For instance, any standard numbering of Turing machines can be used for this purpose.)

Formally, we use Smullyan's double recursion theorem [30]:

Smullyan's double recursion theorem. [27, 30] For any recursive functions g and h, there exist m and n such that

$$\varphi_m = \varphi_{g(<m,n>)}, \text{ and } \varphi_n = \varphi_{h(<m,n>)}.$$

This theorem can easily be generalized:

Lemma 4. *[30] For any s-tuple of total recursive functions $(h_1, h_2, \cdots, h_s)$ there exists an s-tuple of natural numbers $< y_1, y_2, \cdots, y_s >$ such that*

$$\varphi_{y_1} = \varphi_{h_1(<y_1,y_2,\cdots,y_s>)}, \cdots, \varphi_{y_s} = \varphi_{h_s(<y_1,y_2,\cdots,y_s>)}.$$

Proof of Theorem 3. We define $f_{ab} \in U_k$ by induction. There cannot be a universal 3-argument recursive function $U(a, b, x) = f_{ab}(x)$ because otherwise U would be deterministically predictable in the limit. We define $f_{a1}, \cdots, f_{ak}$ only after $f_{a'b}$ have been defined for all $a' < a$. Lemma 4 will be used to prove that φ_a cannot be a frequency algorithm predicting the k-tuple of functions $f_{a1}, \cdots, f_{ak}$.

Basis. We define $f_{01}, \cdots, f_{0k}$ as constant functions equal to zero. We define every number among $n_{01}, \cdots, n_{0k}$ as an integer v such that φ_v is constant zero (every number n_{ab} will be a correct φ-program for the function f_{ab}). We define every number among $z_{01}, \cdots, z_{0k}$ as 0 (every number z_{ab} will be a correct information about all φ-programs for all the functions $f_{a'b'}$, where $a' < a$ and $b' \in \{1, 2, \cdots, k\}$).

Since we define the functions $h_1, h_2, \cdots, h_k$ by a common procedure, we use infinite injury priority method to establish temporal priorities among the numbers $\{1, 2, \cdots, k\}$. These priorities are needed to describe the construction of the functions. We start with the "natural" priority $(1, 2, \cdots, k)$.

Inductive step. Assume that all the functions $f_{a'b}$ where $a' < a$ have already been defined . Let $t_1, t_2, \cdots, t_k$ be arbitrary natural numbers. For each $b \in \{1, 2, \cdots, k\}$ we construct a k-tuple of functions as follows. For all $b \in \{1, 2, \cdots, k\}$ we define the functions $\varphi_{h_b(<t_1,t_2,\cdots,t_s>)}$ stepwise. First, we define

$$\varphi_{h_b(<t_1,t_2,\cdots,t_s>)}(0) =< f_{a-1}(0), n_{a-1} > .$$

This value does not depend on b.

Assume, by a new induction, that each of the functions $\varphi_{h_b(<t_1,t_2,\cdots,t_s>)}$ is already defined on some $(0, 1, \cdots, d_b)$ and the priority among the numbers $\{1, 2, \cdots, k\}$ is now $\{w_1, w_2, \cdots, w_k\}$. For each $j \in \{1, 2, \cdots, k\}$, we define $b(j)$ as the value p such that $w_p = j$. Assume, by induction, that priority is coordinated with the number of values of the arguments where the functions are defined, i.e., $d_{b(1)} \leq d_{b(2)} \leq \cdots \leq d_{b(k)}$.

In a way to serialize the parallel processing of computing predictions by the frequency algorithm φ_a the k-tuple of functions to be constructed (where the values of the functions already constructed are taken as they are but the new values of the target functions are taken equal to zero), we compute (in this order) $\cdots$ q steps of φ_a on all the functions up to the length $d_{b(1)}$, then q steps of φ_a on all the functions up to the length $d_{b(2)}$, $\cdots$, then q steps of φ_a on all the functions up to the length $d_{b(k)}$, then $q + 1$ steps of φ_a on all the functions up to the length $d_{b(1)}$, then $q + 1$ steps of φ_a on all the functions up to the length $d_{b(2)}$, $\cdots$, then $q + 1$ steps of φ_a on all the functions up to the length $d_{b(k)}$, $\cdots$, then $q+1$ steps of φ_a on all the functions up to the length $d_{b(1)} + 1$, then $q+1$ steps of φ_a on all the functions up to the length $d_{b(2)} + 1$, $\cdots$, then $q + 1$ steps of φ_a on all the functions up to the length $d_{b(k)} + 1$, then $q+2$ steps of φ_a on all the functions up to the length $d_{b(1)}$, then $q + 2$ steps of φ_a on all the functions up to the length $d_{b(2)}$, $\cdots$, then $q + 2$ steps of φ_a on all the functions up to the length $d_{b(k)}$, then then $q + 3$ steps of φ_a on all the functions up to the length $d_{b(1)+1}$, then $q + 3$ steps of φ_a on all the functions up to the length $d_{b(2)+1}$, $\cdots$, then $q + 3$ steps of φ_a on all the functions up to the length $d_{b(k)+1}$, $\cdots$, then then $q + 4$ steps of φ_a on all the functions up to the length $d_{b(1)+2}$, then $q + 4$ steps of φ_a on all the functions up to the length $d_{b(2)+2}$, $\cdots$, then $q + 4$ steps of φ_a on all the functions up to the length $d_{b(k)+2}$, $\cdots$ till the first new prediction on one of the functions is found. Say, the prediction for $f_j(m + 1) = e$ is found. Then we define $\varphi_{h_j(<t_1,t_2,\cdots,t_s>)}(d_b + 1) = \varphi_{h_j(<t_1,t_2,\cdots,t_s>)}(d_b + 2) = \cdots = \varphi_{h_j(<t_1,t_2,\cdots,t_s>)}(m) = 0$ and $\varphi_{h_j(<t_1,t_2,\cdots,t_s>)}(m+1) = e+1$, (the prediction by φ_a is made wrong). Additionally, we extend the definition domains for all the functions whose priority comes after j, i.e., using the notation $w_p = j$, for all the functions $\varphi_{h_{w_p}(<t_1,t_2,\cdots,t_s>)}, \cdots, \varphi_{h_{w_k}(<t_1,t_2,\cdots,t_s>)}$.

By Lemma 4, there exists an s-tuple of natural numbers $< y_1, y_2, \cdots, y_s >$ such that

$$\varphi_{y_1} = \varphi_{h_1(<y_1,y_2,\cdots,y_s>)}, \cdots, \varphi_{y_s} = \varphi_{h_s(<y_1,y_2,\cdots,y_s>)}.$$

However, these functions may be not total. We define f_{ab} as φ_{y_b} if it is total, and as

$$f_{ab}(x) = \begin{cases} \varphi_{y_b}(x) & \text{, if } \varphi_{y_b} \text{ is defined on } [0,d] \text{ and } x \in [0,d], \\ 0 & \text{, if } \varphi_{y_b} \text{ is defined on } [0,d] \text{ and } x \geq d. \end{cases}$$

It is easy to see that f_{ab} either is equal to φ_{y_b} and the frequency algorithm φ_a makes infinitely many incorrect predictions on this function or φ_{y_b} is a function defined on a finite segment $[0,d]$ and f_{ab} is a total function extending φ_{y_b} but φ_a produces no predictions after the segment $[0,d]$.

Nonetheless, U_k is $(n, n+k)$-predictable for arbitrary natural n. Indeed, from the values $g_1(0), g_2(0), \cdots, g_{n+k}(0)$ the frequency algorithm can find the maximum value of a such that the target functions are $f_{ab} \in U_k$. The programs for all functions with $a' < a$ can be computed knowing $f_{ab}(0)$. No more than k distinct target functions can correspond to the maximum value of a. □

Theorem 4. *There are two classes U_1 and U_2 of total recursive functions such that:*
1) U_1 is deterministically predictable,
2) U_2 is deterministically predictable,
3) If $U_1 \cup U_2$ is (m,n)-predictable then $m = 0$.

Proof. Following example of [4], we define U_1 as the class of all total recursive functions f such that for all but a finite number of values of x it is true that $f(x) = 0$, and U_2 is the class of all total recursive functions f such that $\forall x(\varphi_{f(0)}(x) = f(x))$.

Now we prove that $U_1 \cup U_2$ is not (m,n)-predictable with $m > 0$. Assume from the contrary that it is (m,n)-predictable by a frequency algorithm φ_a. In order to use Lemma 4 we define an n-tuple of recursive functions $(h_1, h_2, \cdots, h_n)$. The functions take values $\varphi_{h_1(t_1,\cdots,t_n)}(0) = t_1, \cdots, \varphi_{h_n(t_1,\cdots,t_n)}(0) = t_n$,
$\varphi_{h_1(t_1,\cdots,t_n)}(1) = 1, \cdots, \varphi_{h_n(t_1,\cdots,t_n)}(1) = n$.

To define the values of these functions for $x > 1$ we compute predictions made by φ_a initial fragments of these functions of length 2, 3, $\cdots$ supposing that the subsequent values of the functions are zeros (but we do not add any new values to these functions) till for at least one of these predictions equal zero.

Copying the method used in the proof of Theorem 3 we get that all the constructed functions are either defined on a finite initial fragment of the sequence of natural numbers (and then the algorithm φ_a has produced only a finite number of predictions for this function) or the algorithm φ_a has produced infinitely many wrong predictions for this function. □

It was proved in [2] that Theorem 4 cannot be generalized to 3 classes U_1, U_2 and U_3. More precisely, it was proved in [2] that deterministic predictability of $U_1 \cup U_2, U_2 \cup U_3$ and $U_1 \cup U_3$ implies deterministic predictability of $U_1 \cup U_2 \cup U_3$.

Theorem 5. *Let n be a natural number such that $n \geq 7$, $m > \frac{n}{2}$ and $U_1, U_2, \cdots, U_n$ be classes of total recursive functions such that:*
1) $U_1 \cup U_2 \cup \cdots \cup U_{n-1}$ is (m,n)-predictable,
2) $U_1 \cup U_2 \cup \cdots \cup U_{n-2} \cup U_n$ is (m,n)-predictable,
3) $U_1 \cup U_2 \cup \cdots \cup U_{n-3} \cup U_{n-1} \cup U_n$ is (m,n)-predictable,
$\cdots$
n) $U_2 \cup U_3 \cup \cdots \cup U_n$ is (m,n)-predictable.
Then $U_1 \cup U_2 \cup \cdots \cup U_n$ is $(2m,2n)$-predictable.

Proof. We describe the processing by the $(2m,2n)$-algorithm the $2n$-tuple of functions $f_1, \cdots, f_{2n}$ from the class $U_1 \cup U_2 \cup \cdots \cup U_n$. The new frequency algorithm A has $2n$ distinct functions $f_1, \cdots, f_{2n}$ as the input. The old n algorithms $A_1, A_2, \cdots, A_n$ have only n functions as the input. The new algorithm A uses all possible $(2n)!/n!$ copies of each of algorithms $A_1, A_2, \cdots, A_n$ by choosing n functions out of $f_1, \cdots, f_{2n}$. To predict the next value of any function $f_i \in \{f_1, \cdots, f_n\}$ the algorithm A considers predictions of all old frequency algorithms on all n-tuples such that the tuple contains f_i. This restriction removes $\frac{n.(2n-1)!}{n!}$ for every old algorithm leaving $n(\frac{(2n)!}{n!} - \frac{n.(2n-1)!}{n!})$ predictions in total for the current value of f_i.

Our frequency algorithm A for $U_1 \cup U_2 \cup \cdots \cup U_n$ always considers the set of the pairs (algorithm A_j, n-tuple of inputs f_i) having made the least number of wrong number predictions for the function f_i under question. Since $n \geq 7$, and $m > \frac{n}{2}$, most of these pairs allow only a finite number of wrong predictions. When a pair has made a wrong prediction, this pair is removed from the set. When the first (!) prediction is made by one of the pair in the set, this prediction is output as the result of the new frequency algorithm for $U_1 \cup U_2 \cup \cdots \cup U_n$. Since $n \geq 7$ and $m > \frac{n}{2}$, at least $2m$ of the functions f_i get only finite number of wrong predictions. □

References

1. Farid M. Ablayev, Rūsiņš Freivalds. Why sometimes probabilistic algorithms can be more effective. *Lecture Notes in Computer Science,* vol. 233, pp. 1–14, 1986.
2. Kalvis Apsītis, Rūsiņš Freivalds, Mārtiņš Krikis, Raimonds Simanovskis, Juris Smotrovs. Unions of identifiable classes of total recursive functions. *Lecture Notes in Computer Science,* vol. 642, pp. 99–107, 1992.
3. Holger Austinat, Volker Diekert, Ulrich Hertrampf, Holger Petersen. Regular frequency computations. *Theoretical Computer Science,* vol. 330 No. 1, pp. 15–20, 2005.
4. Jānis Bārzdiņš (Y.M. Barzdin). Two theorems on limiting synthesis of functions. *Theory of algorithms and programs,* Riga, University of Latvia, vol. 1, pp. 82–88, 1974 (in Russian).
5. Jānis Bārzdiņš (Y.M. Barzdin), Rūsiņš Freivalds (R.V. Freivald). On the prediction of general recursive functions. *Soviet Mathematics Doklady,* vol. 13, pp. 1224–1228, 1972.
6. 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.

7. 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.
8. A.N.Degtev. On (m,n)-computable sets. *Algebraic Systems,* Edited by D.I. Moldavanskij, Ivanovo Gos. Universitet, pp. 88–99, 1981.
9. R.Freivalds. On the growth of the number of states in result of the determinization of probabilistic finite automata. *Avtomatika i Vichislitel'naya Tekhnika*, No. 3, pp. 39–42, 1982 (Russian)
10. Rūsiņš Freivalds, Marek Karpinski. Lower Space Bounds for Randomized Computation. *Lecture Notes in Computer Science,* vol. 820, pp. 580–592, 1994.
11. Rūsiņš Freivalds. Complexity of probabilistic versus deterministic automata. *Lecture Notes in Computer Science,* vol. 502, pp. 565–613, 1991.
12. Rūsiņš Freivalds. Inductive Inference of Recursive Functions: Qualitative Theory. *Lecture Notes in Computer Science,* vol. 502, pp. 77–110, 1991.
13. Rūsiņš Freivalds, Jānis Bārzdiņš, Kārlis Podnieks. Inductive Inference of Recursive Functions: Complexity Bounds. *Lecture Notes in Computer Science,* vol. 502, pp. 111–155, 1991.
14. Rūsiņš Freivalds. Models of computation, Riemann Hypothesis and classical mathematics. *Lecture Notes in Computer Science,* vol.1521, pp. 89–106, 1998.
15. Rūsiņš Freivalds. Non-constructive methods for finite probabilistic automata. *International Journal of Foundations of Computer Science,* vol. 19, No. 3, pp.565–580, 2008.
16. Rūsiņš Freivalds. Amount of nonconstructivity in finite automata. *Theoretical Computer Science,* vol. 411, No. 38-39, pp.3436–3443, 2010.
17. E.M. Gold. Language identification in the limit. *Information and Control,* vol. 10, No. 5, pp. 447–474, 1967.
18. 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.
19. Maren Hinrichs and Gerd Wechsung. Time bounded frequency computations. *Information and Computation,* vol. 139, pp. 234-257, 1997.
20. Efim B. Kinber. Frequency calculations of general recursive predicates and frequency enumeration of sets. *Soviet Mathematics Doklady,* vol. 13, pp. 873–876, 1972.
21. Efim B. Kinber. On frequency real-time computations. *Teoriya Algoritmov i Programm,* (Edited by Ya.M.Barzdin) vol. 2, pp. 174–182, 1973 (Russian)
22. Efim B. Kinber. Frequency computations in finite automata, *Kibernetika,* No. 2, pp. 7–15, 1976(Russian; English translation in Cybernetics 12 (1976) 179-187).
23. Martin Kummer. A proof of Beigel's Cardinality Conjecture. *The Journal of Symbolic Logic,* vol. 57, No. 2, pp. 677–681, 1992.
24. E.F. Moore. Gedanken-experiments on sequential machines. *Automata Studies (Ann. of Math. Studies, No. 34),* Princeton Univ. Press, Princeton, N.J., pp. 129–153, 1956.
25. Robert McNaughton. The Theory of Automata, a Survey. *Advances in Computers,* vol. 2, pp. 379–421, 1961.
26. Michael O. Rabin and Dana Scott. Finite automata and their decision problems. *IBM Journal of Research and Development,* vol. 3, No. 2, pp. 115?E25, 1959.
27. Hartley Rogers, Jr. *Theory of Recursive Functions and Effective Computability.* MIT Press, 1987.
28. Gene F. Rose. An extended notion of computability. *Abstracts of International Congress for Logic, Methodology and Philosophy of Science,* p.14, 1960.

29. Gene F. Rose and Joseph S. Ullian. Approximations of functions on the integers. *Pacific Journal of Mathematics,* vol.13 , No.2, pp.693–701, 1963.
30. Raymond M. Smullyan. *Theory of Formal Systems, Annals of Mathematics Studies,* No. 47, Princeton, N.J., 1961.
31. Boris A. Trakhtenbrot. On the frequency computation of functions. *Algebra i Logika,* vol. 2, pp.25–32, 1964 (Russian)