Frequency Pushdown Automata

Ilmārs Pužulis and Rūsiņš Freivalds*

Institute of Mathematics and Computer Science, University of Latvia,
Raiņa bulvāris 29, Riga, LV-1459, Latvia
Faculty of Computing, University of Latvia
Raiņa bulvāris 19, Riga, LV-1586, Latvia
ilmars.puzulis@hotmail.com

Abstract. Frequency computation was introduced in [16]. Trakhtenbrot [17] proved the existence of a continuum of functions computable by frequency Turing machines with frequency $\frac{1}{2}$. In contrast, every function computable by a frequency Turing machine with frequency exceeding $\frac{1}{2}$ is recursive. Essentially similar results for finite automata and other types of machines have been proved in [12] and [1]. We consider frequency pushdown automata. They are specific types of automata because allowing several pushdown stores would add too much computation power but allowing only one pushdown store restricts the computation power.

1 Introduction

During a discussion of the paper [8] at the conference *LATA 2013* in Bilbao, Spain, E. Shamir asked whether the results on frequency Turing machines and frequency finite automata hold for pushdown automata as well. The difficulty of the question is in the fact that an (n, n)-Turing machine or an (n, n)-finite automaton can be presented as a Cartesian product of n separate Turing machines or finite automata and this construction does not seem to increase the power of the machine. However, an arbitrary Turing machine can be simulated by an automaton with 3 pushdown tapes (and allowing some re-arrangement, even with 2 pushdown tapes) [2]. Hence the definition of frequency pushdown automata should avoid the use of several pushdown stores in a single automaton.

The notion of frequency computation was introduced in [16] as an attempt to have a deterministic notion of computation with properties similar to probabilistic algorithms. Let $\mathbb{N} = \{0, 1, 2, \ldots\}$ denote the set of all natural numbers, $\mathbb{N}_+ = \mathbb{N} \setminus \{0\}$. Fix $m, n \in \mathbb{N}, 1 \leq m \leq n$. The ith component of the m-tuple $(x_1, \cdots, x_m)$ is denoted by $(x_1, \ldots, x_m)_i$.

A function $f\colon \mathbb{N} \to \mathbb{N}$ is (m, n)-*computable* if there exists a recursive function $R\colon \mathbb{N}^n \to \mathbb{N}^n$ such that for all n-tuples $(x_1, \ldots, x_n) \in \mathbb{N}^n$ of mutually distinct natural numbers we have:

$$|\{|i \mid (R(x_1, \ldots, x_n))_i = f(x_i)\ ,\ 1 \leq i \leq n\} \geq m.$$

* Supported by the project 271/2012 from the Latvian Council of Science.Partially supported by Latvian State Research programme NexIT project No. 1.

Answering a problem by Myhill (see McNaughton [15]), Trakhtenbrot proved in [17] that: 1) if $2m > n$ then every (m, n)-computable function is recursive, and 2) if $2m = n$, then f can be not recursive. Kinber in [11, 12] extended these results by considering frequency enumeration of sets and proved that the class of (m, n)-computable sets equals the class of recursive sets if and only if $2m > n$.

The notion of frequency computation has been extended to other models of computation. Frequency computation in polynomial time was discussed in full detail in [10]. For resource bounded computations, the behavior of frequency computability is completely different. For example, under any reasonable resource bound, whenever $n' - m' > n - m$ there exist sets which are (m', n')-computable, but not (m, n)-computable. However, scaling down to finite automata, the analogue of Trakhtenbrot's result holds again: the class of languages (m, n)-recognizable by deterministic frequency 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 frequency automata is uncountable for a two-letter alphabet (cf. [1]).

When restricted to a one-letter alphabet, every (m, n)-recognizable language is regular (cf. [11] and [1]).

Frequency computations became increasingly popular when relations between frequency computation and computation with a small number of queries was discovered [1, 4, 5, 7, 9, 13, 14].

2 Frequency Pushdown Automata

Let Σ be any finite alphabet, and let Σ^* be the free monoid generated by Σ. The binary alphabet $\mathbb{B}$ is denoted by $\mathbb{B}$. Every subset $L \subseteq \Sigma^*$ is said to be a *language*. The elements of Σ^* are called *strings*; $|x|$ denotes the *length of a string* $x \in \Sigma^*$. By $\chi_L \colon \Sigma^* \to \{0, 1\}$ we denote the *characteristic function* of L.

A *deterministic pushdown automaton (PDA)* is a 7-tuple $M = (Q, \Sigma, \Gamma, \delta, q_0, Z, F)$ where Q is a finite set of states, Σ is a finite set which is called the input alphabet, Γ is a finite set which is called the stack alphabet, $q_0 \in Q$ is the start state, $Z \in \Gamma$ is the initial stack symbol, and $F \subseteq Q$ is the set of accepting states. An element $(p, a, A, q, \alpha) \in \delta$ is a transition of M. It has the intended meaning that M, in state $p \in Q$, with $a \in \Sigma \cup \{\varepsilon\}$ on the input and with $A \in \Gamma$ as topmost stack symbol, may read a, change the state to q, pop A, replacing it by pushing $\alpha \in \Gamma^*$. The $(\Sigma \cup \{\varepsilon\})$ component of the transition relation is used to formalize that the PDA can either read a letter from the input, or proceed leaving the input untouched.

For n-frequency pushdown automata we modify the above definition allowing n input words. However, we need to be aware that for the general case input words can be of distinct lengths. Our definition closely models the definition of n-frequency finite automata (see, e.g. [8]).

A *deterministic n-frequency automaton (n-DFA)* is a 7-tuple $\mathcal{A} = [Q, \Sigma, \#, \delta, q_0, \tau, n]$, where $n \in \mathbb{N}$, $n \geq 1$, Q is a finite set of states, q_0 is the

initial state, Σ is a finite alphabet and # is a symbol not in Σ. The mapping $\delta\colon Q \times (\Sigma \cup \{\#\})^n \to Q$ is the transition function; the function $\tau\colon Q \to \mathbb{B}^n$ is the type of state which is used for outputs. The type is interpreted as an n-tuple of answers α_i: its i-th component records whether the i-th input word read from the i-th input up to the current moment belongs to the language. We use the notation $\tau(q, (x_1\#^{\ell_1}, \ldots, x_n\#^{\ell_n}))$ to denote the type after reading the inputs the words $(x_1\#^{\ell_1}, \ldots, x_n\#^{\ell_n})$.

Next we formally describe the behavior of an n-DFA $\mathcal{A}$. Let $n \in \mathbb{N}_+$, and let $x = (x_1, \ldots, x_n) \in (\Sigma^*)^n$ be an input vector. We define $|x| = \max\{|x_i| \mid 1 \leq i \leq n\}$, and $q \circ x = \delta^*(q, (x_1\#^{\ell_1}, \ldots, x_n\#^{\ell_n}))$, where $\delta^*\colon Q \times ((\Sigma \cup \{\#\})^n)^* \to Q$ is the usual extension of δ on n-tuples of strings, and $\ell_i = |x| - |x_i|$ for all $1 \leq i \leq n$. The output of $\mathcal{A}$ is defined to be the type $\tau(q_0 \circ x)$.

A language $L \subseteq \Sigma^*$ is said to be (m, n)-*recognized* by an n-DFA $\mathcal{A}$ if for each n-tuple $(x_1, \ldots, x_n) \in (\Sigma^*)^n$ of pairwise distinct strings the tuples $\tau(q_0 \circ x)$ and $(\chi_L(x_1), \ldots, \chi_L(x_n))$ coincide on at least m components. A language $L \subseteq \Sigma^*$ is called (m, n)-*recognizable* if there is an n-DFA $\mathcal{A}$ that (m, n)-recognizes L.

To define deterministic n-frequency pushdown automata (with only one pushdown store) the transition function will be extended for n-tuples $\delta^*\colon Q \times \Sigma^n \times (\Gamma \cup \{\varepsilon\}) \to Q \times (\Gamma \cup \{\varepsilon\})$.

A *deterministic n-frequency pushdown automaton (n-DFPA)* is a 9-tuple $\mathcal{A} = (Q, \Sigma, \#, \Gamma, \delta, q_0, \tau, Z, F)$, where $\# \notin \Sigma$ and $(Q, \Sigma \cup \{\#\}, \Gamma, \delta, q_0, \tau, Z, F)$ is a PDA.

Let $n \in \mathbb{N}_+$, and let $x = (x_1, \ldots, x_n) \in (\Sigma^*)^n$ be an n-tuple. We define $|x| = \max\{|x_i| \mid 1 \leq i \leq n\}$, and

$$q \circ x = \delta^*(q, (x_1\#^{\ell_1}, \ldots, x_n\#^{\ell_n})),$$

where $\ell_i = |x| - |x_i|$ for all $1 \leq i \leq n$. Then the output of $\mathcal{A}$ is defined to be the type $\tau(q_0 \circ x)$. We emphasize that the n-DFPA contains only one pushdown tape which is used to process all n inputs.

A language $L \subseteq \Sigma^*$ is said to be (m, n)-*recognized* by an n-DFPA $\mathcal{A}$ if for each n-tuple $(x_1, \ldots, x_n) \in (\Sigma^*)^n$ of pairwise distinct strings the tuples $\tau(q_0 \circ x)$ and $(\chi_L(x_1), \ldots, \chi_L(x_n))$ coincide on at least m components. A language $L \subseteq \Sigma^*$ is called (m, n)-*recognizable* if there is an n-DFPA $\mathcal{A}$ that (m, n)-recognizes L.

3 Definitions

By $\mathbb{N} = \{0, 1, 2, \ldots\}$ we denote the set of nonnegative integers and $\mathbb{B} = \{0, 1\}$. $[n] = \{1, 2, \ldots, n\}$. We use $|X|$ to denote the cardinality of a set X.

Let $A \subseteq \mathbb{N}$ be a set. By $c_A : \mathbb{N} \to \mathbb{B}$ we denote the *characteristic function* of A:

$$c_A(x) = \begin{cases} 1, & \text{if } x \in A \\ 0, & \text{if } x \notin A \end{cases}$$

We say that a function is *recursive* if there is an algorithm (Turing machine) that computes the function. If c_A is a total recursive function then we call the set A *recursive*.

Definition 1. *A set A is (m,n)–computable iff there is a total recursive function f which assigns to all distinct inputs $x_1, x_2, \ldots, x_n$ a binary vector $(y_1, y_2, \ldots, y_n)$ such that at least m of the equations $c_A(x_1) = y_1, c_A(x_2) = y_2, \ldots, c_A(x_n) = y_n$ hold.*

By a *structure* of a finite set K we call a set of K's subsets $S \subseteq 2^K$.

We assume that the elements of K are ordered under some fixed ordering $\phi : K \to [n]$ where $n = |K|$.

Definition 2. *A set A is (S,K)–computable (or computable with a structure S) iff there is a total recursive function f which assigns to all distinct inputs $x_1, x_2, \ldots, x_n$ a binary vector $(y_1, y_2, \ldots, y_n)$ such that*
$\exists B \in S\ \forall b \in B\ \chi_A\left(x_{\phi(b)}\right) = y_{\phi(b)}$.

It can be seen that (m,n)–computing is a special case of (S,K)–computing by taking S to be the set of all subsets of K of size m.

4 Fano Plane

In finite geometry, the Fano plane (named after Gino Fano) is the finite projective plane of order 2, having the smallest possible number of points and lines. This plane has 7 points and 7 lines with 3 points on every line and 3 lines through every point. Every two points are on a unique line and every two lines intersect in a unique point.

We consider the first example of what we call *structured frequency algorithm.*

Definition 3. *A set A is* Fano-computable *iff there exists a recursive operator $R : N^7 \to \{0,1\}^7$ such that, for all 7-tuples $(x_0, x_1, \cdots, x_6) \in N^7$ of mutually distinct natural numbers,*

$$[(R(x_0) = c_A(x_0) \wedge R(x_1) = c_A(x_1) \wedge R(x_3) = c_A(x_3)) \vee$$

$$\vee (R(x_1) = c_A(x_1) \wedge R(x_2) = c_A(x_2) \wedge R(x_4) = c_A(x_4)) \vee$$

$$\vee (R(x_2) = c_A(x_2) \wedge R(x_3) = c_A(x_3) \wedge R(x_5) = c_A(x_5)) \vee$$

$$\vee (R(x_3) = c_A(x_3) \wedge R(x_4) = c_A(x_4) \wedge R(x_6) = c_A(x_6)) \vee$$

$$(R(x_4) = c_A(x_4) \wedge R(x_5) = c_A(x_5) \wedge R(x_0) = c_A(x_0)) \vee$$

$$\vee (R(x_5) = c_A(x_5) \wedge R(x_6) = c_A(x_6) \wedge R(x_1) = c_A(x_1)) \vee$$

$$\vee (R(x_6) = c_A(x_6) \wedge R(x_0) = c_A(x_0) \wedge R(x_2) = c_A(x_2))]$$

where $R(x_i)$ denotes the i-th component of $R(x_0, x_1, \cdots, x_6)$.

Theorem 1. *(K.Balodis,J.Iraids, R.Freivalds [3]) A set A is* Fano-computable *iff it is recursive.*

5 Results

Now consider the language $M = \{w2w^{rev} \mid w \in \mathbb{B}^*\}$ which is clearly $(1,1)$-recognizable by a 2-DFPA.

Theorem 2. *(C.Calude, R.Freivalds, F.Stephan [6])*
The language $M = \{w2w^{rev} \mid w \in \mathbb{B}^*\}$ is $(1,1)$-recognizable, but not $(2,2)$-recognizable by a 2-DFPA.

Theorem 2 can be strengthened as follows:

Theorem 3. *The language $M = \{w2w^{rev} \mid w \in \mathbb{B}^*\}$ is $(1,1)$-recognizable but for all n it is not (n,n)-recognizable by a 2-DFPA.*

Theorem 4. *If a set A is* Fano-computable *by a 7–DFPA, then it is computable by a 1–DFPA.*

References

1. Austinat, H., Diekert, V., Hertrampf, U., Petersen, H.: Regular frequency computations. In Theoretical Computer Science, vol. 330, No. 1, pp. 15–20 (2005)
2. Barzdin, J. M.: On a class of Turing machines (Minsky machines). In Algebra i Logika, vol. 1, No. 6, pp. 42–51 (1962)
3. Balodis, K., Iraids, J., Freivalds, R.: Structured Frequency Algorithms. Unpublished manuscript (2014)
4. Balodis, K., Kucevalovs, I., Freivalds, R.: Frequency Prediction of Functions. In Lecture Notes in Computer Science, vol. 7119, pp. 76–83 (2012)
5. Beigel, R., Gasarch, W.I., Kinber, E.B.: Frequency computation and bounded queries. In Theoretical Computer Science, vol. 163, No. 1/2, pp. 177–192 (1996)
6. Calude, C.S., Freivalds, R., Stephan, F.: Deterministic Frequency Pushdown Automata. Unpublished manuscript (2014)
7. Degtev, A.N.: On (m,n)-computable sets. In: Moldavanskij, D.I. (ed.) Algebraic Systems. Ivanovo Gos. Universitet, pp. 88–99 (1981)
8. Freivalds, R., Zeugmann, T., Pogosyan, G.R.: On the Size Complexity of Deterministic Frequency Automata. In Lecture Notes in Computer Science, vol. 7810, pp. 287–298 (2013)
9. Harizanova, V., Kummer, M., Owings, J.: Frequency computations and the cardinality theorem. In The Journal of Symbolic Logic, vol. 57, No. 2, pp. 682–687 (1992)
10. Hinrichs, M., Wechsung, G.: Time bounded frequency computations. In Information and Computation, vol. 139, pp. 234–257 (1997)
11. Kinber, E.B.: Frequency calculations of general recursive predicates and frequency enumeration of sets. In Soviet Mathematics Doklady, vol. 13, pp. 873–876 (1972)
12. Kinber, E.B.: Frequency computations in finite automata. Kibernetika (Russian), No. 2, pp. 7–15, 1976; English translation in Cybernetics 12, 179–187 (1976)
13. Kinber, E.B., Smith, C.H., Velauthapillai, M., Wiehagen, R.: On Learning Multiple Concepts in Parallel. In Journal of Computer and System Sciences, vol. 50, No. 1, pp. 41–52 (1995)

14. Kummer, M., Stephan, F.: Recursion Theoretic Properties of Frequency Computation and Bounded Queries. In Information and Computation, vol. 120, No. 1, pp. 59–77 (1995)
15. McNaughton, R.: The theory of automata, a survey. In Advances in Computers, vol. 2, pp. 379–421 (1961)
16. Rose, G.F.: An extended notion of computability. In Abstracts of International Congress for Logic, Methodology and Philosophy of Science, p.14 (1960)
17. Trakhtenbrot, B.A.: On the frequency computation of functions. In Algebra i Logika (Russian), vol. 2, pp. 25–32 (1964)