האוניברסיטה העברית בירושלים

THE HEBREW UNIVERSITY OF JERUSALEM

LEARNING CYCLE LENGTH THROUGH FINITE AUTOMATA

By

RON PERETZ

Discussion Paper # 546 **April 2010**

מרכז לחקר הרציונליות

CENTER FOR THE STUDY OF RATIONALITY

Feldman Building, Givat-Ram, 91904 Jerusalem, Israel
PHONE: [972]-2-6584135 FAX: [972]-2-6513681
E-MAIL: ratio@math.huji.ac.il
URL: http://www.ratio.huji.ac.il/

Learning Cycle Length through Finite Automata

Ron Peretz*

April 24, 2010

Abstract

We study the space-and-time automaton-complexity of the CYCLE-LENGTH problem. The input is a periodic stream of bits whose cycle length is bounded by a known number n. The output, a number between 1 and n, is the exact cycle length. We also study a related problem, CYCLE-DIVISOR. In the latter problem the output is a large number that divides the cycle length, that is, a number $k \gg 1$ that divides the cycle length, or (in case the cycle length is small) the cycle length itself. The complexity is measured in terms of the SPACE, the logarithm of the number of states in an automaton that solves the problem, and the TIME required to reach a terminal state. We analyze the worst input against a deterministic (pure) automaton, and against a probabilistic (mixed) automaton. In the probabilistic case we require that the probability of computing a correct output is arbitrarily close to one.

We establish the following results:

- CYCLE-DIVISOR can be solved in deterministic SPACE $o(n)$, and TIME $O(n)$.
- CYCLE-LENGTH cannot be solved in deterministic SPACE $\times$ TIME smaller than $\Omega(n^2)$.
- CYCLE-LENGTH can be solved in probabilistic SPACE $o(n)$, and TIME $O(n)$.
- CYCLE-LENGTH can be solved in deterministic SPACE $O(nL)$, and TIME $O(n/L)$, for any positive $L \leq 1$.

*Center for the Study of Rationality, Hebrew University. This work is part of the author's Ph.D. thesis. The author wishes to thank his advisor Prof. Abraham Neyman for his guidance and support.
Email: ronprtz@math.huji.ac.il.

1 Introduction

The problems in this paper arise in the study of repeated games with finite automata. Neyman [Neyman, 2008] studies repeated two-person zero-sum games where each player is restricted to strategies that can be implemented by finite automata whose size is commonly known. In particular he focuses on the case where one of the players is oblivious. An oblivious automaton with n states is equivalent to a periodic sequence whose cycle length is at most n. He shows that if player one is oblivious and the game is repeated long enough ($\gg n \log n$), the asymptotic value of the game is given by a function $v(\frac{\log n_2}{n_1})$; where n_i is the number of states in player i's automata.

Neyman constructs an automaton for player two. In the first stage the automaton probabilistically learns the (exact) cycle length in $\gg n \log n$ steps. In the second stage, given the cycle length, the automaton deterministically computes a "best reply" sequence in n steps. A close look at the construction shows that the exact cycle length is not necessary. It can be replaced by a number $k \gg 1$ that divides the cycle length. Theorem 1 improves Neyman's result by showing that the asymptotic value can be obtained with a pure strategy guaranteeing that the play enters a cycle within $\mathrm{O}(n)$ steps.

2 Results

A finite automaton is a tuple $\langle \Sigma, S, s_*, f, H, O, g \rangle$, where

- Σ is a finite set, the input alphabet;
- S is a finite set, the states;
- $s_* \in S$ is the initial state;
- $f : S \times \Sigma \to S$ is the transition function;
- $H \subset S$ is the set of terminal states;
- O is a (finite) set, the output domain;
- $g : H \to O$ is the output function.

Given a sequence of input letters $a_1, a_2, \ldots$, the *run* of the automaton is a sequence of states $s_1, s_2, \ldots$ defined recursively by

$$s_1 = s_*,$$
$$s_{t+1} = f(s_t, a_t).$$

We say that an automaton halts at time t given the input a, if t is the first time the run visits a terminal state. That is, $t =$

$\min\{t' : s_{t'} \in H\}$. In this case, we say that, given a, the automaton halts in t steps outputting $g(s_t)$.

Let Σ be a finite alphabet. The set of n-periodic sequences is denoted $\Sigma^{(n)} = \left\{(a_t) \in \Sigma^{\mathbb{N}} : \forall t \in \mathbb{N}\ a_t = a_{t+n}\right\}$. The set of periodic sequences whose cycle length is at most n is denoted $\Sigma^{(\leq n)} = \bigcup_{k=1}^{n} \Sigma^{(k)}$. The exact cycle length of a periodic sequence, a, denoted $\rho(a)$, is the smallest integer n such that a is n-periodic. Formally, for $a \in \bigcup_{k=1}^{\infty} \Sigma^{(k)}$, $\rho(a) = \min\left\{k : a \in \Sigma^{(k)}\right\} = \gcd\left\{k : a \in \Sigma^{(k)}\right\}$. For the rest of the paper Σ is the set $\{0, 1\}$, unless stated otherwise.

Our first theorem provides an upper bound for the complexity of the CYCLE-DIVISOR problem. This is the main result. Theorems 3 and 4 as well as the solution to Neyman's problem in repeated games involve a use of Theorem 1.

Theorem 1. *There exists a deterministic automaton with $2^{\mathrm{O}(\sqrt{n \log n})}$ states, such that for any input $a \in \Sigma^{(\leq n)}$, the automaton halts in $(2 + \mathrm{o}(1))n$ steps outputting a number k that divides $\rho(a)$, and if $k < \sqrt{n \log n}$, then $k = \rho(a)$.*

The next theorem shows that the CYCLE-LENGTH problem is strictly harder than the CYCLE-DIVISOR problem.

Theorem 2. *The deterministic TIME $\times$ SPACE complexity of CYCLE-LENGTH is $\Omega(n^2)$.*

Randomization, however, can speed up the solution.

Theorem 3. *There exists a probabilistic automaton with $2^{\mathrm{O}(\sqrt{n \log n})}$ states that finds the exact cycle length in $(4 + \mathrm{o}(1))n$ steps with probability greater than $1 - \frac{1}{n}$.*

Finally, we show that the lower bound provided by Theorem 2 is tight up to a constant factor.

Theorem 4. *For every $0 < L < 1$ there exists a deterministic automaton with $2^{\mathrm{O}(nL)}$ states that finds the exact cycle length in $\mathrm{O}(n/L)$ steps.*

3 Proofs

We begin with two simple observations.

Claim 5. *The number of elements in $\Sigma^{(\leq n)}$ is less than 2^{n+1}.*

Proof.

$$\left|\Sigma^{(\leq n)}\right| = \left|\bigcup_{k=1}^{n} \Sigma^{(k)}\right| \leq \sum_{k=1}^{n} \left|\Sigma^{(k)}\right| = \sum_{k=1}^{n} 2^k = 2^{n+1} - 2$$

□

Claim 6. *For any finite alphabet Σ, the map $a \mapsto (a_1, \ldots, a_{2n})$, from $\Sigma^{(\leq n)}$ to Σ^{2n}, is injective.*

Proof. Suppose a, b are in $\Sigma^{(\leq n)}$ and $(a_1, \ldots, a_{2n}) = (b_1, \ldots, b_{2n})$. If $a \notin \Sigma^{(k)}$, then there exists $1 \leq i \leq n$ such that $a_i \neq a_{i+k}$; so $b_i \neq b_{i+k}$; so $b \notin \Sigma^{(k)}$. Similarly, if $b \notin \Sigma^{(k)}$ then $a \notin \Sigma^{(k)}$; so $\rho(a) = \rho(b)$. Since $(a_1, \ldots, a_{\rho(a)}) = (b_1, \ldots, b_{\rho(b)})$, $a = b$. □

We are now ready to prove Theorem 1.

Proof of Theorem 1. Let $m = [\sqrt{n \log n}]$. For every input sequence $(a_t)_{t=1}^{\infty} \in \Sigma^{(\leq n)}$, we define a set of positive integers $T_0(a)$ by

$$T_0(a) = \left\{ t \geq 2m : (a_{t-2m+1}, \ldots, a_t) \notin \mathbf{pref}\, \Sigma^{(\leq m)} \right\},$$

where "$\mathbf{pref}\, X$" denotes the set of all finite prefixes of sequences in X. Let

$$t_0(a) = \begin{cases} \min T_0(a) & \text{if } T_0(a) \neq \emptyset, \\ \infty & \text{otherwise.} \end{cases}$$

Note that T_0 is a $\rho(a)$-periodic. Namely, $t \in T_0$ iff $t + \rho(a) \in T_0$, for every $t \geq 2m$. Since $\rho(a) \leq n$, we have either $t_0 = \infty$ or $t_0 < n + 2m$. Note, too, that t_0 is a stopping time. Namely, the question whether $t < t_0$ can be answered by looking at $a_1, \ldots, a_t$.

We describe an automaton $\langle \Sigma, S, s_*, f : S \times \Sigma \to S, H \subset S, \mathbb{N}, g : H \to \mathbb{N} \rangle$. The states are partitioned into two disjoint sets $S = S_1 \dot{\cup} S_2$. The states in S_1 are visited during time $t < t_0$ and the states in S_2 are visited during time $t \geq t_o$. Given an input sequence a, we shall first describe the state visited at any time t, s_t, as a function of $(a_1, \ldots, a_t)$, and then argue that s_t is, indeed, a function of s_{t-1} and a_t.

The set of states S_1 consists of all the possible histories that may occur before time t_0 and no later than time $2n$. Namely,

$$S_1 = \{(a_1, \ldots, a_t) : a \in \Sigma^{(\leq n)}, 0 \leq t \leq 2n, \\ \forall t' = 2m, \ldots, t \ (a_{t'-2m+1}, \ldots, a_{t'}) \in \mathbf{pref}\, \Sigma^{(\leq m)}\}.$$

The initial state, s_*, is the empty history. The terminal states in S_1, $S_1 \cap H$, are the histories of length $2n$ in S_1, $S_1 \cap \Sigma^{2n}$. For

$t < \min\{t_0, 2n+1\}$, $s_t = (a_1, \ldots, a_t)$. Obviously, s_t is a function of s_{t-1} and a_t. For $s \in S_1 \cap H$, $s = (a_1, \ldots, a_{2n}) \in \mathbf{pref}\, \Sigma^{(\leq n)}$. By Claim 6, $(a_1, \ldots, a_{2n})$ determines a unique sequence $a \in \Sigma^{(\leq n)}$. We define $g(s) = \rho(a)$, and (for completeness only) $f(s) = s$.

It remains to show that $|S_1| = 2^{\mathrm{O}(\sqrt{n \log n})}$. We shall prove the following inequality:

$$|S_1| \leq 2n 2^{2m} m^{2n/m} \tag{3.1}$$

To see this, consider an element $(a_1, \ldots, a_t) \in S_1$. Let $l = [t/m]$. Let $k_1, \ldots, k_{l-1} \leq m$ be integers such that $(a_{m(i-1)+1}, \ldots, a_{m(i+1)}) \in \mathbf{pref}\, \Sigma^{(k_i)}$. Note that k_i together with $(a_1, \ldots, a_{mi})$ determine $(a_1, \ldots, a_{m(i+1)})$; therefore the entire sequence $(a_1, \ldots, a_t)$ is determined by the following data:

- t,
- $a_1, \ldots, a_m$,
- $a_{lm+1}, \ldots, a_t$,
- $k_1, \ldots, k_{l-1}$.

Counting the number of possible values for each data item concludes (3.1).

The set of states visited during time $t \geq t_0$, S_2, is defined by

$$\begin{aligned} B =& \{(b_1, \ldots, b_l) \colon l = 1, \ldots 2n, \\ & b_t \in \Sigma, \text{ for every } 1 \leq t \leq l, \\ & \sum_{t=mi+1}^{m(i+1)} b_t \leq 1, \text{ for every } 0 \leq i < [l/m]\}, \\ S_2 =& \Sigma^{2m} \times \Sigma^{2m} \times B. \end{aligned}$$

A straightforward calculation shows that $|S_2| = 2^{\mathrm{O}(\sqrt{n \log n})}$.

Assume $t_0 < \infty$. Recall that this means that $t_0 < n + 2m$. We would like to describe s_t, for $t \geq t_0$, as a function of the input sequence $(a_t)_{t=1}^{\infty}$. Consider the following stationary coding of the input sequence:

$$b_t = \begin{cases} 1 & \text{if } (a_{t-2m+1}, \ldots, a_t) = (a_{t_0-2m+1}, \ldots, a_{t_0}), \\ 0 & \text{otherwise.} \end{cases}$$

Note that there are at least m "zeros" between any two "ones" in $(b_t)_{t=1}^{\infty}$;[1] therefore $(b_{2m}, \ldots, b_{2n+2m-1}) \in B$.

[1] A (finite) sequence can overlap with itself by a shift of l places iff it is in $\Sigma^{(l)}$. This brilliant argument was suggested by Prof. Benjamin Weiss.

For $t_0 \leq t < 2n + 2m$, s_t is defined by

$$\begin{aligned} s_t = \langle & (a_{t_0-2m+1}, \ldots, a_{t_0}), \\ & (a_{t-2m+1}, \ldots, a_t), \\ & (b_{2m}, \ldots, b_t) \rangle \end{aligned}$$

Such a definition allows the automaton to compute the next bit, b_{t+1}, in the transition from time t to $t+1$. Since s_t is a function of $(a_1, \ldots, a_t)$, the transition from time $t_0 - 1$ to t_0 is also well defined.

As a stationary coding of $(a_t)_{t=1}^{\infty}$, $\rho(b)$ divides $\rho(a)$. Since $\rho(b) \leq n$ the entire sequence $(b_t)_{t=1}^{\infty}$ can be deduced from $b_{2m}, \ldots, b_{2n+2m-1}$. At time $2n + 2m - 1$ the automaton outputs $\rho(b)$. As mentioned, the sparseness of $(b_t)_{t=1}^{\infty}$ guarantees that $\rho(b) > m$. □

The proof of Theorem 2 relies on a diagonal argument, called the "fooling set," commonly used in the theory of communication complexity. See, for example, [Kushilevitz and Nisan, 1997, p.10]. The proof reduces the well-studied string equality problem to the CYCLE-LENGTH problem. Consequently, the well-known SPACE × TIME lower bound for string equality applies here. For completeness, we present a self-contained proof.

Proof of Theorem 2. Assume by negation that there exists an automaton with 2^S states that solves CYCLE-LENGTH in T steps, and that $S \cdot T < \frac{1}{16}n^2$. Choose a prime number $n/4 \leq p \leq n/2$. Consider inputs of the form $x, x, \ldots$, where $x \in \Sigma^p$. Any such input yields a sequence of states of the automaton, $s_1, \ldots, s_{\frac{T}{p}}$, where s_j is the state of the automaton at time pj. By the pigeonhole principle, there must be two inputs $x \neq y$, that yield the same sequence of states $s_1, \ldots, s_{\frac{T}{p}}$; therefore the sequence $x, y, x, y, \ldots$ also yields $s_1, s_2, \ldots$. This is a contradiction since the cycle length of $x, x, \ldots$ differs from the cycle length of $x, y, x, y, \ldots$[2]. □

In the next proof we use the Rabin-Karp [Karp and Rabin, 1987] hash function. Although, any other hash function could be applied here, the Rabin-Karp hash function has the advantage that it can be computed incrementally. This fact simplifies the proof significantly.

Proof of Theorem 3. Let $a \in \Sigma^{(\leq n)}$. Let $m = [\sqrt{n \log n}]$. Apply Theorem 1 to find a number k that divides $\rho(a)$ and if $k < m$ then $k = \rho(a)$. This can be done with $2^{O(\sqrt{n \log n})}$ states in $2(n + m)$ steps. If $k < m$, output k. If $k \geq m$, let p be a random prime number,

[2]W.l.o.g., the output is given by the state at time T since we may assume that once the automaton visits a terminal state it stays there forever.

chosen uniformly from the set of prime numbers between 2 and n^3. Let $b_t = \sum_{l=1}^{k} a_{kt+l} 2^{k-l}$. Let c_t be the class of integers congruent to b_t modulo p. That is, $c_t = p\mathbb{Z} + b_t \in \mathbb{Z}/p\mathbb{Z}$.

The mapping[3] $(a_{kt+1}, \ldots, a_{k(t+1)}) \mapsto c_t$ can be computed incrementally according to the rule

$$c_t^i = 2c_t^{i-1} + a_{kt+i} \mod p,$$

where $c_t^0 = 0$ and $c_t = c_t^k$. Since c_t^i and c_t assume values in a set of at most n^3 elements and $[n/k] = \mathrm{O}(\sqrt{n/\log n})$, the sequence $c_1, \ldots, c_{2[n/k]}$ can be learned with $2^{\mathrm{O}(\sqrt{n \log n})}$ states in $2n$ steps.

The number b_t encodes $a_{kt+1}, \ldots, a_{k(t+1)}$; therefore $\rho(a) = k\rho(b)$. Obviously, $\rho(c)|\rho(b) \leq [n/k]$; therefore $\rho(c)$ can be deduced from $c_1, \ldots, c_{2[n/k]}$. The automaton outputs $k\rho(c)$.

In the event that $\forall t \forall s [b_t \neq b_s \rightarrow c_t \neq c_s]$, we also have $\rho(b)|\rho(c)$, and hence $\rho(c) = \rho(b)$, and $k\rho(c) = \rho(a)$. It remains to estimate the probability of this event. The prime numbers theorem[4] and the fact that any integer $x > 2$ has less than $\log(x)$ distinct prime divisors ensures that if $b_t \neq b_s$, then $\Pr(b_t = b_s \mod p) = \mathrm{O}(n^{-3}(\log n)^2)$. Since s and t range between 1 and $[n/k]$, $\Pr(\forall t \forall s [b_t \neq b_s \rightarrow c_t \neq c_s]) = 1 - \mathrm{O}(n^{-2} \log n)$. □

Proof of Theorem 4. Let $a \in \Sigma^{(\leq n)}$ be an input sequence. Let $m = [\sqrt{n \log n}]$. Apply Theorem 1 to find a number k that divides $\rho(a)$. If $k < m$ output k. Let us assume that $k \geq m$ and describe, for each possible value of k, an automaton whose initial state is the state where the automaton of Theorem 1 halts.

For a set $A = \{\alpha_1 < \cdots < \alpha_l\} \subset \{1, \ldots, k\}$, consider the Σ^l-valued sequence b^A, defined by $(b_t^A)_i = a_{kt+\alpha_i}$. Note that for every $A, B \subset \{1, \ldots, k\}$

$$\rho(b^{A \cup B}) = \mathrm{lcm}(\rho(b^A), \rho(b^B)), \text{ and}$$
$$k\rho(b^{\{1,\ldots,k\}}) = \rho(a).$$

Let $l = [kL]$. Choose $A_1, \ldots, A_{\lceil L^{-1} \rceil} \subset \{1, \ldots, k\}$, such that $\bigcup_i A_i = \{1, \ldots, k\}$ and $|A_i| = l$, for every i. Let $n' = k[n/k]$. In the first $2n'$ the automaton learns the sequence b^{A_1}. This can be done since $\rho(b^{A_1}) \leq n'/k$, and the number of states required is $2^{\mathrm{O}(nL)}$. Assume that at time $2n'i$ the automaton has learned $k_i = \mathrm{lcm}(\rho(A_1), \ldots, \rho(A_i))$. In the next $2n'$ steps it learns $b^{A_{i+1}}$ and computes $k_{i+1} = \mathrm{lcm}(k_i, \rho(b^{A_{i+1}}))$. In doing so, the automaton computes $k_{\lceil L^{-1} \rceil} = \rho(a)$ after $2n' \lceil L^{-1} \rceil$ steps. □

[3] This is the Rabin-Karp hash function. See [Karp and Rabin, 1987].

[4] We only use the fact that the number of primes up to n is $\Omega(n/\log n)$.

4 Acknowledgement

We acknowledge Prof. Benjamin Weiss for his significant contribution to the proof of Theorem 1. See footnote 1. The author would like to thank Prof. Michael O. Rabin for a fruitful conversation, and last but not least Prof. Abraham Neyman for many helpful comments.

References

[Karp and Rabin, 1987] Karp, R. M. and Rabin, M. O. (1987). Efficient randomized pattern-matching algorithms. *IBM Journal of Research and Development*, 31(2):249–260.

[Kushilevitz and Nisan, 1997] Kushilevitz, E. and Nisan, N. (1997). *Communication Complexity*. Cambridge University Press, New York.

[Neyman, 2008] Neyman, A. (2008). Learning effectiveness and memory size. Discussion Paper 476, Center for the Study of Rationality, Hebrew University, Jerusalem.