Experiments in Complexity of Probabilistic and Ultrametric Automata

Kristīne Cīpola, Andris Pakulis, 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
kristine.cipola@gmail.com

Abstract. We try to compare the complexity of deterministic, nondeterministic, probabilistic and ultrametric finite automata for the same language. We do not claim to have final upper and lower bounds. Rather these results can be considered as experiments to find advantages of one type of automata versus another type.

1 Introduction

Any deterministic finite automaton accepting the language

$$L_{2015} = \{1^n \mid n \neq 2015\}$$

has at least 2015 states. We started our research with a simple exercise: is there a nondeterministic finite automaton accepting the language and using much less states than 2015.

The automaton that comes to our mind starts its work with one nondeterministic choice. In the first case the automaton uses 5 states to accept the input word if its length is not a multiple of 5. In the second case the automaton uses 13 states to accept the input word if its length is not a multiple of 13. In the third case the automaton uses 31 states to accept the input word if its length is not a multiple of 31. Hence the automaton has 49 states. Unfortunately, this automaton accepts not the language L_{2015} but rather the language

$$M_{2015} = \{1^n \mid 2015 \text{ does not divide } n\}.$$

We add two new cycles. One of the cycles has 47 states but after the 46-th state there is another nondeterministic branching of the computation path. By Sylvester's theorem [5] this automaton accepts all input words whose length exceeds $45 \cdot 46 - 1 = 2070$ and the automaton has 49+47=96 states.

However, it is possible to construct another nondeterministic automaton for the same language with 28 states only. This automaton starts its work with

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

one nondeterministic choice. In the first case the automaton uses 2 states to accept the input word if its length is not a multiple of 2. In the second case the automaton uses 3 states to accept the input word if its length is not congruent to 2 modulo 3. In the third case the automaton uses 5 states to accept the input word if its length is not congruent to 0 modulo 5. In the fourth case the automaton uses 7 states to accept the input word if its length is not congruent to 6 modulo 7.In the third case the automaton uses 11 states to accept the input word if its length is not congruent to 11 modulo 11. Hence the automaton has 28+47= 75 states.

It much more difficult to prove that there is no smaller nondeterministic finite automaton accepting the language L_{2015}. We used a computerized exhaustive search. Probably, it is a difficult problem to establish precise number of states $s(N)$ for nondeterministic finite automata accepting the languages

$$L_N = \{1^n \mid n \neq N\}.$$

A more easy but still nontrivial problem is to establish asymptotical estimates for $s(N)$.

Theorem 1. *The number of states $s(N)$ for nondeterministic finite automata accepting the language L_N does not exceed $O(\frac{(\log N)^2}{\log\log N})$.*

Proof. Following the traditional notation in number theory textbooks (e.g. [2]) we denote the increasing sequence of all prime numbers by $p_l, p_2, p_3, \ldots (p_l = 2, p_2 = 3, p_3 = 5, \ldots)$ Chebyshev function $\vartheta(x)$ is the sum of natural logarithms of all prime numbers not exceeding x.

$$\vartheta(x) = \Sigma_{p \leq x} \log p \sim x.$$

Hence the product $F(t) = p_1 \cdot p_2 \cdot \ldots p_t$ is an exponent of $t \cdot \log t$ while the sum $S(t) = \Sigma_{r \leq t} p_r$ equals

$$S(t) = \frac{t^2}{2}(\log t + \log\log t - \frac{3}{2} + o(1)) = O(\frac{(\log F(t))^2}{\log\log F(t)}).$$

□

2 Probabilistic Automata

To construct an efficient probabilistic finite automaton for the language L_N we use distinct methods to process long and short input words. Of course, the automaton cannot predict whether the current input word will be long or short. If the input word is shorter than $N \times \frac{1}{\epsilon}$ then we need to find such a set of prime modulos that *most part of them* show that the length of the input word differs from N. If the input word is longer then we need to construct a randomized procedure rejecting all the words. To combine these (seemingly contradictory goals) we use an idea proposed by R. Freivalds [3].

Theorem 2. *(R. Freivalds [3]) For arbitrary $\epsilon > 0$, there is a randomized 1-head off-line Turing machine recognizing palindromes with probability $1 - \epsilon$ in time $O(n. \log n)$.*

The method of the proof of Theorem 2 is used to ensure that all input words strictly shorter than N are rejected. In parallel, after reading arbitrary symbol from the input the probabilistic automaton goes to a special rejecting state with a probability $\frac{\epsilon}{2\pi N}$. This ensures that if the length of the input word exceeds $N \times \frac{1}{\epsilon}$ then the input word is rejected with probability at least $1 - 2\epsilon$.

Theorem 3. *The number of states $s(N)$ for minimal probabilistic finite automata accepting the language L_N with a probability $1 - \epsilon$ does not exceed $O((\log N)^2 \log\log N)^2$.*

3 Ultrametric Automata

The notion of p-adic numbers widely used in mathematics but not so much in Computer Science. R. Freivalds [4] introduced a new type of automata and algorithms, called ultrametric automata and ultrametric algorithms where p-adic numbers are used to replace real numbers, called probabilities, as measures of indeterminism. More detailed description of this notion can be found in [1].

In mathematics, a stochastic matrix is a matrix used to describe the transitions of a Markov chain. A *right stochastic matrix* is a square matrix each of whose rows consists of nonnegative real numbers, with each row summing to 1. A *stochastic vector* is a vector whose elements consist of nonnegative real numbers which sum to 1. The *finite probabilistic automaton* is defined as an extension of a non-deterministic finite automaton $(Q, \Sigma, \delta, q_0, F)$, with the initial state q_0 replaced by a stochastic vector giving the probability of the automaton being in a given initial state, and with stochastic matrices corresponding to each symbol in the input alphabet describing the state transition probabilities. It is important to note that if A is the stochastic matrix corresponding to the input symbol a and B is the stochastic matrix corresponding to the input symbol b, then the product AB describes the state transition probabilities when the automaton reads the input word ab. Additionally, the probabilistic automaton has a threshold λ being a real number between 0 and 1. If the probabilistic automaton has only one *accepting state* then the input word x is said to be accepted if after reading x the probability of the accepting state has a probability exceeding λ. If there are several accepting states, the word x is said to be accepted the total of probabilities of the accepting states exceeds λ.

Ultrametric automata are defined exactly in the same way as probabilistic automata, only the parameters called *probabilities of transition from one state to another one* are real numbers between 0 and 1 in probabilistic automata, and they are p-adic numbers called *amplitudes* in the ultrametric automata. Formulas to calculate the amplitudes after one, two, three, $\cdots$ steps of computation are exactly the same as the formulas to calculate the probabilities in the probabilistic automata. Following the example of finite quantum automata, we demand that

the input word x is followed by a special end-marker. At the beginning of the work, the states of the automaton get *initial amplitudes* being p-adic numbers. When reading the current symbol of the input word, the automaton changes the amplitudes of all the states according to the transition matrix corresponding to this input symbol. When the automaton reads the end-marker, the *measurement* is performed, and the amplitudes of all the states are transformed into the p-norms of these amplitudes. The norms are rational numbers and it is possible to compare whether or not the norm exceeds the threshold λ. If total of the norms for all the accepting states of the automaton exceeds λ, we say that the automaton accepts the input word.

However, it is needed to note that if there is only one accepting state then the possible probabilities of acceptance are discrete values $0, p^1, p^{-1}, p^2, p^{-2}, p^3, \cdots$. Hence there is no natural counterpart of *isolated cut-point* or *bounded error* for ultrametric machines.

Theorem 4. *For arbitrary odd prime p the number of states $s(N)$ for minimal p-ultrametric finite automata accepting the language L_N with a probability $1-\epsilon$ does not exceed $O((\log N)^2 \log\log N)$.*

References

1. Ādamsons, V., Jēriņš, K., Krišlauks, R., Lapiņa, M., Pakulis, A. and Freivalds, R.: Advantages of ultrametric counter automata. In Proceedings of SOFSEM 2015, vol. 2 (to be published, 2015)
2. Bach, E. and Shallit, J.: Algorithmic Number Theory. MIT Press (1996)
3. Freivalds, R.: Fast computation by probabilistic Turing machines. In Teorija Algoritmov i Programm (Russian), v. 2, Latvian State University, Riga, pp. 201–205 (1975)
4. Freivalds, R.: Ultrametric finite automata and Turing machines. Lecture Notes in Computer Science, vol. 7907, 1–11 (2013)
5. Sylvester, J.J.: Question 7382. In Mathematical Questions, Educational Times, vol. 41, p. 21 (1884)