THE COMPUTATIONAL ADVANTAGE OF PROBABILISTIC AUTOMATA OVER DETERMINISTIC AUTOMATA IN HYPERBOLIC PLANE*

Alexander Tarvid†, Rūsiņš Freivalds

Faculty of Computer Science, University of Latvia
Raina blvd. 29, Riga, 1459, Latvia

(Received December 5, 2011)

In this paper, we address the question whether a probabilistic finite-state automaton (pfa) could recognize a language not recognizable by a deterministic finite-state automaton in polynomial time. We show that there is such a language in a hyperbolic plane and prove that a 5-way error-bounded pfa can recognize it in polynomial time of the number of nodes on the plane.

DOI:10.5506/APhysPolBSupp.5.145
PACS numbers: 02.50.Cw, 89.20.Ff, 89.70.Eg

1. Introduction

Freivalds [1] proved that a non-regular language $\{0^n1^n|n \in \mathbb{N}\}$ can be recognized by an error-bounded 2-way probabilistic finite automaton (2pfa). However, his algorithm worked in exponential expected time, and Greenberg and Weiss [2] proved that no 2pfa running in $\exp(o(n))$ expected time could recognize this language with probability of error bounded away from 1/2. Dwork and Stockmeyer [3] proved that there is no other non-regular language that can be recognized by a 2pfa in polynomial expected time: if a 2pfa M recognizes a non-regular language with error probability bounded below 1/2, there is a constant $b > 0$ such that, for infinitely many n, the expected running time of M must exceed $\exp(n^b)$. Kaņeps and Freivalds [4] showed that this bound cannot be improved by constructing for arbitrary $k \geq 2$ a specific non-regular language that can be recognized by 2pfa with probability $(1 - \varepsilon)$ in expected time $\exp(n^{1/k})$.

* Presented at the Summer Solstice 2011 International Conference on Discrete Models of Complex Systems, Turku, Finland, June 6–10, 2011.

† Corresponding e-mail address: atarvid@inbox.lv

The current paper continues to explore environments allowing $n$pfa to have a computational advantage over its deterministic colleague. Specifically, we construct a language in discrete hyperbolic plane (DHP) that is recognizable by an error-bounded 5-way probabilistic finite-state automaton in polynomial time, but not recognizable by a 5-way deterministic finite-state automaton.

2. Languages in discrete hyperbolic plane

Hyperbolic geometry was independently created by Lobachevsky and Bolyai in the 19^{th} century by assuming that the parallel postulate of Euclidean geometry does not hold. In other words, it is assumed that given any point P not on line l, there are at least two distinct lines passing through P that do not intersect l.

It was shown that hyperbolic space has unusual computational properties. For instance, Margenstern and Morita [5] showed that 3SAT[1] is solvable in polynomial time in hyperbolic plane. There is also a stronger result reported in [6]. Let C be a classical class of complexity and denote by C_{h} the same class where the computational device is replaced by cellular automata in hyperbolic plane. Then there are only two hyperbolic classes corresponding to seven classical classes: $\text{DLOG}_{\text{h}} = \text{NLOG}_{\text{h}} = \text{P}_{\text{h}} = \text{NP}_{\text{h}}$ and $\text{PSPACE}_{\text{h}} = \text{EXPTIME}_{\text{h}} = \text{NEXPTIME}_{\text{h}}$. One of the crucial properties of hyperbolic space that is used in solving exponential-time problems in polynomial time is that the circumference and the area of a circle is an exponential function of its radius.

In this paper, we are considering discrete hyperbolic planes (DHPs). A DHP starts from the root, which is also referred to as level 1. Each node can have one or two children on the next level, with branching occurring in all or none nodes of a level. For instance, on the DHP of Fig. 1, branching occurs in all nodes of levels 2 and 3, but all nodes of the next two levels have a single child each. As a result, the number of nodes on a level increases exponentially from the distance to the root. Nodes are connected to their left and right neighbours on their level (note that leftmost nodes have no left connection, while rightmost nodes have no right connection). A technical assumption that we will need later is that every node contains a symbol from a finite alphabet; moreover, by definition, all nodes of a given level contain the same symbol.

[1] The 3-satisfiability problem is a decision problem with the question "Given the Boolean formula in conjunctive normal form where each clause contains exactly three literals, is there some assignment of Boolean values to the variables that makes the formula true?" Here, a literal is either one of the variables or its negation. This problem is in class NP.

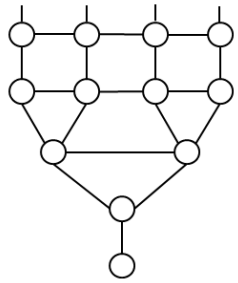

Fig. 1. Example of a discrete hyperbolic plane.

Just as in standard automata models, language defines a set of rules to be obeyed by the structure of an input string word, in our model, language defines a set of rules to be obeyed by the structure of an input DHP. A word of such language is a DHP itself. Such words differ, firstly, on the levels where branching occurs, and secondly, on symbols of the alphabet contained by nodes.

By construction, each node has not more than five connections to other nodes: parent, left and right neighbours, and left and right children. Thus, it is natural to use 5-way automata for moving on this structure, just as it was natural to use 2-way automata for moving on the standard tape. We assume that such an automaton can distinguish all five connections (implicitly, it is assumed that at every node, it can see whether or not branching occurs), knows from which of them it arrived to the current node, and can read a symbol from the node.

The particular language we are analysing in this paper is called SQUARE and is defined as follows. Given an arbitrary positive integer N, a DHP is in SQUARE if the following conditions hold:

- Branching occurs on levels N, $2N$, $3N$, ..., N^2 and no other levels;
- All nodes on levels $1, \ldots, N^2$ contain a '1', all nodes on level $\left(N^2+1\right)$ contain a '0', and the content of more distant levels' nodes is arbitrary.

An example of a word from SQUARE with $N = 3$ is shown in Fig. 2. Informally, SQUARE defines a DHP whose first N^2 levels consist of N blocks, each block consisting of N levels (hence the name SQUARE) and ending with a level with branching; 0's on the $\left(N^2+1\right)$-st level are used to constrain the height of the DHP: if the automaton finds a '0' in a node of some level, it will not consider the DHP deeper than this level.

Theorem 1 *The language SQUARE can be recognized by an error-bounded 5-way probabilistic finite automaton in polynomial time, but cannot be recognized by a 5-way deterministic finite automaton.*

The proof of this theorem is given in the next section.

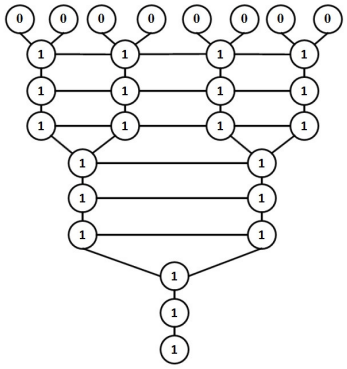

Fig. 2. Example of a word from SQUARE, $N = 3$.

3. Recognizing SQUARE in polynomial time

In this section, we show that it is possible to recognize the language SQUARE in polynomial time from the number of nodes of the input DHP.

To recognize the language $\{0^m1^m\}$ on the standard tape with error probability bounded by an arbitrary $\varepsilon > 0$, Freivalds [1] proposed the gaming approach. We first sketch it here as a reminder. We then adapt this approach to recognizing the language SQUARE.

3.1. Recognizing the language $\{0^m1^m\}$

Essentially, we are given a string 0^k1^n and we need to determine whether $k = n$ with error probability $\varepsilon > 0$. We assume that there are two players, the first playing on the substring of 0's, the second — on the substring of 1's. The idea is to find a game for these two players in which, with arbitrarily high probability, we can determine which of the two substrings is shorter.

The game Freivalds proposed is the following. The game is divided in rounds. In each round, players throw a coin c times on each letter of their substrings, where c is a parameter. Thus, Player 1 throws the coin c times on each letter '0'. If, as a result, he gets c heads, he continues to the next letter '0'. If, on the contrary, at least one tail appeared, Player 1 stops and reports that he has failed the round. If Player 1 is successful and, after going through all k letters of his substring, he has got ck heads, he reports that he has passed the round. Then Player 2 performs the same actions on his substring of n 1's. If exactly one player has passed the round, we report that he has won the round; in case none or both players have passed the round, we report that there is no winner for the round.

The game continues until there are d rounds with a winner, where d is a parameter. After that, we accept the word 0^k1^n if both players have each won at least once. Otherwise, if some player has won all rounds, we say that

the string corresponding to this player is shorter. For instance, if $k < n$, the probability that Player 1 has ck heads in a round is 2^{-ck}, which is higher than 2^{-cn}, the probability that Player 2 has cn heads in a round. Thus, if we do not observe that Player 2 has cn heads when Player 1 does not have ck heads, we conclude that, with high probability, the string of k 0's is shorter than the string of n 1's. In [1], Freivalds showed that, given an arbitrary error probability $\varepsilon > 0$, we can indeed choose parameters c and d such that we can correctly identify whether the word is in $\{0^m 1^m\}$ with probability $(1 - \varepsilon)$.

3.2. Recognizing the language SQUARE

We now turn back to discrete hyperbolic planes. To recognize whether a DHP is in SQUARE, we need to know (1) on what levels branching occurs and (2) which level contains 0's. We remind that we consider DHPs where

- each node is connected to its immediate neighbours on its level,
- branching occurs either in all or in none nodes of a level,
- all nodes of a level contain the same letter (either '0' or '1').

In other words, each node of a level contains all the information about its level that we need. Thus, the 5pfa can infer the structure of the plane by going through any one of the paths connecting the root with any of the last[2] level's nodes. For instance, it can go to the left child on each branching. This allows us to model the movement of the 5pfa in DHP as the movement of the 2pfa in one-dimensional tape. Note that this is done only for illustrative purposes — it is not necessary to convert a DHP to a string word to recognize SQUARE. In the string corresponding to the input DHP, the i-th symbol denotes whether branching occurs on the i-th level ('1' encodes no branching, '2' encodes branching), and the length of the string equals the number of levels starting from the root and ending with the last level. For instance, the string word corresponding to the DHP in Fig. 2 would look like 112112112, as branching occurs only on levels 3, 6, and 9, and there are 9 levels containing 1's. Note how, using the above-stated properties of DHP, we re-formulated the problem of gathering the information about the exponential number of nodes into the problem of gathering the information about the linear number of levels.

To check whether a word is in SQUARE, we use a macro-gaming approach, which is analogical to the gaming approach of Freivalds described in Sec. 3.1. We will be using the string representation of the DHP that we just described. There are $N + 1$ players. Player i, $1 \leq i \leq N$, plays with

[2] By "the last level" we mean the most distant level from the root whose nodes contain 1's.

a substring starting from the letter after the $(i-1)$-st letter '2' and ending with the i-th letter '2'. For instance, if the DHP is characterised by a string 121121112, Player 1 will play with the substring 12, Player 2 — with the substring 112, and Player 3 — with the substring 1112. In our main example, with the string 112112112, the first three players would play with equal strings 112. Player $N+1$ plays with the substring containing all letters '2' from the string. Thus, in our main example, Player 4 would play with the substring 222.

If we denote by L_i the length of the substring of the i-th player, we can note that $L_1, \ldots, L_N$ represent the lengths of the segments between the branching levels, while L_{N+1} represents the number of branching levels. We will now describe a game that checks whether $L_1 = L_2 = \ldots = L_N = L_{N+1}$.

A naïve solution would be to run Freivalds' algorithm $N+1$ times checking that $L_1 = L_2$, $L_2 = L_3$, $\ldots$, $L_{N+1} = L_1$. If the algorithm's answer was positive in all cases, we would respond that the DHP is in SQUARE. The problem with this approach is that the probability of error becomes unbounded. Indeed, each of the $N+1$ comparisons returns the correct result with probability $(1-\varepsilon)$. Thus, the probability that we correctly identify whether the DHP is in SQUARE equals the joint probability that all $N+1$ calls to Freivalds' algorithm provide correct answers, *i.e.*, $(1-\varepsilon)^{N+1}$. Unfortunately, this expression tends to zero as N tends to infinity. Thus, it does not allow us to recognize SQUARE correctly with arbitrarily high probability.

To overcome this problem, we introduce two macro-players, in addition to the $N+1$ ordinary players. These macro-players play the following macro-game.

Macro-player 1 checks c' times whether $L_1 = L_2$ using Freivalds' algorithm. If the macro-player gets c' answers that these two substrings are of equal length, he goes on to compare L_2 with L_3 c' times. Otherwise, if at least one answer is negative, the macro-player stops and reports that he has failed the round. If at the end of the round, Macro-player 1 has got $c'(N+1)$ positive answers from calling Freivalds' algorithm, he passes the current round. Note that if all L_i are indeed equal, the probability of Macro-player 1 passing the round is $(1-\varepsilon)^{N+1}$. If, however, there is a substring of length different from N, the probability that Macro-player 1 passes the round falls to $\varepsilon(1-\varepsilon)^N \ll (1-\varepsilon)^{N+1}$.

Macro-player 2 checks c' times whether $L_1 = L_1$ using Freivalds' algorithm. If he gets c' positive answers, he goes on to compare L_2 with L_2 and so on. Otherwise, if at least one answer is negative, he stops and reports that he has failed the round. As for Macro-player 1, Macro-player 2 passes the round only if he has got $c'(N+1)$ positive answers from calling Freivalds' algorithm.

As in Sec. 3.1, the macro-game is played until d' rounds with a winner, where a round has a winner only if exactly one of the macro-players has passed the round.

The crucial property of Macro-player 2 is that the probability that he passes the round and that $\forall i\,(L_i = L_i)$ is *fixed* at $(1-\varepsilon)^{N+1}$, as no substring can be of different length from itself. Therefore, if the DHP is indeed in SQUARE, both macro-players have equal (albeit, quite small) probabilities of correctly passing the round. This means that if after playing d' macro-rounds with exactly one winner in each, both macro-players have won at least one round, we could say that the DHP belongs to SQUARE.

It is easy to show that to have a probability of error in the macro-game bounded by $\varepsilon' > 0$, the macro-game parameters, c' and d', need to obey the following inequalities

$$\left(\frac{1}{2}\right)^{d'} < \varepsilon', \tag{1}$$

$$\left(\frac{2^{c'}}{1+2^{c'}}\right)^{d'} > 1-\varepsilon'. \tag{2}$$

Indeed, inequality (1) puts a ceiling on the probability of error by requiring that if both macro-players have the same probability of winning a round, the probability that all d' rounds are won by the same player should be bounded by ε'. Inequality (2), in turn, imposes a lower bound on the probability of the correct answer of the algorithm. Given ε', one first sets d' so that inequality (1) holds, and then, in accordance with both ε' and d', sets c' so that inequality (2) holds.

In addition, one has to set the parameters of simple rounds (c, d, and ε). Actually, c and d are set depending on ε using formulas analogous to (1) and (2). It is enough to set the simple-round error probability, ε, at any level below the macro-round error probability, ε'.

3.3. SQUARE is recognized in polynomial time

We have shown in Sec. 3.2 that SQUARE is recognized by a 5-way probabilistic finite automaton. Now we have to show that this is done in polynomial time.

Freivalds' algorithm runs in exponential time [1]. Thus, each game between two ordinary players i and j requires time $O\left(C^{\max\{L_i,L_j\}}\right)$, where C is some constant. Because each macro-player runs this algorithm not more than $c'(N+1)$ times in one round, where c' is a constant, each round requires time

$$\sum_{i=1}^{N} O\left(C^{\max\{L_i,L_{i+1}\}}\right) + O\left(C^{\max\{L_{N+1},L_1\}}\right). \tag{3}$$

Consider two cases. Assume first that the DHP is in SQUARE. Then the expression (3) simplifies to

$$(N+1)O\left(C^N\right) = O\left(NC^N\right) = O\left(\sqrt{L}C^{\sqrt{L}}\right) = O\left(C^L\right) , \tag{4}$$

where L is the number of levels in the input DHP.

Otherwise, the sum (3) is dominated by the largest exponent. We can then roughly approximate it as

$$\sum_{i=1}^{N} O\left(C^{\max\{L_i,L_{i+1}\}}\right) + O\left(C^{\max\{L_{N+1},L_1\}}\right) = O\left(C^L\right) . \tag{5}$$

Thus, in both cases, the running time of one macro-round is exponential from the number of levels. Because the whole macro-game is played until a constant of d' macro-rounds with one winner, the overall time of the algorithm is $O\left(C^L\right)$. Note, however, that the number of nodes containing 1's in the input DHP is exponential from the number of levels, L. This means that the overall running time of the algorithm is *polynomial* from the number of nodes, which completes the proof.

4. Conclusions

This paper contributes to the literature on unusual properties of computation allowed by hyperbolic plane. We have shown that the operation on discrete hyperbolic plane allows error-bounded probabilistic automata to have computational advantage over deterministic automata even in the case when probabilistic automaton works in polynomial time.

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

REFERENCES

[1] R. Freivalds, *Lect. Notes Comput. Sci.* **118**, 33 (1981).
[2] A.G. Greenberg, A. Weiss, *J. Comput. Syst. Sci.* **33**, 88 (1986).
[3] C. Dwork, L. Stockmeyer, *SIAM J. Comput.* **19**, 1011 (1990).
[4] J. Kaŋeps, R. Freivalds, *Lect. Notes Comput. Sci.* **510**, 174 (1991).
[5] M. Margenstern, K. Morita, *Theor. Comput. Sci.* **259**, 99 (2001).
[6] M. Margenstern, *Lect. Notes Comput. Sci.* **2731**, 48 (2003).