Fundamenta Informaticae 72 (2006) 1–18
IOS Press

Automata Recognizing No Words: A Statistical Approach

Cristian S. Calude*
Department of Computer Science, University of Auckland
Private Bag 92019, Auckland, New Zealand
cristian@cs.auckland.ac.nz

Cezar Câmpeanu
Department of Computer Science and Information Technology
University of Prince Edward Island
Charlottetown, P.E.I., C1A 4P3 Canada
cezar@sun11.math.upei.ca

Monica Dumitrescu
Department of Probability Theory Statistics and Operational Research
Faculty of Mathematics and Informatics
Str. Academiei 14, Bucharest, Romania
mdumi@pcnet.ro

Abstract. How likely is that a randomly given (non-) deterministic finite automaton recognizes no word? A quick reflection seems to indicate that not too many finite automata accept no word; but, can this intuition be confirmed?

In this paper we offer a statistical approach which allows us to conclude that for automata, with a large enough number of states, the probability that a given (non-) deterministic finite automaton recognizes no word is close to zero. More precisely, we will show, with a high degree of accuracy (i.e., with precision higher than 99% and level of confidence 0.9973), that for both deterministic and non-deterministic finite automata: a) the probability that an automaton recognizes no word tends to zero when the number of states and the number of letters in the alphabet tend to infinity, b) if the number of states is fixed and rather small, then even if the number of letters of the alphabet of

*Address for correspondence: Department of Computer Science, University of Auckland, Private Bag 92019, Auckland, New Zealand

the automaton tends to infinity, the probability is strictly positive. The result a) is obtained via a statistical analysis; for b) we use a combinatorial and statistical analysis.

The present analysis shows that for all practical purposes the fraction of automata recognizing no words tends to zero when the number of states and the number of letters in the alphabet grow indefinitely. From a theoretical point of view, the result can motivate the search for "certitude", that is, a proof of the fact established here in probabilistic terms.

In the last section we critically discuss the result and the method used in this paper.

Keywords: Finite automata, emptiness problem, statistical analysis, sampling method

1. Introduction

In this paper we ask the question: "How likely is that a randomly given (non-) deterministic finite automaton recognizes no word?" A quick reflection seems to indicate that not too many finite automata accept no word; but, can we offer a proof supporting this intuition? For small automata, i.e., automata with a few states and letters in the alphabet, exact formulae can be obtained; they confirm the intuition. However, it is not clear how to derive similar formulae for 'larger' automata.

A different approach would be to estimate the required probabilities using various techniques of enumerating non-isomorphic finite automata (see, for example, [7]). This method is not only notoriously difficult, but also "problem-sensitive", in the sense that approximations change drastically if we change the problem, e.g., if instead of the emptiness problem we consider the infinity problem. Consequently, in this paper we take a completely new approach, namely we use statistical sampling, see [6, 9]. This approach can be viewed as part of the so-called "experimental mathematics" (see [1, 2, 5]); we will come back to this issue in Section 7.

A deterministic finite automaton (shortly, DFA) $A = (Q, \Sigma, 0, \delta, F)$ consists of a finite set Q of *states*, an input *alphabet* Σ, a fixed *initial state*, 0, a *transition (total) function* $\delta : Q \times \Sigma \to Q$, and a subset F of Q of *final states*. By Σ^* we denote the set of all words (strings) over Σ, with λ as the empty word. The transition function δ extends to $\overline{\delta} : Q \times \Sigma^* \to Q$ by the equations $\overline{\delta}(q, \lambda) = q, \overline{\delta}(q, wa) = \delta(\overline{\delta}(q, w), a)$, for all $q \in Q, a \in \Sigma$ and $w \in \Sigma^*$. The language accepted by A is $L(A) = \{w \in \Sigma^* \mid \overline{\delta}(0, w) \in F\}$.

A non-deterministic finite automaton (shortly, NFA) $A = (Q, \Sigma, 0, \nabla, F)$ consists of the same components as a DFA with the only exception that the transition function ∇ is defined on the power set of Q, that is, $\nabla : Q \times \Sigma \to 2^Q$. The transition function can be naturally extended to $\overline{\nabla} : 2^Q \times \Sigma^* \to 2^Q$ by the equations $\overline{\nabla}(X, \lambda) = X$, $\overline{\nabla}(X, wa) = \bigcup_{q \in \overline{\nabla}(X,w)} \nabla(q, a)$, for all $X \subseteq Q$, $w \in \Sigma^*, a \in \Sigma$. It is seen that $\overline{\nabla}(\overline{\nabla}(X, u), v) = \overline{\nabla}(X, uv)$, for all $X \subseteq Q$, and $u, v \in \Sigma^*$. The language accepted by A is $L(A) = \{w \in \Sigma^* \mid \overline{\nabla}(0, w) \cap F \neq \emptyset\}$.

If $\overline{\nabla}(q, a)$ has just one element for every $q \in Q$ and $a \in \Sigma$, then the automaton is deterministic (and the transition function is denoted by δ). By Δ we will denote either a deterministic transition δ, or a non-deterministic transition ∇.

So, for a DFA or NFA $A = (Q, \Sigma, 0, \Delta, F)$ the question we are interested in is: "How likely is that $L(A) = \emptyset$?" Note that the problem of deciding whether $L(A)$ is empty is decidable in polynomial time. For more details see [10, 11, 12].

In what follows we will fix the states $Q = \{0, 1, \ldots, n-1\}$ and the alphabet $\Sigma = \{1, \ldots, p\}$, and we will count isomorphic copies only once. Let us denote by $\mathbf{DFA}(n,p)$ and $\mathbf{NFA}(n,p)$ the sets of deterministic and non-deterministic finite automata with n states and p letters in the alphabet ($\#(Q) = n, \#(\Sigma) = p$); let $\mathbf{DFAEMPTY}(n,p) = \{A \in \mathbf{DFA}(n,p) \mid L(A) = \emptyset\}$ and $\mathbf{NFAEMPTY}(n,p) = \{A \in \mathbf{NFA}(n,p) \mid L(A) = \emptyset\}$. In order to answer our question we evaluate the proportions of automata accepting the empty language,

$$P_D(n,p) = 100 \cdot \frac{\#\mathbf{DFAEMPTY}(n,p)}{\#\mathbf{DFA}(n,p)}, \; P_N(n,p) = 100 \cdot \frac{\#\mathbf{NFAEMPTY}(n,p)}{\#\mathbf{NFA}(n,p)},$$

and answer the equivalent question: "How likely is that $P_D(n,p) = 0$, $P_N(n,p) = 0$?"

The paper is organized as follows. In the next section we will give exact formulae for the number of DFAs and NFAs recognizing no word. In Section 3 we will describe the statistical method, sampling and prediction. In Sections 4 and 5 we present our main results for DFAs and NFAs, and in Section 6 we briefly describe the programs used for this study. We conclude our paper with a brief section on conclusions, the list of references and data summarizing the main statistical results.

2. Exact formulae

Let $A = (Q, \Sigma, 0, \Delta, F)$ be a DFA or NFA (recall that $\Delta \in \{\delta, \nabla\}$). Assume that Q has n elements and Σ has p elements. A state q is reachable (accessible) in the DFA A if $q = \overline{\delta}(0, w)$, for some $w \in \Sigma^*$; similarly, q is reachable in the NFA A if $q \in \overline{\nabla}(0, w)$, for some $w \in \Sigma^*$. The language $L(A)$ is empty if all reachable states are non-final. This is equivalent to the existence of two sets of states $Q_1, Q_2 \subseteq Q \setminus \{0\}$ such that:

(1) $Q_1 \cup Q_2 = Q \setminus \{0\}$, $Q_1 \cap Q_2 = \emptyset$,

(2) $F \subseteq Q_2$,

(3) $\Delta((Q_1 \cup \{0\}) \times \Sigma) \subseteq Q_1 \cup \{0\}$.

As $Q_2 = Q \setminus (\{0\} \cup Q_1)$, to count the automata accepting the empty language is enough to count the number of sets Q_2 (or Q_1), for each possible set of final states F. Hence, the sets of deterministic and non-deterministic automata with states Q and alphabet Σ accepting the empty language are given by the following formulae:

$$\mathbf{SetDFAEMPTY}(Q, \Sigma, Q_1) = \bigcup_{\substack{Q_1 \subseteq Q \setminus \{0\}, \\ F \not\subseteq (\{0\} \cup Q_1)}} \{(Q, \Sigma, 0, \delta, F) \mid \delta((Q_1 \cup \{0\}) \times \Sigma) \subseteq Q_1 \cup \{0\}\},$$

$$\mathbf{SetNFAEMPTY}(Q, \Sigma, Q_1) = \bigcup_{\substack{Q_1 \subseteq Q \setminus \{0\}, \\ F \subseteq Q \setminus (\{0\} \cup Q_1)}} \{(Q, \Sigma, 0, \delta, F) \mid \nabla((Q_1 \cup \{0\}) \times \Sigma) \subseteq Q_1 \cup \{0\}\}.$$

We first compute the number of DFAs accepting the empty language for a fixed set $Q_1 \subseteq Q$ with k elements, then we multiply the result by the number of subsets Q_1 with k elements. Hence, for a fixed

set of states Q_1 with k elements, the number of DFAs having reachable states in $Q_1 \cup \{0\}$ and final states in $Q \setminus (Q_1 \cup \{0\})$ is

$$\begin{aligned}\#\{(Q,\Sigma,0,\delta,F) \mid \delta((Q_1\cup\{0\})\times\Sigma) \subseteq Q_1\cup\{0\}, & \\ F\setminus(\{0\}\cup Q_1)\} &= (k+1)^{p(k+1)}\cdot n^{(p(n-k-1))}\cdot 2^{n-k-1} \\ &= (k+1)^{p(k+1)}\cdot(2n^p)^{(n-k-1)}.\end{aligned}$$

$$((\#Q_1+1))^{(\#\Sigma\cdot(\#Q_1+1))}\cdot((\#Q))^{(\#\Sigma\cdot(\#Q-\#Q_1-1))}\cdot 2^{\#(Q\setminus Q_1\setminus\{0\})}.$$

For non-deterministic automata, this number is

$$\begin{aligned}\#\{(Q,\Sigma,0,\nabla,F) \mid \nabla((Q_1\cup\{0\})\times\Sigma) \subseteq Q_1\cup\{0\}, & \\ F\setminus(\{0\}\cup Q_1)\} &= (2^{(k+1)})^{(p\cdot(k+1))}\cdot(2^n)^{(p\cdot(n-k-1))}\cdot 2^{(n-k-1)} \\ &= 2^{p(k+1)^2}\cdot 2^{(np+1)(n-k-1)} \\ &= 2^{p(k+1)^2+(np+1)(n-k-1)}.\end{aligned}$$

If $Q_1' \subset Q_1$ and Q_1, Q_1' have properties 1.) – 3). above, then the automata accepting the empty language considered for Q_1' are included in the set of automata accepting the empty language considered for Q_1; therefore, to count them only once, we have to eliminate duplicates. To this aim, the number of DFAs with n states over an alphabet with p letters, accepting the empty language and having exactly $k+1$ reachable states will be denoted by

$$\begin{aligned}\mathbf{emd}(n,p,k) = \#\{(Q,\Sigma,0,\delta,F) \mid \#Q_1 = k, \delta((Q_1\cup\{0\})\times\Sigma)\subseteq Q_1\cup\{0\}, & \\ F\setminus(\{0\}\cup Q_1), \text{ and for all } Q_1'\subset Q_1, & \\ \delta((Q_1'\cup\{0\})\times\Sigma)\not\subseteq Q_1'\cup\{0\} \text{ or } F\not\subseteq Q\setminus(\{0\}\cup Q_1')\}. & \end{aligned} \tag{1}$$

For NFAs, this number will be denoted by

$$\begin{aligned}\mathbf{emn}(n,p,k) = \#\{(Q,\Sigma,0,\nabla,F) \mid \#Q_1 = k, \nabla((Q_1\cup\{0\})\times\Sigma)\subseteq Q_1\cup\{0\}, & \\ F\subseteq Q\setminus Q_1\setminus\{0\}, \text{ and for all } Q_1'\subset Q_1, & \\ \nabla((Q_1'\cup\{0\})\times\Sigma)\not\subseteq Q_1'\cup\{0\} \text{ or } F\not\subseteq Q\setminus(\{0\}\cup Q_1')\}. & \end{aligned} \tag{2}$$

Now, we can write the formulae as:

$$\mathbf{DFAEMPTY}(n,p)=\sum_{k=0}^{n-1}\mathbf{emd}(n,p,k),\ \mathbf{NFAEMPTY}(n,p)=\sum_{k=0}^{n-1}$$

For example, in the case $p=1$, these formulae become:

$$\mathbf{emd}(n,1,k)=\binom{n-1}{k}\cdot(k+1)\cdot n^{n-k-1}\cdot 2^{n-k-1},$$

and

$$\mathbf{emn}(n,1,k)=\binom{n-1}{k}\cdot(2^k 2^{k+1})\cdot(2^n)^{n-k-1}\cdot 2^{n-k-1},$$

therefore,

$$\#\mathbf{DFAEMPTY}(n,1) = \sum_{k=0}^{n-1} \mathbf{emd}(n,1,k) = \sum_{k=0}^{n-1} \binom{n-1}{k} \cdot (k+1) \cdot n^{n-k-1} \cdot 2^{n-k-1} = \sum_{k=1}^{n} \binom{n-1}{k+1} \cdot k \cdot n^{n-k} \cdot 2^{n-k},$$

$$\#\mathbf{NFAEMPTY}(n,1) = \sum_{k=0}^{n-1} \mathbf{emn}(n,1,k) = \sum_{k=0}^{n-1} \binom{n-1}{k} \cdot (2^k 2^{k+1}) \cdot 2^{n\,n-k-1} \cdot 2^{n-k-1} = \sum_{k=1}^{n} \binom{n-1}{k+1} \cdot 2^{2k+1+(n+1)(n-k)}.$$

Since computing the above functions is difficult for arbitrary p, we restrict the computation to $n = 1, 2, 3$.

For DFAs, we have the following formulae:

1. $\#\mathbf{DFAEMPTY}(1,p) = 1$,
2. $\#\mathbf{DFAEMPTY}(2,p) = 2^p(1+2^p)$,
3. $\#\mathbf{DFAEMPTY}(3,p) = 3^{3\cdot p} + 3^{2\cdot p+1} + 2^{p+1} \cdot 3^p(2^{2\cdot p} - 1)$.

Thus, the proportions of DFAs accepting the empty language are:

$$P_D\,(2,p) = 100 \cdot \frac{2^p(1+2^p)}{2^2 \cdot 2^{2p}} = \frac{100}{2^2} + \frac{100}{2^{p+2}},$$

$$P_D\,(3,p) = 100 \cdot \frac{3^{3\cdot p} + 3^{2\cdot p+1} + 2^{p+1} \cdot 3^p(2^{2\cdot p} - 1)}{2^3 \cdot 3^{3p}} = \frac{100}{2^3} + \frac{100}{2^3 \cdot 3^{p-1}} + \frac{100}{2^2}\left(\frac{2}{3}\right)^p \cdot \frac{2^p - 1}{3^p}.$$

Hence,

$$\lim_{p\to\infty} P_D\,(2,p) = 25\%, \quad \lim_{p\to\infty} P_D\,(3,p) = 12.5\%.$$

For NFAs, we have the following formulae:

1. $\#\mathbf{NFAEMPTY}(1,p) = 2^p$,
2. $\#\mathbf{NFAEMPTY}(2,p) = 2^{4p} + 2^{3p}$,
3. $\#\mathbf{NFAEMPTY}(3,p) = 2^{9p} + 5 \cdot 2^{7p} - 2^{6p+1}$.

Thus, the proportions of NFAs accepting the empty language are:

$$P_N\,(2,p) = 100 \cdot \frac{2^{4p} + 2^{3p}}{2^{2(2p+1)}} = \frac{100}{2^2} + \frac{100}{2^{2+p}},$$

$$P_N\,(3,p) = 100 \cdot \frac{2^{9p} + 5 \cdot 2^{7p} - 2^{6p+1}}{2^{3(3p+1)}} = \frac{100}{2^3} + \frac{500}{2^{2p+3}} - \frac{100}{2^{3p+2}}.$$

Hence,

$$\lim_{p\to\infty} P_N(2,p) = 25\%, \quad \lim_{p\to\infty} P_N(3,p) = 12.5\%.$$

These results can be verified against the exact results obtained using brute force algorithms in Table 2 and Table 4.

3. Sampling and Prediction

The formulae established for $n = 2, 3$ offer the exact values of $P_D(2,p)$, $P_D(3,p)$, $P_N(2,p)$, $P_N(3,p)$, for any $p = 2, 3, \ldots$ As for $n > 3$ it is very difficult to obtain exact formulae, we use a statistical approach in order to construct a predictor of $P(n,p)$ (here P is P_D or P_N). Using the vector notation $\mathbf{t} = (n,p)^T$, we construct a predictor $\widetilde{P} = 100 - g(\mathbf{t})$, where g is an unknown, smooth surface. The steps of the statistical approach are the following:

- Choose a *grid* of k classes of automata of type (n_i, p_i), $i = 1, \ldots, k$.
- For each i, take a random sample of size m from the family of automata characterized by $\mathbf{t}_i = (n_i, p_i)^T$ and determine the proportion of automata recognizing the empty language in the sample. Thus we obtain an estimation P_i of $P(\mathbf{t}_i)$.
- Consider the set of available data, obtained through random sampling

$$\left(\mathbf{t}_i = (n_i, p_i)^T, P_i\right), i = 1, \ldots, k.$$

  Since P depends on (n,p), we use the traditional statistical interpretation: $\mathbf{t} = (n,p)^T$ is the *design variable*, and P is the *response variable.* A statistical model of this dependence can be presented as

$$P = 100 - g(\mathbf{t}) + error,$$

  where g is an unknown, smooth surface, verifying the condition

$$g(\mathbf{t}_i) = 100 - P_i, i = 1, \ldots, k.$$

  We estimate the function $g(\mathbf{t})$ through the natural thin plate spline interpolant.

The populations we will sample from are the sets of DFAs or NFAs, and their parameters are pairs (n,p), with $n = 2, 3, \ldots$, $p = 2, 3, \ldots$ The volumes of these populations (the total number M of automata) increase exponentially with n and p according to the following formulae: $M_1 = 2^n \cdot n^{np}$, for DFAs, and $M_2 = 2^{n(np+1)}$, for NFAs. In order to classify these populations according to their sizes, we will use the results in Table 1 and Table 5.

From a statistical point of view, populations with $M \leq 5,000$ are considered *small* sized and, for their investigations, one would take a census. Families with $5,000 < M \leq 20,000$ are considered *medium* sized, and those with $M > 20,000$ would be looked upon as *large* populations.

For each family, characterized by a couple (n,p), we are interested in the estimation of the proportion P of automata which recognize no words, *the property* $\mathcal{P}$. For medium sized populations, sampling without replacement (according to a hyper-geometric scheme) has been used, while for large ones we

used sampling with replacement. The estimator $\hat{P}$ is the proportion of automata in the sample accepting the empty language. The size m of the sample has been established in such a way that the estimator $\hat{P}$ offers a specified level of precision. This precision can be expressed in terms of the coefficient of variation

$$c_0 = cv\left(\hat{P}\right) = \frac{\sqrt{Var\left(\hat{P}\right)}}{E\left(\hat{P}\right)} = \sqrt{\frac{M-m}{M-1} \cdot \frac{1-P}{mP}},$$

for medium sized populations. We take into consideration the most "severe" case $P = 1/2$, therefore, for a specified precision c_0, the sample size m is given by the expression

$$m = \frac{M}{1 + c_0^2 (M-1)}.$$

For large families, the normal approximation can be applied. Hence, the following relation is true:

$$\Pr\left(\mid \hat{P} - P \mid < z_{1-(\alpha/2)} \sqrt{Var\left(\hat{P}\right)} \right) = 1 - \alpha,$$

where $z_{1-(\alpha/2)}$ is the $(1-(\alpha/2))$ quantile of the normal $N(0,1)$ distribution. The sample size m which offers the precision c_0 is the solution of the equation

$$z_{1-(\alpha/2)} \cdot \sqrt{\frac{P(1-P)}{m}} = c_0.$$

In the absence of any prior knowledge about P, we will choose the value which maximizes the product $P(1-P)$, that is $P = 1/2$. Hence, for large (infinite) populations, the "safest" estimation of the sample size m is

$$m = \frac{z_{1-(\alpha/2)}^2 \cdot 2,500}{c_0^2},$$

with c_0 expressed as a percentage; see [6, 8].

In our study, we use the precision $c_0 = 1\%$ and the confidence level $1 - \alpha = 0.9973$. Hence, the sample sizes m_1 (for DFAs) and m_2 (for NFAs) are presented in Table 3 and Table 6.

Actually, as we have exact formulae for P for $n = 2$ and for $n = 3$, we do not perform random sampling for automata of the types $(2,p)$ and $(3,p)$. Therefore, *all generated samples for families of automata with* (n,p), $n > 3$, *have the size* $m = 22,500$.

For prediction, let us assume that a grid of k classes of automata characterized by (n_i, p_i), $i = 1, \ldots, k$, has been chosen, and $P_i = P(n_i, p_i)$ has been estimated through $\hat{P}_i$ by the above method.

Given the data $((n_i, p_i), P_i)$, $i = 1, \ldots, k$, the natural way to view the relationship between the design variable $\mathbf{t} = (n,p)^T$ and the response variables P by fitting a model of the form

$$P = 100 - g(\mathbf{t}) + error,$$

to the available data. Here g is a $2-$dimensional *surface*, and its estimation can be obtained by a roughness penalty method. Since the main purpose of this approach is to use the design-response model for prediction, we want to find a smooth curve g that interpolates the points $((n_i, p_i), P_i)$, that is $g(n_i, p_i) = P_i$,

for all $i = 1, \ldots, k$. The method we use, called *thin plate splines*, is a natural generalization of cubic splines and the associated predictor is called the *thin plate spline predictor*, see [9].

Suppose that $\mathbf{t}_i = (n_i, p_i)^T$, $i = 1, \ldots, k$, are the available knots in $\mathbb{R}^2$, and z_i, $i = 1, \ldots, k$, are known values. We look for a smooth function $g(\mathbf{t})$, such that $g(\mathbf{t}_i) = z_i$ for $i = 1, \ldots, k$. To this aim we define the function $\eta(r)$ by

$$\eta(r) = \begin{cases} \frac{1}{16\pi} r^2 \log r^2, & \text{for } r > 0, \\ 0, & \text{for } r = 0, \end{cases}$$

and the matrix

$$\mathbf{T} = \begin{pmatrix} 1 & 1 & \ldots & 1 \\ n_1 & n_2 & \ldots & n_k \\ p_1 & p_2 & \ldots & p_k \end{pmatrix}. \tag{3}$$

A function $g(\mathbf{t})$ is called a *thin plate spline* on the data set $\mathbf{t}_i$, $i = 1, \ldots, k$, if g is of the form

$$g(\mathbf{t}) = \sum_{i=1}^{k} \delta_i \cdot \eta(\| \mathbf{t} - \mathbf{t}_i \|) + (a_1 + a_2 n + a_3 p),$$

for suitable constants δ_i and a_i. If the vector δ of coefficients δ_i satisfies the equation $\mathbf{T}\delta = \mathbf{0}$, then g is said to be a *natural thin plate spline* (NTPS).

Interpolation will be based on the following result presented in [9]: *Suppose that* $\mathbf{t}_i = (n_i, p_i)^T$, $i = 1, \ldots, k$, *are non-collinear knots in* $\mathbb{R}^2$, *and* z_i, $i = 1, \ldots, k$, *are given values. There exists a unique NTPS* g, *such that* $g(\mathbf{t}_i) = z_i$, $i = 1, \ldots, k$, *which uniquely minimizes* $J(g)$, *where*

$$J(g) = \iint_{R^2} \left\{ \left(\frac{\partial^2 g}{\partial n^2} \right)^2 + 2 \left(\frac{\partial^2 g}{\partial n \partial p} \right)^2 + \left(\frac{\partial^2 g}{\partial p^2} \right)^2 \right\} dn dp.$$

Based on the above result we can use the following NTPS interpolation algorithm. The input data consists of:

1. k is the number of interpolation knots,
2. $\mathbf{t}_i$, $i = 1, \ldots, k$, are the points in $\mathbb{R}^2$, $\mathbf{t}_i = (n_i, p_i)^T$,
3. $z_i = \hat{P}_i$, $i = 1, \ldots, k$, are the calculated percentages of automata recognizing at least one word (the estimated values obtained by sampling).

As matrices we use $\mathbf{T}$ in (3) and define the $k \times k$ matrix $\mathbf{E} = (E_{ij})$ by

$$E_{ij} = \eta(\| \mathbf{t}_i - \mathbf{t}_j \|) = \frac{1}{16\pi} \| \mathbf{t}_i - \mathbf{t}_j \|^2 \log \| \mathbf{t}_i - \mathbf{t}_j \|^2.$$

Denote $\mathbf{z} = (z_1, \ldots, z_k)^T$. To construct the NTPS interpolant (predictor) we calculate the coefficients $\delta = (\delta_1, \ldots, \delta_k)^T$, $\mathbf{a} = (a_1, a_2, a_3)^T$ of the NTPS $g(\mathbf{t})$, interpolating the values z_i, as the solution of the linear system

$$\begin{pmatrix} \mathbf{E} & \mathbf{T}^T \\ \mathbf{T} & \mathbf{0} \end{pmatrix} \begin{pmatrix} \delta \\ \mathbf{a} \end{pmatrix} = \begin{pmatrix} \mathbf{z} \\ \mathbf{0} \end{pmatrix},$$

whose matrix is of full rank.

The knots we use for interpolation, $\left((n_i, p_i), \hat{P}_i\right), i = 1, \ldots, k$, are obtained by statistical means and the confidence level for the estimations $\hat{P}_i, i = 1, \ldots, k$ is $(1 - \alpha)$, at a specified precision c_0. Hence, *the prediction based on the NTPS $g(\mathbf{t})$ has the same precision c_0, with the confidence level $(1 - \alpha)$.* In our study we use the precision $c_0 = 1\%$ and the confidence level $1 - \alpha = 0.9973$.

Using the function $g(\mathbf{t})$, estimated through the thin-plate spline method, we obtain a predictor for the percent–empty, which can be used for all $\mathbf{t} = (n, p)^T$. The predictor is forced to tend to a flat function (a plane) for $n \to \infty$, $p \to \infty$. Of course, one would not expect negative values for P, therefore, the predictor we choose is $\widetilde{P} = \max\{100, g(\mathbf{t})\}$.

4. Deterministic Finite Automata

This section presents the samples, estimations, and predictions for both DFAs and NFAs corresponding to a precision of $c_0 = 1\%$ and confidence level $1 - \alpha = 0.9973$.

Table 7 gives the number of DFAs accepting the empty language and the computed percent of DFAs accepting a non-empty language, using randomly generated samples.

We tested DFA samples, randomly generated for the first 13 values of n and p in Table 7, obtaining the corresponding percentage of DFAs accepting the empty language for each such pair (n, p). Using these values, we computed the NTPS predictor g for the last 6 values of n and p in Table 7, obtaining the results in Table 8.

Computing the percent of DFAs accepting the empty language for values of n and p ranking from 14 to 24 (see Table 7) and the corresponding NTPS predictor g for the last 6 values of n and p in Table 7, we obtain the results in Table 9. As we can see, the difference between the statistical results obtained by generating samples and the estimated percent computed using the NTPS predictor (the "Precision" column) is less than 1% in both cases, for all six values of $(n, p) : (4, 7), (8, 6), (9, 5), (10, 4), (13, 3), (15, 2)$.

The prediction of the proportion P of DFAs recognizing at least one word can be expressed in terms of the NTPS, by taking advantage of the exact formulae too.

Thus, for $n = 2$, the exact predictor of P is

$$P(2, p) = \frac{300}{2^2} - \frac{100}{2^{p+2}}, \lim_{p \to \infty} P(2, p) = 75\%.$$

In a similar way, the exact predictor of P when $n = 3$ is

$$P(3, p) = \frac{700}{2^3} - \frac{100}{2^3 \cdot 3^{p-1}} - \frac{100}{2^2}\left(\frac{2}{3}\right)^p \cdot \frac{2^p - 1}{3^p}, \lim_{p \to \infty} P(3, p) = 87.5\%.$$

The construction of the NTPS predictor has been made by using $k = 20$ knots, which have been chosen to "cover" the region $n \geq 3, p \geq 2$. (As we have mentioned before, the predictor has the precision $c_0 = 1\%$ and the confidence level $1 - \alpha = 0.9973$.)

The validation of the predictor has been obtained by comparisons between predictions and statistically generated values of P for six different points $\left((n_i, p_i), \hat{P}_i\right)$. The numerical results are presented in Table 8 and Table 9, where we can see that the precision of the prediction is always less than $c_0 (= 1.0\%)$. Consequently, *with a high degree of accuracy (i.e., with precision higher than 99% and level of confidence 0.9973), the probability that a DFA recognizes no word tends to zero when the number of states and the number of letters in the alphabet tend to infinity.*

5. Non-deterministic Finite Automata

Table 10 gives the number of NFAs accepting the empty language and the computed percent of NFAs accepting a non-empty language using randomly generated samples.

Applying the same procedure described for DFAs, but this time for NFAs, we obtain the NTPS predictor g for NFA accepting the empty language. Using the first 13 values from Table 10, we obtain the results for the NTPS predictor g in Table 11. Using the first 13 values and the supplementary 11 values from Table 10, we get the results in Table 12.

As we can see, the difference between the percentage obtained by generating samples and the one computed using the NTPS predictor is again less than 1.65%, if we are using only 13 knots, and less than 0.999%, if we are using 24 knots.

Again, for NFAs we obtained the same conclusion as for DFAS: *with a high degree of accuracy (i.e., with precision higher than 99% and level of confidence 0.9973), the probability that an NFA recognizes no word tends to zero when the number of states and the number of letters in the alphabet tend to infinity.*

6. Programs

We used the following uniform binary representation of both finite deterministic and non-deterministic automata $A = (\Sigma, Q, \Delta, 0, F)$ of type (n, p):

- $Q = \{0, 1, \ldots, n-1\}$, $\Sigma = \{1, \ldots, p\}$;
- states are represented by their characteristic functions, i.e., state i is represented by the binary vector $(0, 0, \ldots, 0, 1, 0, \ldots, 0)$ with 1 on the ith position; $(1, 0, \ldots, 0)$ represents the initial state;
- the transition Δ and the vector F are represented by an array V consisting of $n \times p \times n + n$ 0's and 1's; the first $n \times p \times n$ binary digits of V represent the characteristic vector of the transition function Δ, so we have $n \times p$ groups of n digits, each of them representing the characteristic vector of a value of $\Delta(i, j)$, $1 \leq i \leq n$, $1 \leq j \leq p$;
- the last n digits of V represent the characteristic vector of the final states F.

Both DFAs and NFAs use the same representation, the only difference being that for $\Delta(i, j)$ we have a characteristic vector with *exactly one* value of 1 for DFAs, while for NFAs, the number of 1's can be $0, 1, \ldots, n$. Therefore, we use the same code for testing the emptiness property for both DFAs and NFAs: first, we compute reachable states, afterwards, we check if any reachable state is final.

For example, the DFA $A = (\Sigma, Q, \delta, 0, F)$ where $\Sigma = \{1, 2\}, Q = \{0, 1\}, F = \{0, 1\} = \{(\mathbf{1}, \mathbf{0}), (\mathbf{0}, \mathbf{1})\}$ and $\delta(0, 1) = 1 = (\mathbf{0}, \mathbf{1}), \delta(0, 2) = 0 = (\mathbf{1}, \mathbf{0}), \delta(1, 1) = \delta(1, 2) = 1 = (\mathbf{0}, \mathbf{1})$ is represented by the binary string $\mathbf{0110010111}$. The NFA $B = (\Sigma, Q, \Delta, 0, F)$ where all components are the same as in A except the transition $\Delta(0, 1) = \{0, 1\} = (\mathbf{1}, \mathbf{1}), \Delta(0, 2) = \{0\} = (\mathbf{1}, \mathbf{0}), \Delta(1, 1) = \Delta(1, 2) = \{1\} = (\mathbf{0}, \mathbf{1})$ is represented by the binary string $\mathbf{1110010111}$.

For computing the number of automata accepting the empty language, for fixed values of n and p, we generate in lexicographical order all possible binary vectors V and test each of them whether it accepts or not no word. Obviously, the number of automata grows exponentially with n and p ($n^{np} \cdot 2^n$ for DFAs and $2^{n^2 p + n}$ for NFAs). The method was used for the values presented in Table 2 and Table 4. One can see that our formulae obtained in Section 2 match the results in these tables. For sampling, we test randomly

generated automata (DFAs and NFAs of different types) by a simple Mathematica program. The results are presented in Table 7 and Table 10. Note that the statistics is very close to those in Table 2 and Table 4, respectively. For most of them, the difference is less than 1%. We always consider 0 to be the initial state. Since we generate (in lexicographical order) binary strings with the last n digits being interpreted as an array of final states, each of the first n^{np} generated automata recognizes the empty language for DFA, and each of the first 2^{n^2p} generated automata recognizes the empty language for NFA. The last $n^{np}2^{n-1}$ generated automata recognize a non-empty language for DFA and the last 2^{n^2p+n-1} generated automata recognize a non-empty language for NFA.

For the NTPS predictor we have codes for solving systems of linear equations using the substitution lemma, computing the function η, building the system of equations for the NTPS predictor, constructing the NTPS predictor g, and computing the NTPS predictor corresponding to the given values n and p.

We use the language C, compiled with a GNU compiler for Linux. The programs were run on a PC Pentium 4 1.6A with 64 MB memory, for more than 1 week to obtain the results in Table 2 and Table 4. The size of memory was not important since every time we store only one automaton and no swapping of data is used. All programs and data used for this paper can be found at the website `http://www.csit.upei.ca/~ccampeanu/Research/Automata/ProbNoWords/`.

7. Conclusions

In this paper we offered an answer to the question: "How likely is that a randomly given (non-) deterministic finite automaton recognizes no word?" The intuition seems to indicate that not too many finite automata accept no word; but, is there a proof supporting this intuition? For small automata, i.e., automata with a few states and letters in the alphabet, exact formulae can be obtained; they confirm the intuition. However, it is not clear how to derive similar formulae for 'larger' automata (see [7] for formulae which might be relevant; enumeration is not only notoriously difficult, but also "problem-sensitive", in the sense that approximations change drastically if we change the problem). Consequently, in this paper we took a completely new approach, namely, statistical sampling, see [6, 9].

We have shown that, with a high degree of accuracy (i.e., with precision higher than 99% and level of confidence 0.9973), for both deterministic and non-deterministic finite automata: a) the probability that an automaton recognizes no word tends to zero when the number of states and the number of letters in the alphabet tend to infinity, b) if the number of states is fixed and rather small, then even if the number of letters of the alphabet of the automaton tends to infinity, the probability is strictly positive.

It is interesting to examine briefly the meaning of our results. First and foremost, *the main claims of the paper are statistically true*: the statements a) and b) above are true with a high degree of accuracy (i.e., with precision higher than 99% and level of confidence 0.9973). Is this just a simple 'guess'? Do we use a valid method for ascertaining mathematical truth? Does this analysis really add anything to our knowledge of the phenomenon studied? Is the result interesting?

The sampling method is neither simple "guess", nor "bad mathematics". It is part of a trend called "experimental mathematics", in which we proceed heuristically and 'quasi-inductively', with a blend of logical and empirical–experimental arguments (see, for example, [1, 2, 5]). It is one of the possible ways to cope with the complexity of mathematical phenomena, a valid method for ascertaining mathematical truth. The present analysis shows that for all practical purposes the fraction of automata recognizing no words tends to zero when the number of states and the number of letters in the alphabet grow indefinitely.

Of course, the result obtained in this note is not unexpected. Therefore, some may argue that it is not very interesting from the point of view of automata theory. We believe this is not the case for the following reasons. a) Sampling and simulation are current methods in other areas of mathematics and computer science, and their absence in automata theory was a matter of time. b) We have a probabilistic result which can motivate/guide the search for "certitude", that is, a proof of the fact established here in probabilistic terms. c) In fact, the method used is much more important than the result itself, and this is the reason we tested it for such a simple problem. The method is "general" in the sense that it can be applied to a variety of questions in automata theory, certainly more difficult ones than the problem solved in this note. For example, an interesting question is "How likely is that a randomly given (non-) deterministic finite automaton recognizes an infinite set of words?".

Acknowledgement: We thank Sheng Yu for useful suggestions leading to a better presentation.

References

[1] J.M. Borwein, D. Bailey: *Mathematics by Experiment: Plausible Reasoning in the 21st Century*. A.K. Peters, Natick, MA, 2003.

[2] J.M. Borwein, D. Bailey, R. Girgensohn: *Experimentation in Mathematics: Computational Paths to Discovery*. A.K. Peters, Natick, MA, 2004.

[3] C.S. Calude, E. Calude, M.J. Dinneen: What is the value of $Taxicab(6)$? *J. UCS*, 9, 10 (2003), 1196–1203.

[4] C.S. Calude, E. Calude, T. Chiu, M. Dumitrescu, R. Nicolescu: Testing computational complementarity for Mermin automata. *J. Multi Valued Logic*, 6 (2001), 47–65.

[5] C.S. Calude, S. Marcus: Mathematical proofs at a crossroad? In *Theory Is Forever* (J. Karhumäki, H. Maurer, G. Păun, G. Rozenberg, eds.), LNCS 3113, Springer, Berlin, 2004, 15–28.

[6] W.G. Cochran: *Sampling Techniques*, 3rd edition. Wiley, New York, 1977.

[7] M. Domaratzki, D. Kisman, J. Shallit: On the number of distinct languages accepted by finite automata with n states. *J. Automat. Lang. Comb.*, 7 (2002), 469–486.

[8] M. Dumitrescu: *Statistical Surveys and Applications*. Editura Tehnică, Bucharest, 2000 (in Romanian).

[9] P.J. Green, B.W. Silverman: *Non-parametric Regression and Generalized Linear Models*. Chapman & Hall, London, 1994.

[10] D. Kozen: *Automata and Computability.* Springer, New York, 1997.

[11] A. Salomaa: *Computation and Automata*. Cambridge University Press, Cambridge, 1985.

[12] S. Yu: Regular languages. In *Handbook of Formal Languages, Vol. 1* (G. Rozenberg, A. Salomaa, eds.). Springer, Berlin, 1997, 41–110.

Appendix: Data

In this section we present the main statistical data on which our analysis is based upon.

Table 1. DFAs recognizing no words

$M_1 : n/p$	2	3	4	5	6
2	64	256	$1,024$	$4,096$	$16,384$
3	5832	1.5746×10^5	4.2515×10^6	1.1479×10^8	3.0994×10^9
4	1.0486×10^6	2.6844×10^8	...	...	...
5	3.125×10^8	...	...	...	...

Table 2. DFA exact results

No.	n	p	Total number of DFAs	DFAs accepting empty languages	DFAs accepting non-empty languages	Non-empty percent
1	2	2	64	20	44	68.75%
2	2	3	256	72	184	71.875%
3	2	4	1024	272	752	73.4375%
4	2	5	4096	1056	3040	74.2188%
5	2	6	16384	4160	12224	74.6094%
6	2	7	65536	16512	49024	74.8047%
7	2	8	262144	65792	196352	74.9023%
8	2	9	1048576	262656	785920	74.9512%
9	2	10	4194304	1049600	3144704	74.9756%
10	2	11	16777216	4196352	12580864	74.9878%
11	2	12	67108864	16781312	50327552	74.9939%
12	2	13	268435456	67117056	201318400	74.9969%
13	2	p	exact formulae			
14	3	2	5832	1188	4644	79.6296%
15	3	3	157464	24894	132570	84.1907%
16	3	4	4251528	590004	3661524	86.1225%
17	3	5	114791256	15008166	99783090	86.9257%
18	3	p	exact formulae			
19	4	2	1048576	148640	899936	85.8246%
20	4	3	268435456	26036864	242398592	90.3005%
21	5	2	312500000	32383000	280117000	89.6374%

Table 3. Sample sizes for DFAs

$m_1 : (n/p)$	2	3	4	5	6
2	64	256	1,024	4,096	$6,211$
3	$3,684$	$22,500$	$22,500$	$22,500$	$22,500$
4	$22,500$	$22,500$	$22,500$	$22,500$	$22,500$
5	$22,500$	$22,500$	...	...	...

Table 4. NFA: exact results

No.	n	p	Total number of NFAs	NFAs accepting non-empty languages	NFAs accepting non-empty languages	Non-empty percent
1	2	2	1024	704	320	68.75%
2	2	3	16384	11776	4608	71.875%
3	2	4	262144	192512	69632	73.4375%
4	2	5	4194304	3112960	1081344	74.2188%
5	2	p	exact formulae			
6	3	2	2097152	1761280	335872	83.9844%
7	3	3	1073741824	929562624	144179200	86.5723%
8	3	p	exact formulae			
9	4	2	68719476736	63671631873	5047844863	92.6544%
10	5	2	36028797018963968	Beyond the computing power	N/A	N/A

Table 5. NFAs recognizing no words

$M_1 : n/p$	2	3	4	5
2	$1,024$	$16,384$	2.6214×10^5	4.1943×10^6
3	2.0972×10^6	1.0737×10^9	...	...
4	6.8719×10^{10}	...	...	...

Table 6. Sample sizes for NFAs

$m_2 : (n//p)$	2	3	4	5
2	1024	$6,211$	$22,500$	$22,500$
3	$22,500$	$22,500$	$22,500$	$22,500$
4	$22,500$	$22,500$	...	...

Table 7. The number of DFAs accepting the empty language using randomly generated samples

No.	n	p	Total number of DFAs	DFAs accepting non-empty languages	DFAs accepting non-empty languages	Non-empty percent
1	3	2	22500	17893	4607	79.52 %
2	3	3	22500	19017	3483	84.52 %
3	3	8	22500	19695	2805	87.53 %
4	3	15	15500	13498	2002	87.08 %
5	4	2	22500	19425	3075	86.33 %
6	4	6	22500	21063	1437	93.61 %
7	6	2	22500	21034	1466	93.48 %
8	6	6	22500	22122	378	98.32 %
9	6	10	22500	22155	345	98.47 %
10	8	2	22500	21761	739	96.72 %
11	8	3	22500	22308	192	99.15 %
12	8	8	22500	22413	87	99.61 %
13	10	5	22500	22471	29	99.87 %
14	4	10	22500	21068	1432	93.64 %
15	5	3	22500	21376	1124	95 %
16	5	5	22500	21743	757	96.64 %
17	5	10	22500	21779	721	96.8 %
18	6	4	22500	22115	385	98.29 %
19	6	8	22750	22426	324	98.58 %
20	6	11	22500	22118	382	98.3 %
21	7	9	22500	22312	188	99.16 %
22	9	4	22500	22434	66	99.71 %
23	10	3	22500	22435	65	99.71 %
24	14	2	22500	22371	129	99.43 %
25	4	7	22500	21064	1436	93.62 %
26	8	6	22500	22402	98	99.56 %
27	9	5	22500	22454	46	99.8 %
28	10	4	22500	22476	24	99.89 %
29	13	3	22500	22487	13	99.94 %
30	15	2	15400	15332	68	99.56 %

Table 8. Comparative results for DFA NTPS predictor using 13 knots

No	n	p	NFAs tested	DFAs accepting at least one word	% DFAs accepting at least one word	$g(n,p)$	NTPS estimated empty percent	Precision
1	4	7	22500	21064	93.620%	92.754207	7.245793	0.865793
2	8	6	22500	22402	99.560%	99.693417	0.306583	-0.133417
3	9	5	22500	22454	99.800%	99.869872	0.130128	-0.069872
4	10	4	22500	22476	99.890%	99.844233	0.155767	0.045767
5	13	3	22500	22487	99.940%	100.00	0.00	-0.06
6	15	2	15400	15332	99.560%	100.00	0.00	-0.44

Table 9. Comparative results for DFA NTPS predictor using 24 knots

No	n	p	NFAs tested	NFAs accepting at least one word	% NFAs accepting at least one word	$g(n,p)$	NTPS estimated empty percent	Precision
1	4	7	22500	21064	93.620%	93.358144	6.641856	0.261856
2	8	6	22500	22402	99.560%	99.330491	0.669509	0.229509
3	9	5	22500	22454	99.800%	99.484803	0.229509	0.315197
4	10	4	22500	22476	99.890%	100	0.00000	-0.11
5	13	3	22500	22487	99.940%	100.000	0.000	-0.06
6	15	2	15400	15332	99.560%	100.000	0.000	-0.44

Table 10. The number of NFAs accepting the empty language using randomly generated samples

No.	n	p	Total number of NFAs	NFAs accepting non-empty languages	NFAs accepting non-empty languages	Non-empty percent
1	3	2	22500	18906	3594	84.03 %
2	3	3	22500	19491	3009	86.63 %
3	3	8	22500	19651	2849	87.34 %
4	3	15	22500	19775	2725	87.89 %
5	4	2	22500	20842	1658	92.63 %
6	4	6	22500	21098	1402	93.77 %
7	4	7	22500	21121	1379	93.87 %
8	4	10	22500	21094	1406	93.75 %
9	5	5	22500	21778	722	96.79 %
10	5	10	22500	21807	693	96.92 %
11	6	2	22500	22127	373	98.34 %
12	6	4	22500	22147	353	98.43 %
13	6	6	22500	22126	374	98.34 %
14	6	8	22500	22161	339	98.49 %
15	6	10	22500	22137	363	98.39 %
16	6	11	22500	22130	370	98.36 %
17	7	9	22500	22352	148	99.34 %
18	8	2	22500	22407	93	99.59 %
19	8	3	22500	22419	81	99.64 %
20	8	6	22500	22403	97	99.57 %
21	8	8	22500	22426	74	99.67 %
22	9	4	22500	22458	42	99.81 %
23	9	5	22500	22448	52	99.77 %
24	10	3	22500	22477	23	99.9 %
25	10	4	22500	22479	21	99.91 %
26	10	5	22500	22477	23	99.9 %
27	13	3	22500	22499	1	100 %
28	14	2	22500	22498	2	99.99 %
29	15	2	22500	22499	1	100 %

Table 11. Comparative results for NFA NTPS predictor using 13 knots

No	n	p	NFAs tested	NFAs accepting at least one word	% NFAs accepting at least one word	$g(n,p)$	NTPS estimated empty percent	Precision
1	6	8	22500	22161	98.490%	99.462347	0520447	-0.972347
2	6	10	22500	22137	98.390%	100.000	0.000	-1.61
3	6	11	22500	22130	98.360%	100.000	0.000	-1.64
4	7	9	22500	22352	99.340%	100.000	0.000	-0.66
5	8	2	22500	22407	99.590%	100.000	0.000	-0.41
6	8	3	22500	22419	99.640%	100.000	0.000	-0.36
7	8	6	22500	22403	99.570%	100.000	0.000	-0.43
8	8	8	22500	22426	99.670%	100.000	0.000	-0.33
9	9	4	22500	22458	99.810%	100.000	0.000	-0.19
10	9	5	22500	22448	99.770%	100.000	0.000	-0.23
11	10	3	22500	22477	99.900%	100.000	0.000	-0.01
12	10	4	22500	22479	99.910%	100.000	0.000	-0.09
13	10	5	22500	22477	99.900%	100.000	0.000	-0.1
14	13	3	22500	22499	100.000%	100.000	0.000	0
15	14	2	22500	22498	99.990%	100.000	0.000	-0.01
16	15	2	22500	22499	100.000%	100.000	0.000	0

Table 12. Comparative results for NFA NTPS predictor using 24 knots

No	n	p	NFAs tested	NFAs accepting at least one word	% NFAs accepting at least one word	$g(n,p)$	NTPS estimated empty percent	Precision
1	4	7	22500	21121	93.870%	92.871409	7.128591	0.998591
2	8	6	22500	22403	99.570%	99.595426	0.404574	-0.025426
3	9	5	22500	22448	99.770%	99.772234	0.227766	-0.002234
4	10	4	22500	22479	99.910%	99.902063	0.097937	0.007937
5	13	3	22500	22499	100%	100.000	0.000	0
6	15	2	22500	22499	100%	100.000	0.000	0