On discerning strings with finite automata

Abuzer Yakaryılmaz
National Laboratory for Scientific Computing
Petrópolis, RJ, 25651-075, Brazil
Email: abuzer@lncc.br

J. Andres Montoya
Departamento de Matemáticas
Universidad Nacional de Colombia, Bogotá
Email: jamontoyaa@unal.edu.co

***Abstract*—We study the problem of discerning strings with deterministic finite state automata (DFAs, for short). We begin with a survey on the historical and algorithmic roots of this problem. Then, we focus on the maximun number of states that are necessary to separate two strings of a given length. We survey the most important results concerning this issue and we study the problem from the point of view of some alternative models of automata. The preliminary results concerning the last issue motivate us to formulate a conjecture stating that DFAs can separate any pair of strings by using a logarithmic number of states. We give some evidence supporting our conjecture.**

Let be an automaton, we say that *separates* two strings and if and only if accepts one of those two strings but rejects the other. In this work we study the problem of separating (discerning) strings by deterministic finite state automata (DFAs, for short).

Let be two binary strings. We use the symbol to denote

there exists a -state DFA separating and

and we use the symbol to denote

We focus on the maximum number of states that are necessary to separate two strings of a given length. In other words, we are interested in the behavior of function .

The study of function is related to an algorithmic problem coming from machine learning, i.e. computing the minimal finite state automaton separating two given lists of strings. It is known that the aforementioned problem is NP-hard [1], but little is known about the algorithmic hardness of its restriction to lists of size one, i.e. the algorithmic complexity of the problem of computing a minimal DFA separating two strings given as input. We argue that a good understanding of the algorithmic hardness of this restricted problem necessitates of a fine-grained knowledge of the asymptotic behavior of function Therefore, we have tried to develop a systematic analysis of function This investigation has led us to study some related problems, which are interesting in its own right.

Organization of the paper and contributions. This paper is organized into three sections. In section one we present a gentle introduction to the problem, focusing on its algorithmic and historical roots. In section three we begin the study of the maximum number of states that are necessary to separate two strings of a given length. We prove a theorem, from which most of the applications of fingerprinting, that were previously studied, can be derived as special cases. In section three we study the problem of constructing pairs of strings that are hard to separate (pairs of strings requiring a superlogarithmic number of states), to this end we consider some alternative models of automata, and we establish some first results which seem to indicate that a logarithmic number of states always suffice.

I. Introduction and Motivations

Suppose that you are Thomas A. Anderson (Neo) and you have realized that the living world is being controlled by an evil algorithm called *The Matrix*. If you want to defeat The Matrix and release the human beings, you have to know as much as possible about the structure of such a mighty algorithm. But the algorithm is a black box which can be observed only through its actions. Then, you are coping with a learning problem: the problem of learning The Matrix.

To begin with, we suppose that the Matrix is a very elementary type of algorithm, a finite state automaton. Then, at a given time instant, the actions of the algorithm that have been observed by the rebels are just two disjoint lists of binary strings, say and , such that the first one is constituted by all the strings that have been accepted by the Matrix, while the second one is constituted by all the strings that have been rejected. Thus, our problem has been reduced to compute a DFA fitting the given data. The problem, so defined, is ill posed because it is completely ambiguous: each one of the learning scenarios considered in the problem admits an infinite number of consistent and non-equivalent hypothesis. Therefore, we use *Occan Razor* to define a more interesting problem (see below).

Problem 1: (**LM**: **Learning the Matrix**)

Input: , where are binary strings.

Problem: Compute a ***minimal*** *finite state automaton that accepts each and rejects each where and*

Thus, if Neo assumes that the powerful Matrix is the most efficient algorithm fitting the given data, he needs to solve problem LM. Now we introduce a decision version of LM.

Problem 2: (**d-LM**)

Input: , where are binary strings and is a positive integer.

Problem: Decide if there exists a -state deterministic finite state automaton, say , such that for all

978-1-4673-9143-6/15/$31.00

accepts the string , while for all rejects the string .

Unfortunately (for Neo and the rebels) the above problem is NP-complete (See [1]), and it implies that LM is NP-hard. It would be a very frustrating tale, if this *impossibility result*[1] were the end of the story. We would like to try some different approaches to deal with the intractability barrier imposed by the NP-hardness of LM. There are a plenty of possibilities to try. We can ask us, for instance, whether LM can be approximated within some polynomial range. Unfortunately this is not the case: Pitt and Warmuth [2] proved that the *dfa consistency problem* (as the problem is usually called in the literature) cannot be approximated within any polynomial range. Although the situation for the rebels begins to become desperate, there are still some other approaches to try. What about studying some relevant restrictions of the problem? It is known, for instance, that the restriction to unary strings can be solved in polynomial time [3]. Which other interesting restrictions of d-LM are tractable? We have chosen to study the restriction of LM to the set of instances that are constituted by two lists of size one. That is, we have chosen to study the following algorithmic problem:

Problem 3: (**2LM**)

Input: , where are binary strings.

Problem: Compute a minimal finite state automaton accepting and rejecting

The above restriction is a natural one and it is related to some interesting problems in data compression [4]. In order to analyse the computational hardness of 2LM, we have to consider the corresponding restriction of d-LM, which we will denote as d-2LM. The proofs of NP-completeness for d-LM cannot be followed for 2-LM and, up to our knowledge, it is still open wether 2LM can be solved in polynomial time or d-2LM is NP-complete.

Let be a function solving the functional equation

Notice that if then d-2LM can be solved in polynomial time. It is the case, given that a -state DFA can be described using bits. Thus, it seems that in order to understand the computational hardness of d-2LM we must understand the behavior of function .

It is known that (see [5]). That means: a polynomial time naive brute force algorithm cannot solve d-LM, which is not an evidence supporting the possible NP-completeness of our problem. The lower bound is the best lower bound we know, and then, if such a bound is tight, d-2LM could be solved using nondeterministic bits.

If function grows slowly, d-2LM can be solved using small space, and then, it is fair to conclude that, the possibilities of d-2LM being NP-complete increases if function grows quickly. Goralcik and Koubek (see [6]) were apparently the first to study function They sketched a proof[2] that . Then, Robson proved that [7]. In the same paper Robson reported on a conjecture of Ch. Choffrut[3]: the function belongs to for all positive real number

It seems that Ch. Choffrut was the first researcher who was aware of the importance of function Choffrut's conjecture remains unsolved after more than 25 years. Note that if one proves that the problem d-2LM is NP-complete, then he is providing strong evidence against such a hard conjecture. Therefore, establishing the computational hardness of d-2LM does not seem easy, and a solution to this problem requires a better understanding of function

II. Basic facts

We start with answering why we mainly focus on binary alphabets. For is the function calculated as

there exists a -state

DFA separating and

Notice that is identical to . Goralcik and Koubek [6] proved that for all the equation holds. As usual, unary alphabets ( ) is a very special case. Given , (by overwriting) denotes

there exists a -state

DFA separating the strings and

Theorem 4: Suppose that . Then, On the other hand, there are infinitely many pairs such that

Proof: Given , we can always find a prime number such that . Thus, we can construct a DFA that simply computes the length of its input modulo , i.e. after reading the inputs and , our DFA reaches two different states. Thus, and can be separated by using states, with

In order to prove the lower bound, we use the following set:

It is easy to check that the pair cannot be separated with less than states. Note that ■

Thus, it is now clear that the binary case is representative of the cases that we do not understand.

Notice that one can use the *fingerprinting trick* used in the above proof to prove that:

Given if then

[1]The NP-hardness of LM suggests that Neo and his followers cannot solve the problem and save the world.

[2]They postponed the detailed proof for an ulterior publication but it never took place since Robson [7] provided a better bound shortly after.

[3]Goralcik and Koubek acknowledged, at the very beginning of their seminal paper, that the problem of studying function was suggested to them by Choffrut.

The above fact is the reason why, we have reduced the study of the bivariate function to the study of the univariate function

III. Looking for pairs that are hard to separate

Our main goal is to reduce the gap between lower and upper bounds for function We start with working on the lower bound side and we look for pairs that are hard to separate.

Most strings can be separated using few states. The expected number of states that are necessary to separate two randomly chosen strings is [5]. This probabilistic result implies that the hard pairs cannot be found by random search. Thus, it is necessary to understand what makes the separation a hard task. Remind that the best and non-trivial lower bound for is , which was shown by using the following infinite set where and We call them as *KG pairs* (because of Koubek and Goralcik [6]). We believe that the KG pairs are not the hardest ones to separate. The next two lemmas provide some strong evidence to our belief.

Let be a finite state automaton model and let be two binary strings. We use the symbol to denote

there exists a -state -machine separating and

and we use symbol to denote the function

We will use the symbol to denote the classes of *alternating finite state automata* (AFA, for short) [8]. Recall that a binary AFA is a quintuple, , where:

is a finite set of *existential states*.

is a finite set of *universal states*.

is the initial state.

is the set of accepting states.

, is the *transition relation* of the automaton.

Recognizing the unary language is also known as *solving counting problem* [8]. It is clear that any -state finite state machine recognizing can be modified in a straightforward way to separate two strings such that , and without any increase in the number of states. It is known that DFAs and NFAs requires states to recognize , but AFAs can recognize the same language with states (see [9], [8]).

Lemma 5: Given , we have that .

Proof: The lengths of and are exponential in and AFAs can separate them by using states. ■

The above lemma indicates that KG pairs become fairly easy to separate, when we consider the model of alternating finite state automata. In fact, it can be much easier when considering some other finite state machines. Some computational models, like probabilistic or quantum, can encode some information into their probabilities or amplitudes, and so they can very efficiently separate KG pairs. Consider the case of *Probabilistic finite automata* (PFAs, for short), those automata can separate any KG pair with bounded-error by using only 2 states. Let be a KG pair for and let be the PFA separating them. The states of are and , and is the initial and only accepting state. After reading each , stays in with probability , and switches to with the remaining probability In any other transition, stays in the same state. Thus, after reading the accepting probability will be , and, after reading , the accepting probability will be .

It could happens that AFAs can separate all the pairs of length by using states. Next lemma shows that it is not the case, next lemma shows that there exist pairs of strings requiring a very much larger number of states.

Lemma 6:

Proof: Suppose that suppose that Alice and Bob get two strings, say and suppose that they are asked to decide if those two strings are different. They must accomplish this task using the minimal amount of communication (Alice get string Bob get string and each one of them has not direct access to the string given to the other). Fortunately, they can receive some help from an all-mighty prover, who could see the two strings, and give to both of them the same advice. If and are different, the all-mighty prover could compute an AFA with states that separates those two strings Such an automaton can be described using bits, because any alternating automaton with states can be described using bits, (the transition relation can be encoded using two boolean matrices of order while the sets of universal, existential and accepting states can be encoded by three -dimensional boolean vectors). Then, given strings and , Alice and Bob can correctly decide if they are different, using bits of advice (the code of an AFA separating strings and) and one communication bit. To this end, they could use the following communication protocol:

1) Alice and Bob receive bits of advice from the prover (the same message for both of them).
2) Alice and Bob decode the message and construct the corresponding automaton, which we denote with the symbol .
3) Alice runs , on input If accepts, Alice sends to Bob, otherwise she sends
4) Bob runs , on input Then, he uses the result of this computation, and the bit he got from Alice, in order to correctly determine wether the two given strings are different.

Then, we have that Alice and Bob can compute the function EQ (the equality function) using nondeterministic bits and communicating at most one bit. Communication complexity theory says us that it is not possible [10], then cannot belong to ■

The above two lemmas show that KG pairs can be separated by alternating automata using very much fewer states than the hardest pairs. This fact can also be seen as an evidence that KG pairs also very far from being the hardest pairs for DFA's.

Question. Which are the hardest pairs for alternating automata? How many states are necessary to separate those pairs? Which pairs require states?

Remark 7: Let be a state-based model of automata, one could consider the problem d-2LM , i.e. computing a minimal -automaton separating two strings given as input. Interesting enough, the complexity of those problems seems to increase when the computational power of the class decreases. Notice that d-2LM can be solved using nondeterministic bits, and notice that d-2LM (stands for Turing machines) can be solved in polynomial time by brute-force search. On the other hand, it seems (we conjecture) that the problem d-2LM DFA is hard, and cannot be solved in polynomial time.

Fingerprinting was used to separate pairs of strings of different length (it includes the case of unary strings, see Theorem 4), using a logarithmic number of states. It can also be used to separate some other types of pairs using few states.

Let be three positive integers. The *-window of gap and preperiod* is the set where Given we use the symbol to denote the subword that is: denotes the subword of that can be observed through the -window of gap and preperiod

Let let and let be a short string (a pattern). We use the symbol to denote the quantity

That is: counts the number of times that pattern can be observed through the -window of gap and preperiod

Theorem 8: Let be two binary strings of length If those two strings can be separated using states.

Proof: Given a regular language requiring states, it is easy to construct a DFA with states that accepts the language

And then, given it is also easy to construct a DFA with states that counts modulo the number of times that pattern can be observed through the -window of gap and preperiod Thus, it is possible to construct a DFA with states, which, on input computes . Suppose that In order to separate those two strings using states, we only have to pick a prime and such that . ■

We can use Theorem 8 to obtain, as easy corollaries, two of the main results concerning this issue (applications of fingerprinting) that are consigned in the survey of Demaine et. al. [5]. First, some notation. Let and let be a short string (a pattern). We use the symbol to denote the quantity

Corollary 9: Let be two binary strings, and let be a short pattern.

1) If then and can be separated using states.
2) Let be the Hamming distance between and we have that .

Proof: To prove Item 1 it is enough to note that for all string and for all pattern the equality holds.

Now, we will prove Item 2. Let be the positions where and differ. The key fact in the proof is that it is possible to compute a prime number such that is the unique position from the set which can be observed through the -window of gap and preperiod 1. Then, it happens that . Thus, we can separate those two strings using states. ■

Now, we would like to return to the task that we have chosen at the beginning of this section: the construction of pairs that are hard to separate. To begin with, we would like to attack a modest goal: constructing an infinite set of pairs requiring states (for some) To achieve this goal, we have to take into account (we have to overcome) the restrictions imposed by Theorem 8 and its corollary, which obligate us to accomplish two seemingly opposite tasks:

We must construct pairs of words which are very similar (given a window, any pattern that can be observed through the window, should be observed the same number of times in both strings) and which are, simultaneously, very different (their Hamming distance is large).

The above task, which at first sight seems to be difficult, is quite easy indeed. Let be an integer, let and let Notice that for all such that and for all , the equality holds. Notice also that the Hamming distance between those two strings is However, those two strings can be separated using states.

Our aim (the construction of pairs that are hard to separate) is reminiscent of *The Reconstruction Problem for Sequences* [11], which we will proceed to define.

Given a string and given (with) we use the symbol to denote the quantity

That is: counts the number of times that occurs as a substring of The *-deck* of is the vector We say that the -deck *determines* if

and only if string can be unambiguously reconstructed from its -deck, that is: if for all it happens that

Or, using the terminology employed in this paper: the -deck of separates this string from any other string. Consider the function defined by

any is determined by its -deck

The Reconstruction Problem asks for determining upper and lower bounds for . Strikingly, the best known lower bound for is and the best known upper bound is ¯ (see [12] and [11]). A naive guess is that the pairs that are hard to separate using the counting of subwords are also hard to separate using DFAs. Once again intuition seems to be wrong. On one hand, the pairs of strings that are used to obtain the logarithmic lower bound for function [12] (which is very far from being trivial), can be separated using only two states, because given one of those pairs, say it happens that either (and) or (and). On the other hand, it is easy to check that KG pairs are separated by their 1-decks (because the substring occurs more times in than in). Thus, it seems that function is not really related to function

We began this section with the confidence that we could construct pairs requiring a superlogarithmic number of states. Now, it is by no means clear if such a goal can be achieved. Let us finish this paper stating a conjecture, which is very much stronger than Choffrut's conjecture.

Conjecture 10:

ACKNOWLEDGMENT

Yakaryılmaz was partially supported by CAPES with grant 88881.030338/2013-01. Moreover, the part of the research work was done while Yakaryılmaz was visiting Universidad Nacional de Colombia in 2014.
The second author would like to thank Universidad Nacional de Colombia and the support provided through the R.P. Hermes 16860.

REFERENCES

[1] M. Gold, "Complexity of automata identification from given data," *Information and control*, vol. 378, no. 3, pp. 302–320, 1978.

[2] L. Pitt and M. Warmuth, "The minimal consistent dfa problem cannot be approximated within any polynomial," *Journal of the ACM*, vol. 40, no. 1, pp. 95–142, 1993.

[3] G. Gramlich and R. Herrmann, "Learning unary automata," in *7th International Workshop on Descriptional Complexity of Formal Systems (DCFS)*, Como, Italy, Jul. 2005, pp. 122–133.

[4] J. Jonhson, "Rational equivalence relations," *Lecture Notes in computer Science*, vol. 226, pp. 167–176, 1986.

[5] E. Demaine, S. Eisenstat, J. Shallit, and D. Wilson, "Remarks on separating words," in *7th International Workshop on Descriptional Complexity of Formal Systems (DCFS)*, Vicinity of Giessen, Germany, Jul. 2011, pp. 147–157.

[6] P. Goralcik and V. Koubek, "On discerning words by automata," *Lecture Notes in Computer Science*, vol. 226, pp. 116–122, 1986.

[7] J. Robson, "Separating strings with small automata," *Information Processing Letters*, vol. 30, pp. 209–214, 1989.

[8] D. Chistikov, "Notes on counting with finite machines," in *34th International Conference on Foundation of Software Technology and Theoretical Computer Science (FSTTCS 2014)*, New Delhi, India, Dec. 2014, pp. 339–350.

[9] O. Kupferman, A. Ta-Shma, and M. Vardi, "Counting with automata," 1999.

[10] E. Kushilevitz and N. Nisan, *Communication Complexity*. New York, USA: Cambridge University Press, 1997.

[11] A. Scott, "Reconstructing sequences," *Discrete Mathematics*, vol. 175, pp. 231–238, 1997.

[12] B. Manvel, A. Meyerowitz, A. Schwenk, K. Smith, and P. Stockmeyer, "Reconstruction of sequences," *Journal of Discrete Mathematics*, vol. 94, no. 3, pp. 209–219, 1991.