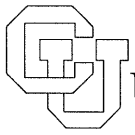
A Proof of Exponential "Permutation Complexity"*

Roldon Pozo and Karl Winklmann

CU-CS-455-90 January 1990



University of Colorado at Boulder
DEPARTMENT OF COMPUTER SCIENCE

*Work supported in part by NSF grant CCR-8702275

A proof of exponential "permutation complexity"*

Roldan Pozo and Karl Winklmann†

January 4, 1990

Tech. Rep. CU-CS-455-90

Abstract

Some nonregular languages can be recognized by a finite-state machine if we allow the machine to read not only the input string x but also a suitable collection of permutations of x. (For each input length n there is a set of permutations $\pi_1^{(n)}, \pi_2^{(n)}, \ldots, \pi_{q(n)}^{(n)}$. The same $q(n)$ permutations are applied to each input string of length n.)

We show that for some languages the necessary number of permutations is exponential in the size of the input. This "permutation complexity" is a version of time complexity in a machine model consisting of finite-state machines with permutation-generating preprocessors.

*Work supported in part by NSF grant CCR-8702275

†Authors' address: Department of Computer Science, University of Colorado at Boulder, Boulder, Colorado 80309-0430

1 Introduction

Exponential lower bounds on computational time complexity appear hard to prove: it is widely conjectured that $P \neq NP$ but no proof has been found. One possible way out of such an impasse is to try to prove exponential lower bounds in *weak models* of computation. The hope, of course, is ultimately to develop proof techniques that work in standard models.

The model we consider amounts to finite-state machines augmented by preprocessors that generate permutations of the input. We do not restrict the computational power of these preprocessors – except that they have to be "oblivious": they may generate arbitrarily complex (even noncomputable) collections of permutations of the input as a function of the input *length* but they may not inspect the actual input characters. Thus the model can solve quite complicated problems, even some that are unsolvable in standard models of computation, but it still is weak in the sense that the finite-state machines may need to look at many permutations of the input to detect properties that a stronger model could detect much more quickly.

The result is about time complexity in such a model. The proof uses standard techniques from automata theory and combinatorial lemmas about strings.

2 Permutation complexity

2.1 Examples

The language $L_1 = \{(ab)^n : n \geq 0\}$ is regular. The language $L_2 = \{a^n b^n : n \geq 0\}$ is not. However, if we presented the bits of an arbitrary string

$$x = x_1 x_2 x_3 \cdots x_{n-2} x_{n-1} x_n \in \{a, b\}^\star \tag{1}$$

to a suitable finite-state machine in the order

$$\pi(x) = x_1 x_n x_2 x_{n-1} x_3 x_{n-2} \cdots \tag{2}$$

then the machine could decide whether or not the original string x was in L_2. In this sense, L_2 is only a permutation away from being regular. We say that L_2 has a "permutation complexity" of 1. (Strictly speaking, π is not a single permutation but rather a sequence of permutations $\pi^{(n)} : \{a, b\}^n \to \{a, b\}^n, n \geq 0$. For simplicity we write $\pi(x)$ instead of $\pi^{(|x|)}(x)$.)

As another example, consider the language $L_1 \cup L_2$. Let π be the same permutation as above. Then there is a finite-state machine M with

$$x \in L_1 \cup L_2 \iff \{\pi(x), x\} \cap L(M) \neq \emptyset. \tag{3}$$

We say that $L_1 \cup L_2$ has a "permutation complexity" of 2 or less. (In this example, the second permutation is the identity.)

Other languages are more complex. Consider the language $B = \{x \in \{a,b\}^\star : x \textit{ is balanced}\}$, where a string x is called *balanced* if the number of occurrences of the character a in x differs by at most one from the number of occurrences of the character b. This language B has exponential permutation complexity: for there to exist a finite-state machine M with

$$x \in B \Longleftrightarrow \{\pi_i(x) : 1 \leq i \leq q(|x|)\} \cap L(M) \neq \emptyset, \tag{4}$$

an exponential number q of permutations is necessary and sufficient. This is the result of the present paper.

2.2 Definitions

We use standard terminology and notation from automata theory as found in any of a number of textbooks, for example [HU79,DW83,LP81]. In addition, a *permutation machine* $P =< M, \Pi >$ is defined to consist of a finite-state machine M and a collection Π of permutations $\pi_i^{(n)} : \{a,b\}^n \rightarrow \{a,b\}^n, n \geq 0$ and $1 \leq i \leq q(n)$. The function q is the *complexity of* the machine P. A string $x \in \{a,b\}^\star$ *is accepted by* P if

$$\{\pi_i^{(|x|)}(x) : 1 \leq i \leq q(|x|)\} \cap L(M) \neq \emptyset. \tag{5}$$

The *language accepted by* P is the set of all strings in $\{a,b\}^\star$ that are accepted by P.

3 An exponential lower and upper bound

Recall that a string $x \in \{a,b\}^\star$ is *balanced* if the number of occurrences of the character a in x differs by at most one from the number of occurrences of the character b.

Theorem 1 *The language $B = \{x \in \{a,b\}^\star : x \textit{ is balanced}\}$ is accepted by a permutation machine of complexity no more than $c_1 + c_2^n$, for some constants c_1 and c_2.*

Proof We consider only strings of even length. Strings of odd length can be handled very similarly.

We define a suitable collection of permutations $\pi_i^{(2n)} : \{a,b\}^{2n} \to \{a,b\}^{2n}, n \geq 0$ and $1 \leq i \leq q(2n)$, for which

$$x \in B \iff \{\pi_i^{(|x|)}(x) : 1 \leq i \leq q(|x|)\} \cap L_1 \neq \emptyset, \tag{6}$$

where $L_1 = \{(ab)^n : n \geq 0\}$. We define $\binom{2n}{n}$ permutations for each $n \geq 0$: one permutation $\pi_S^{(2n)}$ for each subset $S \subseteq \{1,\ldots,2n\}$ of size n. The permutation $\pi_S^{(2n)}$ is chosen to map S onto $\{1,3,5,\ldots,2n-1\}$ (and hence maps $\{1,\ldots,2n\} \setminus S$ onto $\{2,4,6,\ldots,2n\}$). (We choose an arbitrary permutation from among the many that satisfy this requirement.) Let x be an arbitrary balanced string of length $2n$. Then $\pi_S^{(2n)}(x) = (ab)^n$ for $S = \{i : x_i = a\}$, which proves the implication from left to right in (6). The implication from right to left is easily seen to be true because all $\pi_S^{(2n)}$ are permutations and because L_1 contains only balanced strings. □

Theorem 2 *Any permutation machine that accepts the language* $B = \{x \in \{a,b\}^\star : x \text{ is balanced}\}$ *is of complexity at least* $c_1 + c_2^n$, *for some constants* c_1 *and* c_2 *with* $c_2 > 1$.

Proof Again we consider only strings of even length.

Let $\pi_i^{(n)} : \{a,b\}^n \to \{a,b\}^n, n \geq 0$ and $1 \leq i \leq q(n)$, be a collection of permutations and M a one-way finite-state machine such that

$$x \in B \iff \{\pi_i^{(|x|)}(x) : 1 \leq i \leq q(|x|)\} \cap L(M) \neq \emptyset, \tag{7}$$

for all $x \in \{a,b\}^\star$. Let k be the number of states of M. We may assume that k is even.

A string $y \in \{a,b\}^\star$ is *k-shuffled* if in every substring of y, the number of occurrences of the character a differs by at most k from the number of occurrences of the character b.

Claim 1 *If* M *accepts* y *then* y *is balanced and* k*-shuffled.*

Proof 1. If y were not balanced then neither would be $x = (\pi_1^{(|y|)})^{-1}(y)$, contradicting (7).

2. Assume that M accepts some balanced string $y \in \{a,b\}^\star$ that is not k-shuffled. Without loss of generality we can assume that some substring w of y has $k+1$ more occurrences of the character a than of the character b. Let $t_i, i = 1,\ldots,k+1$, the length of the shortest prefix of w that has exactly i more

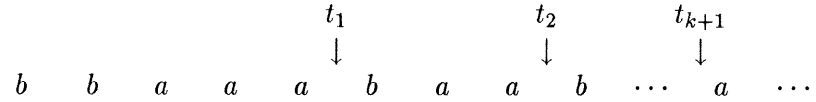


Figure 1: An example of a substring w of y in the proof of Claim 1.

occurrences of the character a than of the character b. Figure 1 illustrates this. Since M has only k states it must be in the same state at two positions t_i and t_j, for some $i < j$. By a standard pumping argument M will also accept a string y' which is obtained from y by inserting at position t_j two more copies[1] of the segment between t_i and t_j. Such a string y' is not balanced: it has $2*(j-i) > 1$ more occurrences of the character a than of the character b.

Consider the string $x' = (\pi_1^{(|y'|)})^{-1}(y')$. This string x' is unbalanced, because y' is and because $(\pi_1^{(|y'|)})^{-1}$ is a permutation. But the string $y' = \pi_1^{(|x'|)}(x')$ is accepted by M, contradicting (7). □

Claim 2 *For every balanced x, at least one of the strings $\pi_i^{(|x|)}(x)$, $1 \leq i \leq q(|x|)$, is k-shuffled.*

Proof By (7), M accepts at least one string y among $\pi_i^{(|x|)}(x)$, $1 \leq i \leq q(|x|)$. By Claim 1, such a string has to be k-shuffled. □

What remains to be shown is that there are relatively few k-shuffled balanced strings of length $2n$ when compared to all balanced strings of length $2n$. In light of Claim 2, this then implies that many permutations are needed to achieve the reduction in (7).

Claim 3 *There are $\binom{2n}{n}$ balanced strings in $\{a,b\}^{2n}$.*

Proof This is a basic fact about binomial coefficients. □

Claim 4 *For each $k \geq 1$ there is a constant $\epsilon_k > 0$ such that the number of k-shuffled balanced strings in $\{a,b\}^{2n}$ is bounded from above by $(4-\epsilon_k)^n$.*

[1] Inserting just one copy would not do because it might result in a string of odd length in which the number of occurrences of the character a differs by one from the number of occurrences of the character b — still a balanced string by our definition.

							1									
					1				1							
				1			2			1						
			1			3			3			1				
		1		4			6			4			1			
	0		5			10			10			5			0	
0		5		15			20			15			5			0
	0		20			35			35			20			0	
0		20		55			70			55			20			0
	0		75			125			125			75			0	
0		75		200			250			200			75			0
	0		⋮			⋮			⋮			⋮			0	
0		⋮		⋮			⋮			⋮			⋮			0

Figure 2: The 9-column version of Pascal's triangle.

Proof The number of all balanced strings in $\{a,b\}^{2n}$ is $\binom{2n}{n}$, which is the middle entry in the $(2n+1)$–st row of Pascal's triangle. Similarly, the number of k-shuffled balanced strings is the middle entry in the $(2n+1)$–st row of a "$(2k+1)$–column version" of Pascal's triangle: all entries outside the middle $2k+1$ columns are set to zero and each entry in the middle $2k+1$ columns is the sum of its two neighbors in the row above (with the topmost 1 being the obvious exception). Figure 2 shows the 9-column version, for $k=4$. We show that the numbers in the middle column of this $(2k+1)$–column triangle grow by no more than a factor of $4-\epsilon_k$ from one such number to the next, for some constant $\epsilon_k > 0$. (This is in contrast to the full Pascal's triangle, where this growth factor approaches a limit of 4.)

Let

$$a_k \quad a_{k-1} \quad \cdots \quad a_1 \quad a_0 \quad a_1 \quad \cdots \quad a_{k-1} \quad a_k \tag{8}$$

be the $(k+1)$–st row of the $(2k+1)$–column triangle. (This is the last row on which the $(2k+1)$–column triangle agrees with the full triangle. In the example of Figure 2 it is the row containing 1 4 6 4 1. Since we assumed k to be even, this row indeed does have form (8).) Choose $\epsilon_k > 0$ small enough so that

$$a_0 \leq (4 - \epsilon_k)^k \tag{9}$$

and

$$a_i \leq a_0 \times (1 - \epsilon_k \times 2^i), \tag{10}$$

for $i = 1, \ldots, k$. Such a constant ϵ_k exists because $a_0 = \binom{2k}{k} < 4^k$ for any $k \geq 1$ and because a_0 is larger than any other entry in the row.

Let

$$b_k \quad b_{k-1} \quad \cdots \quad b_1 \quad b_0 \quad b_1 \quad \cdots \quad b_{k-1} \quad b_k \tag{11}$$

be the numbers in the $(k+3)$–rd row, i.e., two rows below. It is straightforward to verify that

$$b_0 \leq a_0 \times (4 - \epsilon_k) \tag{12}$$

and that

$$b_i \leq a_0 \times (4 - \epsilon_k) \times (1 - \epsilon_k \times 2^i), \tag{13}$$

for $i = 1, \ldots, k$. We may replace the actual value of b_0 by its upper bound $b_0' = a_0 \times (4 - \epsilon_k)$ from (12) and recompute accordingly all the values in the triangle that depend on it. This is valid because it can only make the values larger for which we are proving upper bounds. Then condition (10) is satisfied again, with b_0' and b_i, $i = 1, \ldots, k$, in place of a_0 and a_i, $i = 1, \ldots, k$:

$$b_i \leq b_0' \times (1 - \epsilon_k \times 2^i), \tag{14}$$

for $i = 1, \ldots, k$. Inductively, the entries in the middle column do not grow by more than a factor of $4 - \epsilon_k$ from row $2n + 1$ to row $2n + 3$, for any $n \geq k/2$. Combined with (9), this proves the Claim. □

By Claim 2, each of the $\binom{2n}{n}$ balanced strings in $\{a, b\}^{2n}$ must be mapped into a k-shuffled balanced string by at least one of the permutations $\pi_i^{(2n)}$, $i = 1, \ldots, q(2n)$. Thus the number $q(2n)$ of permutations has to be at least as large as the number of balanced strings in $\{a, b\}^{2n}$ divided by the number of k-shuffled balanced strings in $\{a, b\}^{2n}$:

$$\begin{aligned} q(2n) &\geq \binom{2n}{n} \Big/ (4 - \epsilon_k)^n \\ &\geq \frac{4^n}{\sqrt{\pi n} \times (4 - \epsilon_k)^n \times (1 + O(\frac{1}{n}))} \quad . \end{aligned}$$

(The second inequality is justified by Stirling's approximation for $n!$.) This implies that $q(2n) \geq c_1 + c_2^n$, for all $n > 0$ and some constants c_1 and c_2 with $c_2 > 1$. □

4 An open question

One way to strengthen our model is to let the finite-state machine look not only at each permutation separately but to let it inspect a single string that contains all the permutations, e.g., a string of the form

$$\Diamond\Diamond\pi_1^{(n)}(x)\Diamond\pi_2^{(n)}(x)\Diamond\pi_3^{(n)}(x)\Diamond\cdots\Diamond\pi_{q(n)}^{(n)}(x)\Diamond\Diamond \tag{15}$$

where $\Diamond$ is some special character. This immediately suggests a further strengthening of the model. Instead of insisting that the string presented to the finite-state machine be of the form shown in (15), we can allow any sequence that consists of repeated occurrences of input characters and special characters. Of course, this kind of "repetition sequence" needs to be defined in an "oblivious" way, i.e., without knowledge of the actual input characters, just like the permutations in our basic model were. Can our proof be adapted to this model?

References

[DW83] Martin D. Davis and Elaine J. Weyuker. *Computability, Complexity, and Languages.* Academic Press, New York, 1983.

[HU79] J. E. Hopcroft and J. D. Ullman. *Introduction to Automata Theory, Languages, and Computation.* Addison Wesley, Reading, 1979.

[LP81] H. R. Lewis and C. H. Papadimitriou. *Elements of the Theory of Computation.* Prentice Hall, Englewood Cliffs, 1981.