Symmetry of information and nonuniform lower bounds

Sylvain Perifel

Symmetry of information and nonuniform lower bounds

Sylvain Perifel

LIP*, École Normale Supérieure de Lyon.
Sylvain.Perifel@ens-lyon.fr

December 12, 2006

Abstract. In a first part we provide another proof of the result of [4] that for all constant c, the class EXP is not included in P/n^c. The proof is based on a simple diagonalization, whereas it uses resource-bounded Kolmogorov complexity in [4]. This method enables us to show a nonuniform lower bound for BPP-consistent languages.
In the second part, we investigate links between resource-bounded Kolmogorov complexity and nonuniform classes in computational complexity. Assuming a version of polynomial-time symmetry of information, we show that exponential-time problems do not have polynomial-size circuits (in symbols, $\mathrm{EXP} \not\subset \mathrm{P/poly}$).
Keywords: computational complexity, nonuniform lower bounds, resource-bounded Kolmogorov complexity, symmetry of information

1 Introduction

Time and space hierarchy theorems have long been the main tools for separation of classes in computational complexity theory. This has provided uniform lower bounds by proving for instance that some problems solvable in exponential-time do not have polynomial-time algorithms. The lack of similar proofs for nonuniform lower bounds suggests that such simple techniques are not suitable for advice classes in the sense that the diagonalization needs in this case a too high complexity in order to be performed. This feeling is strengthen by the fact that these techniques often relativize, which hints that major open questions concerning nonuniform lower bounds won't be solved thanks to them. Yet the advantage of diagonalization is its simplicity and this paper aims at showing that it can be used even for nonuniform classes.

In particular, we are interested in the question of whether the class EXP of problems decided in exponential time has polynomial-size circuits (in symbols, whether $\mathrm{EXP} \subset \mathrm{P/poly}$). As mentionned above, the separation $\mathrm{EXP} \neq \mathrm{P}$ is

* UMR 5668 ENS Lyon, CNRS, UCBL, INRIA. Research report RR2006-50.

well-known but this nonuniform counterpart is still open. On this problem, two approaches have yielded significant results.

The first approach was to find the smallest uniform class provably not contained in P/poly. In this direction, Kannan [5] proved that $\mathrm{NEXP}^{\mathrm{NP}}$ does not have polynomial-size circuits and similar ideas enabled Schöning [11] to prove that EXPSPACE does not have polynomial-size circuits either. Here, we see that performing a diagonalization out of P/poly requires more than exponential time. Later, the second approach was to obtain the best nonuniform lower bound for EXP problems. Homer and Mocas [4] showed that EXP does not have circuits of size n^c for any fixed constant c.

The first part of the present paper provides another proof of this last result using techniques similar as [11]. An advantage of this new proof is its simplicity (it consists in a usual diagonalization whereas the original proof makes use of resource-bounded Kolmogorov complexity). As a corollary, we obtain a nonuniform lower bound on BPP-consistent sets, which can be seen somehow as a hardness result for PromiseBPP (see [3]).

In the second and main part, we show that an assumption of resource-bounded Kolmogorov complexity enables to combine both approaches described above. Namely, if polynomial-time symmetry of information holds true, then $\mathrm{EXP} \not\subset \mathrm{P/poly}$. The proof once again consists in a simple diagonalization.

Symmetry of information is a beautiful theorem in Kolmogorov complexity and can roughly be stated as follows. If x and y are two words, the sum of the quantity of information x contains about y and that contained in y equals the quantity of information contained in the concatenation xy. This theorem is due to Levin [13] and Kolmogorov [6].

When requiring polynomial time bound on the computations, however, the similar property, called polynomial-time symmetry of information, is a challenging open problem in resource-bounded Kolmogorov complexity. This problem has already been related to computational complexity by at least two results. First, Longpré and Watanabe [9] show that if $\mathrm{P} = \mathrm{NP}$ then the polynomial-time symmetry of information holds. Second, more recently and closer to our present preoccupations, Lee and Romashchenko [7] show that if polynomial-time symmetry of information holds, then $\mathrm{EXP} \neq \mathrm{BPP}$.

In this paper we use a slightly stronger version of polynomial-time symmetry of information and prove a stronger result (namely $\mathrm{EXP} \not\subset \mathrm{P/poly}$) since $\mathrm{BPP} \subset \mathrm{P/poly}$ (see [1]). Symmetry of information enables to divide advices into small blocks on which diagonalization can be performed in EXP.

All these results teach us that polynomial-time symmetry of information is a central but hard question to study. Indeed, if it holds then EXP does not have polynomial-size circuits (or at least $\mathrm{EXP} \neq \mathrm{BPP}$ with a weaker version

of symmetry of information); if it does not, then P $\neq$ NP. In both case, a fundamental question in complexity theory would find an answer.

Organization of the paper. Section 2 is devoted to definitions and notations in computational complexity and resource-bounded Kolmogorov complexity. Section 3 consists of another proof of the result of [4] that exponential-time problems do not have circuits of any fixed polynomial size n^c. A corollary is also shown there, namely a nonuniform lower bound on BPP-consistent sets.

Section 4 precisely state the hypothesis of polynomial-time symmetry of information as well as some simple results about this. Finally, Section 5 proves the main result, namely that polynomial-time symmetry of information implies that exponential-time problems do not have polynomial-size circuits.

2 Preliminaries

For references on computational complexity, we recommend the book [2]. For Kolmogorov complexity, we refer to [8]. The notions used in this paper are standard, though stated from the unifying point of view of universal Turing machines.

Universal machines. If M is a Turing machine, its boolean encoding will also be denoted by M. Therefore M is also seen as a program. The construction of a universal Turing machine is described for example in [10] and can be summed up as follows.

Proposition 1. *There exists a universal Turing machine $\mathcal{U}$, with three tapes, which, on input (M, x), simulates the two-tape machine M on input x. There is a constant $c > 0$ depending only on the machine M such that the simulation of t steps of $M(x)$ takes ct steps of $\mathcal{U}$.*

Such a universal Turing machine $\mathcal{U}$ is fixed in the remainder of the paper. The machine M simulated by $\mathcal{U}$ will also be called the *program* of $\mathcal{U}$. For instance, we will say that the program M decides the language A if for all x, the computation $\mathcal{U}(M, x)$ halts, and it accepts iff $x \in A$.

Complexity classes. If $t : \mathbf{N} \to \mathbf{N}$ is a function, the usual complexity class DTIME($t(n)$) is the set of languages A recognized in time $O(t(n))$. More precisely, $A \in$ DTIME($t(n)$) if there exist a constant $c > 0$ and a fixed program $M \in \{0,1\}^*$ such that for all word x, the computation $\mathcal{U}(M, x)$ stops before $ct(|x|)$ steps and it accepts if and only if $x \in A$. We call EXP the class $\cup_{k \geq 0}$DTIME(2^{n^k}).

Similarly, the advice class DTIME($t(n)$)/$a(n)$ is the set of languages A such that there exist a program M, a constant $c > 0$ and a family (a_n) of advices (that is to say, words) satisfying:

1. $|a_n| \leq a(n)$;
2. $\mathcal{U}(M, x, a_{|x|})$ stops in less than $ct(|x| + a(|x|))$ steps;
3. $\mathcal{U}(M, x, a_{|x|})$ accepts iff $x \in A$.

Now, P/poly is the class $\cup_{k\geq 0}\text{DTIME}(n^k)/n^k$ (i.e. polynomial working time and polynomial-size advice) and similarly, EXP/poly is the class $\cup_{k\geq 0}\text{DTIME}(2^{n^k})/n^k$ (i.e. exponential working time and polynomial-size advice). By this definition, it is easy to see that

$$\text{EXP} \subset \text{P/poly} \iff \text{EXP/poly} = \text{P/poly}.$$

Another complexity measure deals with the space needed to decide a language. Space complexity counts the number of cells used by the machine. Similarly as above, $\text{DSPACE}(s(n))$ is the set of languages A recognized in space $O(s(n))$, and advice classes are defined accordingly. The class PSPACE is $\cup_{k\geq 0}\text{DSPACE}(n^k)$.

In this paper, we shall also quickly meet the complexity class PP. This is the class of languages A such that there exist a language $B \in \text{P}$ and a polynomial $p(n)$ satisfying

$$x \in A \iff \#\{y \in \{0,1\}^{p(|x|)} \ : \ (x, y) \in B\} \geq 2^{p(|x|)-1}.$$

In other words, A is recognized by a polynomial-time probabilistic Turing machine without error.

Resource-bounded Kolmogorov complexity. For two words x, y and an integer t, we denote by $C^t(x|y)$ the time t bounded Kolmogorov complexity of x conditional to y, that is, the size of a shortest program M which, when run on the universal Turing machine $\mathcal{U}$ on input y, outputs x in time $\leq t$. For a Turing machine M (and in particular for $\mathcal{U}$), we denote by $M^t(x)$ the word written on the output tape of the machine M after t steps of computation on input x. Thus in symbols we have

$$C^t(x|y) = \min\{k \ : \ \exists M \text{ of size } k \text{ such that } \mathcal{U}^t(M, y) = x\}.$$

We will also use the notation $C^t(x)$, defined to be $C^t(x|\epsilon)$ where ϵ is the empty word.

Advice and programs. For a fixed word length n, the words $x \in \{0,1\}^n$ of size n are lexicographically ordered and the i-th one is called $x^{(i)}$ (for $1 \leq i \leq 2^n$). Let A be a language. The *characteristic string* of $A^{=n}$ is the word $\chi \in \{0,1\}^{2^n}$ defined by $\chi_i = 1$ iff $x^{(i)} \in A$.

We will often consider programs that output characteristic strings rather than programs that decide languages. We rely on the following obvious lemma.

Lemma 1. *If A is a language in* DTIME$(t(n))/a(n)$ *(where $t(n) \geq n$), then there exist constants $\alpha, k > 0$ and a family (M_n) of programs satisfying:*

1. $|M_n| \leq k + a(n)$;
2. *for $1 \leq i \leq 2^n$ in binary, $\mathcal{U}(M_n, i)$ outputs the i first bits of the characteristic string χ of $A^{=n}$ in time $\alpha it(n + a(n))$.*

Proof. Let M be a DTIME$(t(n))$ machine deciding A with advice of size $a(n)$. The program M_n merely enumerates the i first words x of size n and simulates $M(x)$: M_n is therefore composed of the code of M, of an enumeration routine for the i first words of size n and of the advice for the length n. □

3 Diagonalizing out of n^c advice length

We provide another proof of the following proposition of [4]. The initial proof of [4] makes use of resource-bounded Kolmogorov complexity whereas it consists here in a usual diagonalization. This is indeed similar to the proof of Schöning [11] that EXPSPACE does not have polynomial-size circuits (see [2, Th. 5.6]): at each step of the diagonalization process, we eliminate half of the possible programs. This method enables us to prove a kind of hardness result on PromiseBPP problems as a corollary.

Proposition 2. *For all constants $c_1, c_2 \geq 1$, there is a sparse language A in* DTIME$(2^{O(n^{1+c_1c_2})})$ *but not in* DTIME$(2^{O(n^{c_1})})/n^{c_2}$.

Proof. Let us define $A^{=n}$ for all n, therefore fix n. Recall that $x^{(1)} < x^{(2)} < \ldots < x^{(2^n)}$ are the words of $\{0,1\}^n$ sorted in lexicographic order. We will diagonalize over the programs M of size at most $n + n^{c_2}$ (of which there are $2^{n+n^{c_2}+1} - 1$), and the universal machine $\mathcal{U}$ will be simulated for $t(n) = 2^{n^{1+c_1c_2}}$ steps. The set $A^{=n}$ is defined word by word as follows:

$$x^{(1)} \in A^{=n} \iff \begin{array}{l}\text{for at least half of the programs } M \text{ of size } \leq n + n^{c_2},\\ \text{the first bit of } \mathcal{U}^{t(n)}(M) \text{ is 0,}\end{array}$$

that is, at least half of the programs give a wrong answer for $x^{(1)}$. Let V_1 be the set of programs M giving the right answer for $x^{(1)}$, i.e. such that the first bit of $\mathcal{U}^{t(n)}(M)$ corresponds to "$x^{(1)} \in A$". Hence $|V_1| < 2^{n+n^{c_2}}$ (less than half of the programs of size $\leq n + n^{c_2}$ remain). We then go on with $x^{(2)}$:

$$x^{(2)} \in A^{=n} \iff \begin{array}{l}\text{for at least half of the programs } M \in V_1,\\ \text{the second bit of } \mathcal{U}^{t(n)}(M) \text{ is 0,}\end{array}$$

that is, among the programs that were right for $x^{(1)}$, at least half make a mistake for $x^{(2)}$. Let V_2 be the set of programs $M \in V_1$ giving the right answer for $x^{(2)}$. We go on like this:

$$x^{(i)} \in A^{=n} \iff \begin{array}{l}\text{for at least half of the programs } M \in V_{i-1},\\ \text{the } i\text{-th bit of } \mathcal{U}^{t(n)}(M) \text{ is } 0\end{array}$$

until V_i is empty. Call k the first i such that $V_i = \emptyset$. We decide arbitrarily that $x^{(j)} \notin A^{=n}$ for $j > k$. Note that $k \leq n + n^{c_2} + 1$ because $|V_i|$ is halved at each step, therefore A is sparse.

If $A \in \mathrm{DTIME}(2^{O(n^{c_1})})/n^{c_2}$, then by Lemma 1 there would be a constant k and a family (M_n) of programs of size $\leq k+n^{c_2}$ writing down the characteristic string of $A^{=n}$ in time $\alpha(n + n^{c_2} + 1)2^{O(n^{c_1 c_2})} \leq 2^{\beta n^{c_1 c_2}}$ for some β. This is not possible as soon as $n \geq k$ and $t(n) > 2^{\beta n^{c_1 c_2}}$ since all programs of size $n + n^{c_2}$ must make a mistake on some input of size n. Therefore $A \notin \mathrm{DTIME}(2^{O(n^{c_1})})/n^{c_2}$.

Now, in order to decide if $x^{(i)} \in A$ it is enough to decide if $x^{(j)} \in A$ for all $j \leq i$. This is done in the order $j = 1, \ldots, i$ because we need the answer of j for $j + 1$. For $x^{(j)}$ we proceed as follows: we enumerate all the programs M of size $\leq n + n^{c_2}$, compute $\mathcal{U}^{t(n)}(M)$ by simulating $\mathcal{U}$ for $t(n)$ steps, we test whether $M \in V_{j-1}$ (this is done by comparing for each $k < j$ the k-th bit of $\mathcal{U}^{t(n)}(M)$ with the already computed value of "$x^{(k)} \in A$"), and count how many $M \in V_{j-1}$ produce an output whose j-th bit is 0. If there are more than half such M, then $x^{(j)} \in A$, otherwise $x^{(j)} \notin A$. The overall running time of this algorithm is $(n + n^{c_2})2^{O(n^{c_2})}t(n)$, thus $A \in \mathrm{DTIME}(2^{O(n^{1+c_1 c_2})})$. □

The same proof also works for space complexity.

Proposition 3. *For all constants $c_1, c_2 \geq 1$, there is a sparse language A in* $\mathrm{DSPACE}(n^{1+c_1 c_2})$ *but not in* $\mathrm{DSPACE}(n^{c_1})/n^{c_2}$.

The following corollary is now immediate.

Corollary 1. *For every constant $c > 0$,* $\mathrm{EXP} \not\subset (\mathrm{P}/n^c)$ *and* $\mathrm{PSPACE} \not\subset (\cup_k \mathrm{DSPACE}(\log^k n)/n^c)$.

The construction of the language A in the proof of Proposition 2 in fact enables us to prove a "hardness result" for PromiseBPP problems. We won't define precisely PromiseBPP but just explain the notion of BPP-consistency; see for instance [3] for further details.

Definition 1. *Let N be a probabilistic Turing machine. We say that the language A is* BPP*-consistent with N if the following conditions hold:*

– *if* $N(x)$ *accepts with probability* $\geq 2/3$ *then* $x \in A$*;*
– *if* $N(x)$ *rejects with probability* $\geq 2/3$ *then* $x \notin A$*.*

Remark that we don't impose anything in the case where $N(x)$ *accepts with probability* $1/3 < \alpha < 2/3$*.*

We now obtain a lower bound on BPP-consistent languages in the sense of the following proposition.

Proposition 4. *Let* $k > 0$*. There exists a polynomial-time probabilistic Turing machine* N *such that no language* $L \in \mathrm{DTIME}(n^k)/(n - \log n)$ *is* BPP*-consistent with* N*.*

Proof. In the definition of the language A in the proof of Proposition 2, deciding whether "for at least half of the programs $M \in V_{i-1}$, the i-th bit of $\mathcal{U}^{t(n)}(M)$ is 0" is a PP problem (let us call it C) as soon as the simulation time $t(n)$ of M is polynomial. Let us take $t(n) = n^{3+k}$, diagonalize over programs of size $\leq n + \log n$ (of which there are less than $n2^{n+1}$) and call N the polynomial-time probabilistic machine that decides the above language $C \in \mathrm{PP}$. Hence $A \notin \mathrm{DTIME}(n^{2+k})/n$.

Suppose that there exists a language $L \in \mathrm{DTIME}(n^k)/(n - \log n)$ which is BPP-consistent with N. As for BPP, one can reduce the probability of error by repeating the whole computation several times and taking the most frequent answer (this can be proved using Chernoff bounds). With a number of repetitions of $O(n)$, we obtain a language $B \in \mathrm{DTIME}(n^{1+k})/(n - \log n)$ such that for all x of size n:

– if $N(x)$ accepts with probability $\geq 1/2 + 2^{-n}$ then $x \in B$;
– if $N(x)$ rejects with probability $\geq 1/2 + 2^{-n}$ then $x \notin B$.

Let us replace the PP problem C in the definition of the language A of Proposition 2 by B, yielding a new language A'. At each step of the definition of A', we discard at least a $(1/2 - 2^{-n})$ fraction of the programs (instead of $1/2$ in the original definition of A). Therefore, after the i-th step, the number of remaining programs (that is, $|V_i|$) is $|V_i| \leq n2^{n+1}(1/2 + 2^{-n})^i$. Hence $n + 2 + \log n$ steps are enough for completing the diagonalization process because $n2^{n+1}(1/2 + 2^{-n})^{n+2+\log n} < 1$ for large enough n.

Note that if we have B as oracle, we can decide A' with $n + 2 + \log n$ queries to B on words of size $n + 2 + \log n$ (the history of the answers of B). Since $B \in \mathrm{DTIME}(n^{1+k})/(n - \log n)$ we have $A' \in \mathrm{DTIME}(n^{2+k})/n$ which is a contradiction. □

Since for any probabilistic polynomial-time Turing machine M, the class PP provides languages BPP-consistent with M, the preceding proposition therefore shows that PP is not in $\mathrm{DTIME}(n^k)/(n - \log n)$ for all k. This is to be compared with the stronger result of [12] that PP does not have circuits of size n^k for any fixed k.

4 Symmetry of information

In this section we state the hypothesis of resource-bounded symmetry of information we will use. For the sake of completeness, we first state a version of symmetry of information for exponential time bounds (the time bounds here are not meant to be optimal). For a proof one can easily adapt the unbounded case, see for instance [8, Th. 2.8.2 p. 182].

Theorem 1. *There exist constants $\alpha, \beta \geq 1$ such that for all words x, y and all $t \geq 2^{3(|x|+|y|)}$, the following equality holds:*

$$C^t(x, y) \geq C^{\alpha t^2}(x) + C^{\alpha t^2}(y|x) - \beta \log(|x| + |y|).$$

We now precisely state the version of polynomial-time symmetry of information we are going to use. As mentioned before, this is a slightly stronger version than the usual one since the time bound in the right-hand side is usually $q(p(n))$ instead of $q(n)p(n)$. Note that we state and use only one direction of the symmetry of information (the hard one).

(SI) There exist a constant $\beta > 0$ and a polynomial q such that for all polynomial p and all words x, y, z of size $|x| + |y| + |z| = n$,

$$C^{p(n)}(x, y|z) \geq C^{p(n)q(n)}(x|z) + C^{p(n)q(n)}(y|x, z) - \beta \log n.$$

The following lemma investigates the consequences of iteratively applying symmetry of information: note in particular that the initial error $\beta \log n$ becomes rather big but can still be small in comparison with nk.

Lemma 2. *Suppose* (SI) *holds and take a corresponding polynomial q. Let $u_1, \ldots, u_n$ be words of size s and z another word of arbitrary size. We define $m = ns + |z|$ the size of all these words. Let p be a polynomial and $t \geq p(m)$. Suppose there exists a constant k such that for all $j \leq n$, we have $C^{q(m)^{\log n} t}(u_j|u_1, \ldots, u_{j-1}, z) \geq k$.*

Then $C^t(u_1, \ldots, u_n|z) \geq nk - (n-1)\beta \log m$.

Proof. We show the result by induction on n. This is clear for $n = 1$. For $n > 1$, by (SI),

$$C^t(u_1, \ldots, u_n|z) \geq C^{tq(m)}(u_1, \ldots, u_{n/2}|z) + C^{tq(m)}(u_{n/2+1}, \ldots, u_n|u_1, \ldots, u_{n/2}, z) - \beta \log m.$$

By induction hypothesis at rank $n/2$, the right-hand side is at least $2((n/2)k - (n/2-1)\beta \log m) - \beta \log m = nk - (n-1)\beta \log m$. □

Let us now establish the link between Kolmogorov complexity of the characteristic string of a set and the length of the advice needed to recognize this set.

Lemma 3. *Let A be a language and $\chi^{(n)}$ the characteristic string of $A^{=n}$ (i.e. $\chi_i^{(n)} = 1$ iff $x^{(i)} \in A^{=n}$). We denote by $\chi^{(n)}[1..i]$ the string consisting of the i first bits of $\chi^{(n)}$. Let $r(n)$ be a function and suppose that there exist an unbounded function $s(n) \geq 0$ such that for all constant $\alpha > 0$, there exist infinitely many n and $1 \leq i \leq 2^n$ satisfying*

$$C^{\alpha ir(n+a(n))}(\chi^{(n)}[1..i]) > s(n) + a(n).$$

Then $A \notin \mathrm{DTIME}(r(n))/a(n)$.

Proof. Suppose that $A \in \mathrm{DTIME}(r(n))/a(n)$. Then there exist a fixed program M together with an advice function $c(n)$ of size $\leq a(n)$, such that for all $x \in \{0,1\}^n$, $\mathcal{U}(M, x, c(n))$ works in time $O(r(n + a(n)))$ and accepts iff $x \in A^{=n}$. By enumerating the first i words of size n in lexicographic order and simulating the program M on each of them, there is another program N with advice $c(n)$ that enumerates $\chi^{(n)}[1..i]$ in time $O(ir(n + a(n)))$. Hence there exists a constant $\alpha > 0$ such that for all n and for all $i \leq 2^n$, $C^{\alpha ir(n+a(n))}(\chi^{(n)}[1..i]) \leq |N| + a(n)$. □

5 Diagonalizing out of polynomial advice length

We are now interested in diagonalizing over all polynomial advices, not just of size n^c for some fixed c. We will use the hypothesis of symmetry of information above. Here the main difficulty is to diagonalize over all advices without enumerating them, otherwise we would go outside of EXP. The idea is simple and can be summed up as follows.

– Two different "useful" and "independent" parts of size k_1 and k_2 of an advice "must" carry roughly $k_1 + k_2$ bits of information. This is where symmetry of information comes into play.

– We therefore decompose the advice in small blocks (of size $O(n)$) and diagonalize over the blocks instead of the whole advice, while making sure that these blocks are "independent".

Theorem 2. *If polynomial-time symmetry of information* (SI) *holds true, then*

$$\mathrm{EXP} \not\subset \mathrm{P/poly}.$$

Proof. Suppose (SI) holds true: this gives a corresponding polynomial q. We diagonalize over programs ("blocks") M of length $\leq n-1$ and simulate the universal machine $\mathcal{U}$ for $t(n) = q(n^{1+\log n})^{\log^2 n} n^{2+\log n+\log^3 n}$ steps. Define the language A as follows, by length as in the proof of Proposition 2. Remark that the n first steps of the definition are the same as for Proposition 2; the difference occurs only after, when we reuse the initial segment of $A^{=n}$ in our simulation.

$$x^{(1)} \in A^{=n} \iff \begin{array}{l}\text{for at least half of the programs } M \text{ of size } \leq n-1,\\ \text{the first bit of } \mathcal{U}^{t(n)}(M) \text{ is } 0,\end{array}$$

that is, at least half of the programs M of length $\leq n-1$ give a wrong answer for $x^{(1)}$. Let V_1 be the set of programs M giving the right answer for $x^{(1)}$, i.e. such that the first bit of $\mathcal{U}^{t(n)}(M)$ corresponds to "$x^{(1)} \in A$". Hence $|V_1| < 2^{n-1}$ (less than half of the programs of size $\leq n-1$ remain). We then go on with $x^{(2)}$:

$$x^{(2)} \in A^{=n} \iff \begin{array}{l}\text{for at least half of the programs } M \in V_1,\\ \text{the second bit of } \mathcal{U}^{t(n)}(M) \text{ is } 0,\end{array}$$

that is, among the programs that were right for $x^{(1)}$, at least half make a mistake for $x^{(2)}$. Let V_2 be the set of programs $M \in V_1$ giving the right answer for $x^{(2)}$. We go on like this:

$$x^{(i)} \in A^{=n} \iff \begin{array}{l}\text{for at least half of the programs } M \in V_{i-1},\\ \text{the } i\text{-th bit of } \mathcal{U}^{t(n)}(M) \text{ is } 0,\end{array}$$

until V_i is empty. Call k the first i such that $V_i = \emptyset$: $k \leq n$ since $2^n - 1$ programs M were to be tested.

Call $u^{(1)}$ the characteristic string of the initial segment of size n of $A^{=n}$ defined above: thus $|u^{(1)}| = n$ and for $i \leq n$, $u_i^{(1)} = 1$ if and only if $x^{(i)} \in A^{=n}$. We now define the next segment of size n of $A^{=n}$.

$$x^{(n+1)} \in A^{=n} \iff \begin{array}{l}\text{for at least half of the programs } M \text{ of size } \leq n-1,\\ \text{the first bit of } \mathcal{U}^{t(n)}(M, u^{(1)}) \text{ is } 0,\end{array}$$

that is, at least half of the programs M of length $\leq n-1$ give a wrong answer for $x^{(n+1)}$ even with the string $u^{(1)}$ as advice. Let V_1 be the set of programs M giving the right answer for $x^{(n+1)}$, i.e. such that the first bit of $\mathcal{U}^{t(n)}(M, u^{(1)})$ corresponds to "$x^{(n+1)} \in A$". Hence $|V_1| < 2^{n-1}$. We then go on with $x^{(n+2)}$:

$$x^{(n+2)} \in A^{=n} \iff \begin{array}{l}\text{for at least half of the programs } M \in V_1,\\ \text{the second bit of } \mathcal{U}^{t(n)}(M, u^{(1)}) \text{ is 0,}\end{array}$$

that is, among the programs that were right for $x^{(n+1)}$, at least half make a mistake for $x^{(n+2)}$. And we go on like this:

$$x^{(n+i)} \in A^{=n} \iff \begin{array}{l}\text{for at least half of the programs } M \in V_{i-1},\\ \text{the } i\text{-th bit of } \mathcal{U}^{t(n)}(M, u^{(1)}) \text{ is 0.}\end{array}$$

Call $u^{(2)}$ the characteristic string of the second segment of size n of $A^{=n}$ defined above: thus $|u^{(2)}| = n$ and $u^{(2)}_i = 1$ if and only if $x^{(n+i)} \in A^{=n}$. We define the third segment of size n of $A^{=n}$ analogously:

$$x^{(2n+1)} \in A^{=n} \iff \begin{array}{l}\text{for at least half of the programs } M \text{ of size } \leq n-1,\\ \text{the first bit of } \mathcal{U}^{t(n)}(M, u^{(1)}, u^{(2)}) \text{ is 0,}\end{array}$$

etc. Going on like this we have (for the $(j+1)$-th segment):

$$x^{(jn+i)} \in A^{=n} \iff \begin{array}{l}\text{for at least half of the programs } M \in V_{i-1},\\ \text{the } i\text{-th bit of } \mathcal{U}^{t(n)}(M, u^{(1)}, u^{(2)}, \ldots, u^{(j)}) \text{ is 0.}\end{array}$$

We stop when $j = n^{\log n}$ and decide arbitrarily that $x^{(k)} \notin A$ for $k > n \times n^{\log n}$.

Let us first show that $A \notin \mathrm{P/poly}$. Note that at each step of the definition of A, we have $C^{t(n)}(u^{(j)}|u^{(1)} \cdots u^{(j-1)}) \geq n-1$ because no program of length $\leq n-1$ writes $u^{(j)}$ in time $\leq t(n)$ on input $u^{(1)} \cdots u^{(j-1)}$. Since (SI) holds and by definition of $t(n)$, Lemma 2 asserts that for $i = n \times n^{\log n}$, $C^{nin^{\log^3 n}}(\chi^{(n)}[1..i]) \geq (n-1) \times n^{\log n} - n^{\log n}\beta \log i \geq n^{\log n}$ for large enough n.

Hence by Lemma 3, if we let $a(n) = n^{\log n} - n$ and $r(n) = n^{\log n}$, we have $A \notin \mathrm{DTIME}(r(n))/a(n)$. In particular, $A \notin \mathrm{P/poly}$.

It is straightforward to see that $A \in \mathrm{EXP}$, and the theorem follows. □

Remark – The same proof also works for space complexity if we assume the corresponding version of symmetry of information for polylogarithmic space bounded Kolmogorov complexity. That is, under this assumption we can prove that $\mathrm{PSPACE} \not\subset (\cup_k \mathrm{DSPACE}(\log^k n)/\mathrm{poly})$.

6 Further research

It would be interesting to prove that the usual version of polynomial-time symmetry of information (with $p(q(n))$ time bound instead of $p(n)q(n)$) also implies EXP $\not\subset$ P/poly. This might be useful for proving unconditional results by using the same techniques for CAMD complexity (see [7]), since polynomial-time symmetry of information would hold for CAMD under the assumption EXP $\subset$ P/poly (which implies EXP = AM).

The author wants to thank Andrei Romashchenko for the useful and numerous discussions on Kolmogorov complexity (in particular on symmetry of information) and also Pascal Koiran for pointing out the open problem "EXP $\subset$ P/poly?".

References

1. L. M. Adleman. Two theorems on random polynomial time. In *Proceedings of the 19th IEEE Symposium on Foundations of Computer Science*, pages 75–83, October 1978.
2. J. L. Balcázar, J. Díaz, and J. Gabarró. *Structural Complexity I.* Number 11 in EATCS monographs on theoretical computer science. Springer-Verlag, 1988.
3. H. Buhrman and L. Fortnow. One-sided versus two-sided error in probabilistic computation. In *Proceedings of the 16th Symposium on Theoretical Aspects of Computer Science*, volume 1563 of *Lecture Notes in Computer Science*, pages 100–109. Springer, 1999.
4. S. Homer and S. Mocas. Nonuniform lower bounds for exponential time classes. In *Mathematical Foundations of Computer Science, 20th symposium*, volume 969 of *Lecture Notes in Computer Science*, pages 159–168, 1995.
5. R. Kannan. Circuit-size lower bounds and non-reducibility to sparse sets. *Information and Control*, 55:40–56, 1982.
6. A. Kolmogorov. Combinatorial foundations of information theory and the calculus of probabilities. *Russian Mathematical Surveys*, 38(4):29–40, 1983.
7. T. Lee and A. Romashchenko. Resource bounded symmetry of information revisited. *Theoretical Computer Science*, To appear. Earlier version in 29th Symposium on the Mathematical Foundations of Computer Science, 2004.
8. M. Li and P. Vitányi. *An introduction to Kolmogorov complexity and its applications.* Graduate texts in computer science. Springer, second edition, 1997.
9. L. Longpré and O. Watanabe. On symmetry of information and polynomial time invertibility. *Information and Computation*, 121(1):14–22, August 1995.
10. C. H. Papadimitriou. *Computational Complexity.* Addison-Wesley, 1994.
11. U. Schöning. *Complexity and structure*, volume 211 of *Lecture Notes In Computer Science.* Springer-Verlag, 1985.
12. N. V. Vinodchandran. A note on the circuit complexity of PP. In *Electronic Colloquium on Computational Complexity, Report No. 56*, July 2004.
13. A. Zvonkin and L. Levin. The complexity of finite objects and the development of the concepts of information and randomness by means of the theory of algorihms. *Russian Mathematical Surveys*, 25(6):83–124, 1970.