Electronic Colloquium on Computational Complexity, Report No. 64 (2004)

ECCC

Exponential time reductions and sparse languages in **NEXP**

Piotr Faliszewski *

June 25, 2004

Abstract

In this paper we define a many-one reduction which is allowed to work in exponential time but may only output polynomially many symbols. We show that there are no **NEXP**-hard sparse languages under our reduction unless **EXP** = **UEXP**.

Introduction

Sparse languages play a central role in complexity theory. One of the standard questions regarding complexity classes is wether they have sparse hard/complete sets with respect to some reducibility type or what consequences would follow if they had. For example Mahaney [Mah82] showed that there are no **NP**-hard languages unless **P** = **NP**. The same was done for bounded truth table reductions by Ogiwara and Watanabe [OW91]. Their result also gives a simpler proof of the Mahaney's Theorem. The case of Turing reductions was resolved by Karp and Lipton [KL82]. Many other beautiful results regarding sparse sets were obtained – see [CO97] for a survey and more references.

In this paper we analyze the existence of **NEXP**-hard sparse languages with respect to a certain exponential time many-one reducibility. The reduction procedure is allowed to work in exponential time but may only output polynomially many symbols. Our main result says that if there are sparse **NEXP**-hard languages with respect to this type of reduction then **EXP** = **UEXP**. The natural question is wether one can strengthen this to **EXP** = **NEXP**.

The definition of the exponential time reduction was motivated by the observation that in the polynomial time setting the power of the polynomial-time many-one reductions is comparable with the power of the class **P** whereas if we consider **EXP** and **NEXP** then the power of the reduction is much smaller than that of **EXP**. The introduction of the exponential time reductions makes the situation more symmetric. Another profit that might come from the exponential time reductions is that it seems reasonable to expect that there are languages in **NEXP** complete with respect to $\leqslant_m^e$ but not with respect to standard reductions. Clearly, if **EXP** $\neq$ **NEXP** then such sets would be **NEXP**-intermediate (we consider a set **NEXP**-intermediate if it belongs to **NEXP** $-$ **EXP** but is

*Institute of Computer Science, AGH University of Science and Technology, al. Mickiewicza 30, Kraków, Poland, email: pfali@agh.edu.pl

ISSN 1433-8092

not complete with respect to the polynomial time many-one reduction). Exploring properties of intermediate languages is interesting as on one hand there are no known natural intermediate languages in many classes and on the other we strongly suspect their existence (see e.g. [Lad75, BD82]).

1 Preliminaries

We use standard notation, e.g. as in [Pap94, HO02], whenever possible. Without the loss of generality we assume that all languages are over the alphabet $\Sigma = \{0, 1\}$. If L is a language then $L^{\leqslant n} = \{x \in L : |x| \leqslant n\}$. The language L is called sparse if $|L^{\leqslant n}|$ is bounded by some polynomial.

The classes **EXP** and **NEXP** contain problems solveable in exponential deterministic and nondeterministic time respectively (polynomials in the exponents). The class **UEXP** is the set of languages decidable in exponential time by unambiguous nondeterministic Turing machines. A nondeterministic machine is called unambiguous if on any input at most one computation path accepts. The relation between **EXP** and **UEXP** is the same as between **P** and **UP**.

We assume that every nondeterministic Turing machine considered has the following property. At every computation step the machine has at most two nondeterministic choices. Every nondeterministic Turing machine can easily be converted to satisfy this requirement; every multiple choice can be split into many binary ones. Therefore, if some computation path of the machine M on input x requires n steps then this path can be encoded as a binary string of length n where zeroes and ones represent the nondeterministic decisions made by the machine.

2 Main result

We start with the formal definition of the exponential time many-one reduction that we will use in this paper.

Definition 1 *We say that a language A exponentially many-one reduces to the language B, denoted as $A \leqslant_m^e B$, if there is a function $f : \Sigma^* \to \Sigma^*$ such that:*

- $x \in A \Leftrightarrow f(x) \in B$.
- *$f(x)$ is computable in exponential time with respect to $|x|$;*
- *$|f(x)|$ is polynomial with respect to $|x|$;*

The restriction on the number of symbols that the reduction may output was given in order to make the reduction transitive. The following theorem states basic properties of the reduction.

Theorem 2 *It holds that:*

1. $A \leqslant_m^p B \Rightarrow A \leqslant_m^e B$;
2. $A \leqslant_m^e B$ *does not necessarily imply* $A \leqslant_m^p B$;
3. $(A \leqslant_m^e B \wedge B \leqslant_m^e C) \Rightarrow A \leqslant_m^e C$.

Proof: Property 1 is obviously true as every polynomial-time many-one reduction is also an exponential-time reduction.

Property 2 holds because every language in **EXP**, except for the pathological cases of $\emptyset$ and Σ^*, is complete with respect to the exponential reduction whereas this is not true for the polynomial time reductions.

Finally, we will show that the reduction is transitive. Let A, B, C be some languages and f_1, f_2 be functions witnessing that $A \leqslant_m^e B$ and $B \leqslant_m^e C$. Let p_1 and p_2 be two polynomials such that it holds that on input x f_i halts after at most $2^{p_i(|x|)}$ steps and outputs at most $p_i(|x|)$ symbols. It is now enough to show that $g(x) = f_2(f_1(x))$ exponentially reduces A to C. Clearly, $x \in A$ if and only if $h(x) \in C$. Since $|f_1(x)| \leqslant p_1(|x|)$ we have a polynomial bound on $|h(x)|$ as $|h(x)| = |f_2(f_1(x))| \leqslant p_2(p_1(|x|))$. Computing $h(x)$ takes at most $2^{p_1(x)} + 2^{p_2(p_1(x))}$ which is exponential in $|x|$. Therefore, $A \leqslant_m^e C$. ■

Our main result is similar to the Mahaney's Theorem; however, it works in the exponential time setting.

Theorem 3 *If* **EXP** $\neq$ **UEXP** *then there are no sparse* **NEXP***-hard languages with respect to the exponential many-one reduction.*

Proof: We will do a proof by contradiction. Assume that there is a sparse language T such that T is **NEXP**-hard with respect to the exponential many-one reductions. Let L be an arbitrary language from **UEXP** and M_L a nondeterministic unambiguous Turing machine that witnesses $L \in$ **UEXP**. We will show a deterministic exponential time algorithm for L – this will be a contradiction as we assumed **EXP** $\neq$ **UEXP**. Define L' to be the language:

$$L' = \{\langle x, y, b\rangle \quad : \quad x \in L, b \in \{0,1\} \text{ is the } y\text{'th nondeterministic decision that } M_L \text{ makes during its work on the string } x\}.$$

The number y is encoded in binary. It is easy to see that $L' \in$ **NEXP** (in fact, it is in **UEXP**) and therefore it is exponentially reducible to T. Let f be the exponential time reduction witnessing that $L' \leqslant_m^e T$.

We give a description of an algorithm that decides L in exponential time provided it has access to the oracle T. Later on we will show how the oracle can be removed without increasing the time complexity of the algorithm beyond exponential time. Let x be the input string and $n = |x|$. Without the loss of generality we assume that on strings of length n the machine M_L makes exactly $2^{p(n)}$ nondeterministic choices on every computation path.

```
TEST(x)
begin
    w := ε;
    for y := 1 to 2^{p(n)} do
        if f(⟨x, y, 0⟩) ∈ T then w := w0;
        else w := w1;
    Accept if and only if w is an accepting computation path of M_L on x
end
```

The TEST procedure checks wether $x \in L$ provided a T oracle is available. If $x \in L$ then there is only one accepting computation path of M_L and the procedure finds it bit by bit by querying L' and verifying the answer with the T

oracle. At the end, the computation path w is checked. Clearly, this test passes if and only if $x \in L$ – the TEST procedure correctly decides L.

Let us analyze the time complexity of the algorithm excluding the cost of T membership testing. Each iteration of the *for* loop requires computing $f(\langle x, y, 0\rangle)$. We have that $|y| \leqslant p(n)$ so, by the definition of the exponential time reduction, $f(\langle x, y, 0\rangle)$ can be computed in exponential time. There exists a polynomial q such that $|f(\langle x, y, 0\rangle)| \leqslant q(n)$. There are exponentially many iterations so executing the whole loop also takes exponential time. Finally, the test wether w is an accepting path of M_L on x can be performed in exponential time as well (precisely in time $2^{O(p(n))}$).

It remains to show how the T oracle can be removed. Firstly, note that if we would run the TEST procedure on some $x \notin L$ then regardless of the oracle answers the algorithm would reject as the test in the last line would fail.

Secondly, recall that the queries to the oracle are never longer than $q(|x|)$. Therefore, if we ran the procedure on input x with the oracle $T^{\leqslant q(|x|)}$ then it would give the correct answer. These observations enable us to remove the oracle T from the algorithm in the following way.

Let r be a polynomial such that $|T^{\leqslant n}| \leqslant r(n)$ for all natural n. Such a polynomial must exists as T is sparse. For the given input string x consider all the languages T_i with the following properties:

- $|T_i| \leqslant r(q(|x|))$;
- $y \in T_i \Rightarrow |y| \leqslant q(|x|)$.

There are only exponentially many such sets and they can easily be enumerated. We execute the TEST procedure with every T_i as oracle and accept if and only if at least one of the runs does.

There must be an index j such that $T_j = T^{\leqslant q(|x|)}$. Therefore, if $x \in L$ then the TEST procedure executed with T_j as oracle must accept and as a consequence the whole algorithm accepts. On the other hand, if $x \notin L$ then the TEST procedure rejects regardless of the oracle so the whole algorithm rejects. The algorithm still runs in exponential time as one merely needs to generate T_i's and execute TEST(x) exponentially many times. Multiplying exponential functions gives an exponential function.

Therefore, we have shown a deterministic exponential time algorithm for an arbitrary language from **UEXP**. As a result it holds that **EXP** = **UEXP**, contradicting our assumptions. The theorem is proved. ■

A natural question that arises is wether a stronger version of the theorem holds. That is, wether the existence of sparse **NEXP** hard languages with respect to the exponential time reductions implies **EXP** = **NEXP**.

References

[BD82] J.L. Balcazar and J. Diaz. A note on a theorem by Ladner. *Information Processing Letters*, 1982.

[CO97] Jin-Yi Cai and Mitsunori Ogihara. Sparse sets versus complexity classes. In *Complexity Theory Retrospective II*, pages 53–80. Springer-Verlag, 1997.

[HO02] Lane A. Hemaspaandra and Mitsunori Ogihara. *The Complexity Theory Companion*. Springer-Verlag, 2002.

[KL82] Richard Karp and Richard Lipton. Turing machines that take advice. *L'enseignement Mathématique*, 28(3/4):191–209, 1982.

[Lad75] Richard E. Ladner. On the structure of polynomial time reducibility. *Journal of the ACM (JACM)*, 22(1):155–171, 1975.

[Mah82] Michael R. Mahaney. Sparse complete sets for np: solution of a conjecture of berman and hartmanis. *Journal of Computer System Science*, 25, 1982.

[OW91] Mitsunori Ogiwara and Osamu Watanabe. On polynomial-time bounded truth-table reducibility of np sets to sparse sets. *SIAM Journal on Computing*, 20, 1991.

[Pap94] Christos Papadimitrou. *Computational complexity*. Addison-Wesley Longman, 1994.

ECCC ISSN 1433-8092
http://www.eccc.uni-trier.de/eccc
ftp://ftp.eccc.uni-trier.de/pub/eccc
ftpmail@ftp.eccc.uni-trier.de, subject 'help eccc'