COMBINATORICA
Akadémiai Kiadó — Springer-Verlag

COMBINATORICA 9 (4) (1989) 385—392

QUERY COMPLEXITY, OR WHY IS IT DIFFICULT TO SEPARATE $NP^A \cap coNP^A$ FROM P^A BY RANDOM ORACLES A?

G. TARDOS

Received April 13, 1987
Revised July 1, 1987

By the *query-time complexity* of a relativized algorithm we mean the total length of oracle queries made; the *query-space complexity* is the maximum length of the queries made. With respect to these cost measures one can define polynomially time- or space-bounded deterministic, nondeterministic, alternating, etc. Turing machines and the corresponding complexity classes. It turns out that all known relativized separation results operate essentially with this cost measure. Therefore, if certain classes do not separate in the query complexity model, this can be taken as an indication that their relativized separation in the classical cost model will require entirely new principles.

A notable unresolved question in relativized complexity theory is the separation of $NP^A \cap$ $\cap\, co\, NP^A$ from P^A under random oracles A. We conjecture that the analogues of these classes actually coincide in the query complexity model, thus indicating an answer to the question in the title. As a first step in the direction of establishing the conjecture, we prove the following result, where polynomial bounds refer to query complexity.

If two polynomially query-time-bounded nondeterministic oracle Turing machines accept precisely complementary (oracle dependent) languages L^A and $\{0, 1\}^* \setminus L^A$ under every oracle A then there exists a deterministic polynomially query-time-bounded oracle Turing machine that accept L^A. The proof involves a sort of greedy strategy to selecting deterministically, from the large set of prospective queries of the two nondeterministic machines, a small subset that suffices to perform an accepting computation in one of the nondeterministic machines. We describe additional algorithmic strategies that may resolve the same problem when the condition holds for a $(1-\varepsilon)$ fraction of the oracles A, a step that would bring us to a non-uniform version of the conjecture. Thereby we reduce the question to a combinatorial problem on certain pairs of sets of partial functions on finite sets.

1. Introduction

One of the purposes of relativized complexity theory has been to support the intuition that certain complexity classes like P and NP are different. Bennett and Gill have seen an even stronger evidence in separation by random oracles; in fact they conjectured [6] that if two appropriately defined complexity classes are separated by almost every oracle then they are different in the unrelativized world. Although an example due to S. Kurtz [16] refuted the Random Oracle Hypothesis, relativization still seems to work as a rule of thumb: if two classes do not appear to be amenable for oracle separation, one might try to show they are actually equal. (Admittedly, the proof [14] of the equivalence of the two versions of interactive proof systems [13], [4] arose along such a line of thought.)

Our aim is to indicate a limitation of currently used methods in proving separation by oracles and to point out a curious instance of the broadly interpreted

AMS subject classification (1980): 68Q15.

Bennett—Gill conjecture. Indeed, we indicate that a separation of $NP^A \cap coNP^A$ from P^A by random oracles may have unrelativized consequences such as a separation of BPP from some randomized version of NP and so conceivably P from $PSPACE$.

By Kolmogorov's zero-one law for tail events, random oracles separate any two relativized complexity classes (which do not depend on finite portions of the oracle) with probability zero or one. Bennett and Gill [6] prove that for almost every oracle A, $NP^A \neq coNP^A$. It follows from Yao's result [20] that almost every oracle A separates the polynomial time hierarchy from $PSPACE$ [11], [3]. On the other hand, not only the separation of the levels of the polynomial time hierarchy [21], [20] but even the seemingly harmless question of separation of $NP^A \cap coNP^A$ from P^A by random oracles A remains an open question [6].

We give some indication that this latter might be substantially more difficult to prove. We consider a model where cost is defined by the total length of oracle queries. (This model is closely related to but not identical with models studied by Book et al. [8], [9], [10].) The class corresponding to NP^A then consists of languages accepted with only a polynomial number of oracle queries of polynomial length along any computation path. We call the corresponding machines polynomially query-time-bounded. The analogues of other classes are similarly defined. One can copy all the known proofs of oracle separation of classical complexity classes to the query complexity model (Proposition 2.4), thus showing that the essence of those proofs is a separation by frustrating oracle access. On the other hand, we are able to prove, that if two polynomially query—time-bounded nondeterministic machines accept precisely complementary (oracle dependent) languages L^A and $\{0, 1\}^* \setminus L^A$ under (almost) every oracle A then there exists a deterministic, polynomially query-time-bounded machine accepting L^A. This result may be viewed as a first step toward establishing our conjecture that for polynomially query-time-bounded computations, the analogue of $NP^A \cap coNP^A$ is *equal* to the analogue of P^A for almost every oracle A. This would exclude the possibility of a proof of random oracle separation of P^A from $NP^A \cap coNP^A$ by currently available techniques, all of these being based on oracle access arguments.

2. The model

Let M be a deterministic, nondeterministic, or alternating oracle Turing machine that is guaranteed to halt. We measure the *query-time* of computation as the total length of oracle queries made along any computation path. The *query-space* is the maximum length of the queries made. Imposing a bound $\leq T(n)$ and $\leq S(n)$, resp., on these quantities, we obtain the complexity classes $QTIME(T(n))^A$, $QNTIME(T(n))^A$, $QALT_k(T(n))^A$ and $QSPACE(S(n))^A$. Taking unions over polynomially bounded $T(n)$ and $S(n)$, we obtain the classes QP^A, QNP^A, $Q\sum_k^{P,A}$, and $QPSPACE^A$, resp. Let QPH denote the union of the query-polynomial-time hierarchy.

Remark 2.1. Let $\mathscr{R}$ denote the set of recursive languages. By our definition, for any recursive oracle $A \in \mathscr{R}$,

$$QP^A = QNP^A = \ldots = QPSPACE^A = \mathscr{R}.$$

Remark 2.2. For the purposes of this paper we could restrict our Turing machines to some sufficiently large and robust resource-bounded class such as *PSPACE* without any change to the results and conjectures to follow. Under this restriction, our QP^A and QNP^A become identical with Book's $PQUERY(A)$ and $NPQUERY(A)$ [8].

Remark 2.3. There are no space hierarchies in the query-cost model.

Proposition 2.4. *All known relativized oracle separation results hold in the query complexity model. In particular, there exist oracles A under which* $QP^A \neq QNP^A \cap$ $\cap coQNP^A \neq QNP^A$; $QPH^A \neq QPSPACE$; *and the levels of the query-polynomial-time hierarchy are separated. Moreover, random oracles will, with Probability* 1, *separate* QNP^A *from* $coQNP^A$ *and* QPH^A *from* $QPSPACE^A$.

Proof. For the first part, copy [5]. For *QPSPACE* vs. *QPH*, copy the [12] proof combined with Yao's result [21] confirming the *FSS* conjecture on bounded depth parity circuits. Similarly, the oracle-separation of the levels of the query-polynomial-time hierarchy follows along the lines of [20], using [21]. The random oracle separation of QNP^A and $coQNP^A$ proceeds as in [6]; and the last statement follows as in [11] and [3]. ∎

Remark 2.5. Just as in the classical model, we are unable to prove the separation of the levels of the query-polynomial time hierarchy under random oracles.

Remark 2.6. In the separation results of [5], the oracles obtained via diagonalization may be required to be recursive. This is, of course, impossible in our case, by Remark 2.1. On the other hand, if we change the definition of the model according to the Remark 2.2, the oracles obtained may be required to be recursive.

We now state two contrasting conjectures which indicate that the situation in general may be more complicated than suggested by the proof of Proposition 2.4.

Conjecture 2.7. [6] *For almost every oracle A,*

$$NP^A \cap coNP^A \neq P^A.$$

Conjecture 2.8. *For almost every oracle A,*

$$QNP^A \cap coQNP^A = QP^A.$$

The rest of the paper will primarily be devoted to our approach to Conjecture 2.8 which we essentially reduce to problems of combinatorial nature and solve in a weaker formulation.

Why do we believe the truth of Conjecture 2.7? Let *RNP* (random-*NP*) consist of those languages L belonging to NP^A for almost every oracle A. (This class has been introduced in [4]; clearly $NP \subseteq AM \subseteq RNP$. For *AM*, see [2].)

Proposition 2.9. *If* $RNP \cap coRNP \neq BPP$ *then conjecture* 2.7 *holds. This is the case in particular if factoring integers cannot be done in BPP.*

Proof. Immediate by the result that *BPP* consists of all languages which belong to P^A for almost every A [6] (cf. [17]). The second sentence follows since factoring (defined in a natural way as a language recognition problem) is in $NP \cap coNP$ [19]. ∎

What Conjecture 2.8 indicates is that proving Conjecture 2.7 might not be much easier than the hopeless task of proving the condition in Proposition 2.9. At any rate, Conjecture 2.8 implies that query-based cost arguments will not suffice for a proof of Conjecture 2.7 and the reasons for separation must lie in the oracle-free part of the computation. Such a proof might be expected to actually separate unrelativized complexity classes.

Remark 2.10. If we replace random oracles by random *permutation oracles* in Conjecture 2.7, we obtain a true statement [6]. It would be interesting to find two complexity classes where random oracles and random permutation oracles are likely to produce different effects (i.e. separation with probability 1 in one case and with probability 0 in the other).

3. Deterministic simulation without loss of query-time

Our main result is the following.

Theorem 3.1. *Let M_1 and M_2 be polynomially query-time-bounded nondeterministic oracle Turing machines, such that for every oracle A,*

$$L^A(M_1) \cap L^A(M_2) = \emptyset.$$

Then there exists a polynomially query-time-bounded deterministic oracle Turing-machine M such that for every oracle A,

$$L^A(M) \supseteq L^A(M_1),$$

and

$$L^A(M) \cap L^A(M_2) = \emptyset.$$

In particular, if $L^A(M_1)$ and $L^A(M_2)$ are always complementary then $L^A(M)=L^A(M_1)$ for every A.

Remark 3.2. In the condition of the theorem, the phrase "every oracle" is equivalent to saying "almost every oracle". As a matter of fact, if even a single oracle A allowed a string x to be accepted by both machines, then, the accepting computations being dependent on a finite number of oracle entries only, x would be accepted by both machines under a positive fraction of the oracles. — In order to achieve further progress on Conjecture 2.8, we shall need to relax the condition by requiring disjointness of the two languages only under a $(1-\varepsilon)$ fraction of the oracles A. This leads to considerable combinatorial difficulties (cf. Section 4).

In order to facilitate the proof of the Theorem, we introduce a combinatorial model of the situation, and describe the simulation in that context.

Let Ω be a finite set of N elements and let m be a positive integer. (We think of N being exponentially large compared to m but this is not relevant.) We consider (0, 1)-valued functions defined on subsets of Ω. Such a function will be called a *total function* if $\operatorname{dom} f=\Omega$; and a *function germ* of length m, if $|\operatorname{dom} f| \leqq m$. Let $T(\Omega)$ denote the set of all total functions and $G(\Omega, m)$ the set of all function germs of lengths $\leqq m$.

Two partial functions are *compatible* if they agree wherever both are defined. Note that the empty germ ($\operatorname{dom} f=\emptyset$) is compatible with all functions. We shall refer to any set of function germs as a *germ system*. For a germ system F, let $C(F)$ denote the set of all total functions compatible with at least one member of F. Such total functions will be called *compatible with F*. Note in particular, that $C(\emptyset)=\emptyset$ and $C(\{\emptyset\})=T(\Omega)$. Moreover, $C(F\cup F')=C(F)\cup C(F')$.

We call two germ systems F, F' *fully incompatible* if $C(F)\cap C(F')=\emptyset$. Note that in this case, if one of the systems contains the empty germ, then the other system is empty.

We consider the following one-person game with incomplete information.

The Compatibility Game. Let F_1 and F_2 be a given pair of fully incompatible germ systems and A an unknown total function. We have to output either 1 or 2 such that the output should be i whenever $A\in C(F_i)$. (Any output is considered correct if $A\notin C(F_1)\cup C(F_2)$.) We are allowed any amount of computation but only a limited number of queries for the values of A.

The problem is to make the number of queries small.

Lemma 3.4. *If the germs in F_i have length m_i $(i=1, 2)$, then the Compatibility Game can be solved by $\leq m_1m_2$ queries.*

Proof. We describe the strategy. We call the members of F_1 *red germs,* and the members of F_2 *blue germs.*

```
Begin
    If the set of blue germs is empty, output "1", halt.
    Else, if the set of red germs is empty, output "2", halt.
    Else, select a red germ $f$. Query all values $A(j)$ for $j\in \text{dom}\, f$.
    For each $j\in \text{dom}\, f$, modify both germ systems as follows. Delete a germ $g$
    if $j\in \text{dom}\, g$ and $g(j)\neq A(j)$. Modify a germ $g$ (preserving its color) if $j\in \text{dom}\, g$
    and $g(j)=A(j)$ by deleting $j$ from its domain.
(* Clearly, the modified systems continue to be fully incompatible, and $A$ is com-
patible with the new $F_i$ if and only if it was with the old $F_i$. *)
Repeat
```

It is clear that the output will always be correct. The bound on the number of queries follows from the observation that in each round of reductions, the maximum length of blue germs is reduced at least by 1. Indeed, since the two germ systems are fully incompatible, the domain of each blue germ must intersect the domain of the red germ selected. ∎

Remark 3.5. There is a curious analogy between this result and a result of Aho, Ullman and Yannakakis in communication complexity theory [1]. Their result asserts that if nondeterministic communication protocols of length m_i exists that recognize a language and its complement, resp., then a deterministic protocol of length $\geq m_1m_2$ recognizes the same language. In this model, each of two communicating parties knows half the input; the cost is the number of bits sent.

Proof of Theorem 3.1. Let x denote an input of length n. Let $m=n^c$ be a common bound on the maximum number of queries made along any computation path by either machine, and let $m'\leq n^{c'}$ be a bound on the length of all queries. Let Ω be the set of all the oracle entries queried by either machine on input x. For the purposes of computations on input x, we may regard the oracle A as a (0, 1)-valued function with domain Ω.

Let us modify our nondeterministic machines as follows. Each time the machine M_i would call the oracle, it should instead of making a query, guess the outcome. At the end of each computation path, it will thus have guessed a set of at most m_i oracle entries, i.e., a function germ. Only if such a computation would turn out to accept x, will the machine query the oracle to check if the guesses were correct.

This way we associate a set F_i^x of function germs with each machine M_i (depending on the input x). We stress that these sets do not depend on the oracle. M_i accepts x under oracle A precisely if A is compatible with F_i^x. Thus the task of machine M on input x is modeled by the Compatibility Game, and the strategy given there solves the problem in polynomial $(m_1 m_2 m')$ query-time ($\leqq m_1 m_2$ queries, each of length $\leqq m'$). ▌

4. Nearly incompatible germ systems

Let us assume Conjecture 2.8 is false, i.e. for almost every oracle A,

$$QNP^A \cap coQNP^A \neq QP^A.$$

By the Lebesgue Density Theorem it follows that for any $\varepsilon > 0$ there exists a pair M_1, M_2 of polynomially query-time-bounded nondeterministic oracle Turing machines such that, for a random oracle A, each of the following hold with probability $> (1-\varepsilon)$:

(i) $L^A(M_1) \cap L^A(M_2) = \emptyset$;

(ii) $L^A(M_1) \cup L^A(M_2) = \{0, 1\}^*$;

(iii) $L^A(M_1) \notin QP^A$.

If this would hold with probability 1 instead of $(1-\varepsilon)$, it would contradict Theorem 3.1.

In order to handle the situation when (i) and (ii) hold with probability $(1-\varepsilon)$, we need to relax the condition in Theorem 3.1 that the given systems of germs be fully incompatible. We use the terminology of Section 3.

For two germ-systems F_1, F_2 we define the *set of ambiguity* $\text{amb}(F_1, F_2)$ as the set of those total functions $A \in T(\Omega)$ which are either compatible to both systems or to neither.

We call the germ systems F_1 and F_2 *δ-nearly incompatible,* if the number of those total functions $A: \Omega \to \{0, 1\}$ compatible with both systems is less than $\delta 2^N$ (where $N = |\Omega|$), i.e. $|C(F_1) \cap C(F_2)| < \delta 2^N$.

We call the germ systems F_1 and F_2 *δ-nearly complementary,* if $|\text{amb}(F_1, F_2)| < \delta 2^N$. (Such systems are in particular δ-nearly incompatible.)

In the *ε-error Compatibility Game* the output is allowed to be in error for $\leqq \varepsilon 2^N$ total functions A. (Arbitrary output is considered correct if $A \in \text{amb}(F_1, F_2)$.)

Problem 4.1. Show that for some absolute constant c and for every $\varepsilon > 0$ there exists $\delta > 0$ such that for every N and m and every pair of δ-nearly complementary germ systems with parameters m, N there exists a deterministic strategy to play the ε-error Compatibility Game with at most m^c queries for values of the unknown total function A.

Unfortunately the solution of this problem would not yet resolve Conjecture 2.8 because it would not yield a uniform procedure for all inputs x. Uniformity (and thus the Conjecture) would follows from a solution to the following problem.

Instead of the finite set Ω we shall have to consider the set N of natural numbers.

Let $T(\mathrm{N})$ be the set of all functions $\mathrm{N} \to \{0, 1\}$ (oracles). A subset C of $T(\mathrm{N})$ is a *cylinder* if $\exists n_0$ such that $\forall f \in T(\mathrm{N})$, the relation $f \in C$ depends only on the restriction of f to $\{1, \ldots, n_0\}$.

Let $\mathrm{Cyl}(\mathrm{N})$ be the set of cylinders and $S\colon \mathrm{Cyl}(\mathrm{N}) \to \mathrm{Cyl}(\mathrm{N})$ a monotone recursive function. We call S *ε-bounded* if $\forall Y \subset T(\mathrm{N})$ with Lebesgue measure $P(Y) < \varepsilon$ we have $P(S(Y)) < 1/2$.

Problem 4.2. Show that there exist $c, \varepsilon > 0$ and an ε-bounded set operation S: $\mathrm{Cyl}(\mathrm{N}) \to \mathrm{Cyl}(\mathrm{N})$ such that for every m and every pair F_1, F_2 of finite systems of germs of length m over N there exists a strategy to play the Compatibility Game with $\leqq m^c$ queries s.t. the output must be correct for every $A \notin S(\mathrm{amb}(F_1, F_2))$.

(Note: we may assume $|F_i| \leqq 2^m$.)

We devote the rest of this section to recommending strategies to solve Problem 4.1. Fix an appropriate absolute constant d.

Strategy A. Query $j \in \Omega$ if for $i = 1$ or 2

$$|\{f \in F_i\colon j \in \mathrm{dom} f\}| / |F_i| > m^{-d}.$$

Strategy B. Let $\min(F_i, A) = \{f \in F_i\colon A \supset f$ and $\forall g \in F_i$, if $A \supset g$ then $|\mathrm{dom}(g)| \geqq |\mathrm{dom}(f)|\}$.

Query $j \in \Omega$ if for $i = 1$ or 2,

$$\mathrm{Prob}_A(\exists f \in \min(F_i, A) \text{ such that } j \in \mathrm{dom} f) \geqq m^{-d}.$$

After each step of either strategy we reduce the germ systems as in the proof of Lemma 3.4. Even if we combine the two strategies, the procedure will terminate after a polynomial number of queries. When no more move is available, we output "i" if $|C(F_i)| > C(F_{3-i})|$.

Let us call a pair of germ-systems *dispersed* with respect to a given strategy, if it does not allow any moves. What we would need is the following type of result:

$$\forall \varepsilon \exists \delta \forall \text{ dispersed pair } F_1, F_2 \text{ if } |C(F_i)| > \varepsilon 2^N \text{ for } i = 1, 2, \text{ then } \mathrm{amb}(F_1, F_2) > \delta 2^N.$$

We note that if, for randomly selected functions A, the events $A \in C(F_i)$ $(i = 1, 2)$ were independent, then $\delta = \varepsilon^2$ would be an appropriate choice. Being dispersed seems to be close to making these events independent, especially under Strategy B.

Both Strategies A and B are easily seen to solve Lemma 3.1 (the case when $C(F_1)$ and $C(F_2)$ are disjoint).

Remarks. It has come to my attention that Theorem 3.1 has simultaneously and independently been found by Manuel Blum and Russel Impagliazzo [7, Lemma 2.2] and by Juris Hartmanis and Lane Hemachandra [15, Thm. 4.3]. Both papers consider the ordinary cost measure (as opposed to query complexity) and overcome the resulting difficulties by adding the assumption $P = NP$.

N. Nisan [17] points out that Lemma 3.4 can be rephrased in terms of decision tree complexity of Boolean functions. For a Boolean function

$$f\colon \{0, 1\}^n \to \{0, 1\},$$

the deterministic decision tree complexity $d(f)$ is defined as the maximum number of queries of the form "$x_i = ?$" that an optimal player is forced to ask in worst case in order to find out the value $f(x_1, \ldots, x_n)$. (The function f is known to the player but the input $(x_1, \ldots, x_n)$ is not.) The nondeterministic decision tree complexity

$N^{(1)}(f)$ is the maximum over all $\underline{x} \in \{0,1\}^n$ such that $f(\underline{x})=1$ of the number of input bits an optimal nondeterministic prover is forced to reveal in order to prove $f(\underline{x})=1$. The conondeterministic complexity of f is $N^{(0)}(f)=N^{(1)}(1-f)$. Using this terminology, Lemma 3.1 is equivalent to the following result:

$$D(f) \leqq N^{(1)}(f)N^{(0)}(f).$$

Related results on randomised decision tree complexity are derivad in [17].

Acknowledgement. I am indebted to László Babai for suggesting the problem as well as for helpful discussions and encouragement.

References

[1] A. V. Aho, J. D. Ullman and M. Yannakakis, On Notions of Information Transfer in VLSI Circuits, *Proc. 15th STOC*, 1983, 133—139.
[2] L. Babai, Trading group theory for randomness, *Proc. 17th STOC* (1985), 421—429.
[3] L. Babai, Random oracles separate *PSPACE* from the polynomial time hierarchy, Technical Report 86—001 (1986), Dept. Comp. Sci., Univ. of Chicago; to appear in *Inf. Proc. Letters.*
[4] L. Babai, Arthur—Merlin games: a randomized proof system and a short hierarchy of complexity classes, *JCSS, to appear.*
[5] T. Baker, J. Gill and R. Solovay, Relativizations of the $P=?NP$ question, *SIAM J. Comp.*, 4 (1975), 431—442.
[6] C. H. Bennett and J. Gill, Relative to a random oracle A, $P^A \neq NP^A \neq coNP^A$ with probability 1, *SIAM J. Comp.*, 10 (1981), 96—113.
[7] M. Blum and R. Impagliazzo, Generic oracles and oracle classes, *Proc. 28th FOCS* (1987), 118—126. Extended Abstract.
[8] R. V. Book, Bounded query machines: on *NP* and *PSPACE*, *Theoretical Computer Science*, 15 (1981), 27—39.
[9] R. V. Book, T. J. Long and A. L. Selman, Quantitative relativization of complexity classes, *SIAM J. Comp.*, 13 (1984), 461—487.
[10] R. V. Book and C. Wrathall, Bounded query machines: on NP() and NPQUERY(), *Theoretical Computer Science*, 15 (1981), 41—50.
[11] J. Y. Cai, With Probability One A Random Oracle Separates *PSPACE* from the Polynomial Hierarchy, *Proc. 18th STOC* (1986), 21—29.
[12] M. L. Furst, J. Saxe and M. Sipser, Parity, circuits, and the polynomial time hierarchy, *Proc. 22nd FOCS* (1981), 260—270.
[13] S. Goldwasser, S. Micaly and C. Rackoff, The knowledge complexity of interactive proof-systems, *Proc. 17th STOC*, 1985, 291—304.
[14] S. Goldwasser and M. Sipser, Private coins versus public coins in interactive proof systems, *Proc. 18th STOC* (1986), 59—68.
[15] J. Hartmanis and L. A. Hemachandra, One-way functions, robustness, and the non-isomorphism of NP-complete sets, *Proc. 2nd Structurde in Complexity Theory*, (87), 160—173.
[16] S. A. Kurtz, On the Random Oracle Hypothesis, *Proc. 14th STOC* (1982), 224—230.
[17] S. A. Kurtz, A Note on Randomized Polynomial Time, *SIAM J. Comp.*, 16 (1987), 852—853.
[18] N. Nisan, Probabilistic vc. Deterministic Decision Trees and CREW PRAM Complexity, *preprint.*
[19] V. R. Pratt, Every Prime Has a Succinct Certificate, *SIAM J. Comp.*, (1975), 214—220.
[20] M. Sipser, Borel sets and circuit complexity, *Proc. 15th STOC* (1983), 61—69.
[21] A. C.-C. Yao, Separating the polynomial-time hierarchy by oracles, *Proc. 26th FOCS* (1985), 1—10.

Tardos Gábor

Department of Algebra
Eötvös University
Múzeum krt. 6—8
Budapest, Hungary H—1088