1400008469
còpia
FACULTAT D'INFORMÀTICA
BIBLIOTECA
U. P. C.

Only smart oracles help

J.L. Balcázar

Report LSI–88–9

FACULTAT D'INFORMÀTICA
BIBLIOTECA
R. 4880 04 DIC. 1989

ONLY SMART ORACLES HELP

Very short technical note
Written july 1987, revised september 1988

José L. Balcázar
Department of Software (Llenguatges i Sistemes Informàtics)
Universitat Politècnica de Catalunya
08028 Barcelona, SPAIN
e-mail: eabalqui@ebrupc51.bitnet

Abstract. It is shown that every set that is p-cheatable up to a logarithmic extent is a no-1-helper.

Resum. Demostrem que tot conjunt que és "p-chatable" fins al logaritme és també un "no-1-helper".

1. Setting. Recent work on p-cheatable sets and related notions, like terseness and verbosity, includes [2] and [3]. Helping has been studied in [5] and [6]. I point out here a connection between those subjects, by showing that a certain variety of polynomial time cheatable sets are unable to help polynomial time computations. Check these references for undefined notions.

2. Definition. A set A is *polynomial time cheatable up to a logarithmic extent* (briefly *p-cheatable to log*) if and only if for every deterministic oracle machine M with output which runs in polynomial time under A, there is another machine M' which computes the same function relative to A, makes at most $c \cdot \log n$ queries to A on inputs of length n, and runs in polynomial time under A.

3. Remark. The idea behind the previous definition is just that the information provided by the oracle has a lot of redundancy, so that most of the (potentially) polynomially many queries allowed to the oracle machine M by its time bound are "unnecessary". Thus, the same task can be accomplished by a machine M' which queries A only logarithmically many times.

4. Remark. In [2] and [3] p-cheatability is defined in a different form. There, the parameters indicating how many queries are enough to decide how many questions are

constants. I want them to vary as a function of the input length, and therefore I propose this definition. I do not expect these notions to be equivalent.

5. Lemma. If A is p-cheatable to log and A 1-helps the robust machine M, then A 1-helps a robust machine M', accepting the same language as M, which on inputs of length n makes at most $c \cdot \log n$ queries to A.

6. Proof. Define a third oracle machine M'' which, given x and oracle A, computes a polynomially long accepting computation of M on x under A, by simulating M on x. If no such computation exists then it outputs some symbol indicating "undefined". It is clear that it works in polynomial time under oracle A. Call f the function computed by this machine under A. By the hypothesis that A 1-helps M, $f(x)$ is defined if and only if x is accepted by M. Obtain a machine computing the same function with logarithmically many queries, provided by the p-cheatability to log of A, and modify it to construct M': if a value $f(x)$ is obtained, then accept; else run a slow machine for $L(M)$. This machine fulfills the required conditions.

7. Remark. The lemma says that the savings in the number of queries can be obtained without loss of robustness. My initial definition of p-cheatability to log was in terms of accepting languages instead of computing functions. I do not know whether a definition in terms of accepting languages allows one to prove this lemma, or even to prove the next theorem without using it.

8. Theorem. If A is p-cheatable to log then A is a no-1-helper.

9. Proof. Assume that A 1-helps M. We show that $L(M) \in P$. Use the lemma to construct a robust machine M' accepting $L(M)$ and such that A 1-helps M' and is queried only logarithmically many times. Let q be a polynomial bounding the running time of M' under A. The computation of M' on an accepted input x can be viewed as a tree in which some accepting branch has length $q(|x|)$ and contains only logarithmically branching nodes. A polynomial time machine M'' is able to decide whether x is accepted by M' by following systematically each branch until finding an accepting one, pruning each branch as soon as the time or query bounds are exceeded. It is easy to see that the searched part of the tree is polynomial in size.

10. Remark. The previous theorem is not vacuosly true: there are nontrivial p-cheatable to log sets. As examples, the $\log^*$-sparse set constructed in [5], lemma 3.2, and sets constructed in a similar way to [2], theorem 2, are p-cheatable to log (and sometimes, even p-cheatable to 1). See also [1], where other properties of this sort of highly sparse sets are discussed.

11. Remark. Relating the results of [1], a proof very similar to that of the theorem above can be used to show the intuitively clear result that polynomial time Turing

FACULTAT D'INFORMÀTICA BIBLIOTECA U. P. C.

reducibility to a p-cheatable to log set is inherently nonadaptive. This can be formalized as follows.

12. Proposition. If A is p-cheatable, and $B \leq_T^p A$, then $B \leq_{tt}^p A$.

13. Proof. Transform the machine witnessing the reducibility into another that queries only logarithmically many times, and traverse in polynomial time the full computation tree of the new machine collecting the queries that appear in it into a polynomially long list.

14. Remark. Results related to this proposition are theorem 3.10(1) in [4] and corollary 3.6 in [7]. These theorems relate polynomially many nonadaptive queries to logarithmically many adaptive queries to NP oracles. A number of open problems are listed in what follows; some of them might be investigated by the author in the future.

15. Question. What is the exact relationship of the concept of p-cheatability to log with other bounded cheatability definitions in the literature?

16. Question. What is the exact relationship of the p-cheatable to log sets with other no-1-helpers like those in [5]? For instance:

17. Conjecture. Every p-cheatable to log set is in the class LOG-INF.

18. Question. It has been recently shown by Amir (unpublished) that sets Turing reducible in polynomial time to 2-for-1-p-cheatable (i.e. not 2-terse) sets are themselves 2-for-1-p-cheatable. Is the class of p-cheatable to log sets closed under polynomial time Turing reducibility?

19. Question. Are p-cheatable to log sets somehow "low"? For instance: do they have polynomial size circuits?

20. Question. (The BIG one.) *Does the notion of "p-cheatable to log sets" really make sense? Why?*

21. Question. If the answer to the preceding one is NO, do not cry: one can still find interesting questions around. For instance, assume that $P_{1-\text{help}}(A)$ is nontrivial (i.e. neither P nor NP); does it have complete sets?

References

[1] K. Ambos-Spies: "Inhomogeneities in the polynomial-time degrees: the degrees of super sparse sets". *Inf. Proc. Lett.* 22 (1986), 113–117.

[2] A. Amir and W. Gasarch: "Polynomial terse sets". In: Proc. 2nd Structure in Complexity Theory Conference (1987), 22–27.

[3] J. Goldsmith, D. Joseph, and P. Young: "Self-reducible, p-selective, near-testable, and p-cheatable sets: the effect of the internal structure on the complexity of a set". In: Proc. 2nd Structure in Complexity Theory Conference (1987), 50–59.

[4] L. Hemachandra: "The strong exponential hierarchy collapses". In: Proc. 19th ACM Symp. Th. of Comp. (1987), 110–122.

[5] K. Ko: "On helping by robust oracle machines". *Theor. Comp. Sci.* 52 (1987), 15–36.

[6] U. Schöning: "Robust algorithms: a different approach to oracles". *Theor. Comp. Sci.* 40 (1985), 57–66.

[7] K. Wagner: "On restricting the access to an NP oracle". In: Proc. 15th Int. Coll. Aut., Lang., and Prog., Lect. Notes in Comp. Sci. 317, 682–696.