Theoretical Computer Science 40 (1985) 237–243
North-Holland

ON BOUNDED QUERY MACHINES*

Jose L. BALCÁZAR
Facultat d'Informàtica, Universitat Politècnica de Barcelona, Barcelona, 34, Spain

Ronald V. BOOK
Department of Mathematics, University of California, Santa Barbara, CA 93106, U.S.A.

Uwe SCHÖNING
Institut für Informatik, Universität Stuttgart, D-7000 Stuttgart 1, Fed. Rep. Germany

Communicated by M.S. Paterson
Received April 1984
Revised January 1985

Abstract. Simple proofs are given for each of the following results: (a) P = PSPACE if and only if, for every set A, P(A) = PQUERY(A) (Selman et al., 1983); (b) NP = PSPACE if and only if, for every set A, NP(A) = NPQUERY(S) (Book, 1981); (c) PH = PSPACE if and only if, for every set A, PH(A) = PQH(A) (Book and Wrathall, 1981); (d) PH = PSPACE if and only if, for every sparse set S, PH(S) = PQH(S) = PSPACE(S) (Balcázar et al., 1986; Long and Selman, 1986).

1. Introduction

Is the union of the polynomial-time hierarchy equal to PSPACE? This question was considered by Book and Wrathall [4] in the context of relativizations of complexity classes. They showed that there was a restricted relativization of PSPACE( ), denoted PQH( ), such that the union of the polynomial-time hierarchy (PH) is equal to PSPACE if and only if, for every set A, the union of the polynomial-time hierarchy relative to A (PH(A)) is equal to PQH(A). The proofs in [4] depended on language-theoretic characterizations of classes of the form PH(A) and PQH(A) and, so, were not accessible to many of those who are interested in complexity-bounded reducibilities.

In this article we give new characterizations of classes of the form PQUERY(A), NPQUERY(A), and PQH(A) in terms of the P( )-, NP( )-, and PH( )-operators, respectively, and then give simple proofs of the results in [2], [4], and [10] having to do with the "P =? PSPACE", "NP =? PSPACE", and "PH =? PSPACE" problems. Recently, it has been shown [1, 7] that PH is equal to PSPACE if and only if, for every sparse set S, PH(S) is equal to PSPACE(S). We give a new and simple proof of this fact after showing that, for every sparse set S, PSPACE(S) = PQH(S) = $\Delta_2^{PQ}(S)$.

* This research was supported in part by the U.S.A.-Spanish Joint Committee for Education and Cultural Affairs, by the Deutsche Forschungsgemeinschaft, and by the National Science Foundation under Grant No. DCR83-12472.

0304-3975/85/$3.30 © 1985, Elsevier Science Publishers B.V. (North-Holland)

It is assumed that the reader is familiar with the basic literature on relativizations of P, NP, PSPACE, etc., and with the polynomial-time hierarchy (see [11, 12]). Where we refer to the language SAT the reader may substitute any set that is $\leq_T^P$-complete for NP; similarly, where we refer to the language QBF the reader may substitute any set that is $\leq_T^P$-complete for PSPACE.

For a string w, $|w|$ denotes the length of w. For a finite set S, $\|S\|$ denotes the cardinality of S.

Let $<$ denote any standard polynomial-time computable total order defined on Σ^*; lexicographic ordering will do. For a finite set $S \subset \Sigma^*$, say $S = \{y_1, \ldots, y_n\}$ where $i < j$ implies $y_i < y_j$, let $\langle S \rangle = \%y_1\% \ldots \%y_n\%$ where % is a symbol not in Σ. Let $\langle \emptyset \rangle = \%$. We consider $\langle \, , \rangle$ to be an encoding function. Notice that if $S \in \Sigma^*$ is a finite set and $y \in \Sigma^*$, then the predicate "y is in S" can be computed in linear time from the inputs y and $\langle S \rangle$. We use this same notation for pairing functions so that $\langle x_1, \ldots, x_n \rangle$ denotes $\langle \{x_1, \ldots, x_n\} \rangle$ when each x_i is in Σ^*, and $\langle x, S \rangle$ denotes $\langle \{x\} \oplus S \rangle$ when $x \in \Sigma^*$ and $S \subset \Sigma^*$ with S being finite.

2. Bounded query machines

Here we define the restricted relativizations in terms of the corresponding reduction classes and provide new characterizations of the reduction classes.

Definition 2.1. For every set A, let PQUERY(A) (NPQUERY(A)) be the class of languages $L \in$ PSPACE(A) such that there is a deterministic (respectively nondeterministic) polynomial space-bounded oracle machine M with the following properties:

(i) M recognizes L relative to A, and

(ii) there is a polynomial p such that for all x, in every computation of M on x there are at most $p(|x|)$ oracle queries.

Reduction classes of the form PQUERY(A) and NPQUERY(A) are invariant under the following changes in the definition:

(a) 'in every computation' is replaced by 'in some accepting computation';

(b) 'in every computation' is replaced by 'in every accepting computation'.

The following characterization of reduction classes of the form PQUERY(A) will be useful.

Lemma 2.2. *For every set* A, PQUERY(A) = P(QBF$\oplus A$) *and* NPQUERY(A) = NP(QBF$\oplus A$).

Proof. The fact that PQUERY(A) $\supseteq$ P(QBF$\oplus A$) is immediate since QBF $\in$ PSPACE and P(PSPACE) = PSPACE.

Let $L \in$ PQUERY(A) be witnessed by a machine M_1 that uses at most $q(n)$ oracle queries in any computation and uses $p(n)$ work space, where $p(n)$ and $q(n)$ are polynomials. Since M_1 is deterministic and uses at most $p(n)$ work space, we can assume that every computation of M_1 halts and is either accepting or rejecting. Thus, for any configuration I there is a *unique* query either accepting or rejecting configuration J reachable from I without querying the oracle. Further, there is a deterministic transducer T_1 that on input I will output J and T_1 needs use only polynomial work space.

Let $B = \{\langle I, x\rangle \mid I$ is a configuration of M and x is a prefix of the unique query or accepting or rejecting configuration reachable from I without querying the oracle$\}$. Clearly, $B \in$ PSPACE and so $B \leq_T^P$ QBF. A deterministic polynomial time-bounded oracle transducer T_2 that uses binary search can simulate T_1 by computing relative to QBF.

Now, the number of oracle queries that M_1 makes in any computation on an input x is at most $q(|x|)$. Thus, a deterministic polynomial time-bounded oracle machine M_2 can simulate M_1 if relative to QBF$\oplus A$; it generates query (either accepting or rejecting) configurations by simulating T_2 relative to QBF and then queries the oracle about membership in A for the appropriate string encoded in the query configuration. Such a machine M_2 witnesses L's membership in P(QBF$\oplus A$).

The argument in the nondeterministic case is similar but simpler. If M_1 witnesses $L \in$ NPQUERY(A), then consider a nondeterministic polynomial time-bounded oracle transducer that, given nonquery configuration I, will nondeterministically guess a query either accepting or rejecting configuration J and then deterministically check relative to QBF whether a computation of M_1 reaches J from I without querying the oracle. A nondeterministic polynomial time-bounded machine M_2 can simulate M_1 by computing relative to QBF$\oplus A$ and using T_1. □

A characterization of the reduction classes NPQUERY(A) using the notions of formal language theory was given in [2].

The following results show the usefulness of the notions of 'PQUERY( )' and 'NPQUERY( )'.

Theorem 2.3. (a) P = PSPACE *if and only if, for every set A*, P(A) = PQUERY(A) [10].
(b) NP = PSPACE *if and only if, for every set A*, NP(A) = NPQUERY(A) [2].

Proof. Since P(∅) = P, PQUERY(∅) = PSPACE, NP(∅) = NP, and NPQUERY(∅) = PSPACE, the proofs from right to left are trivial. To prove (a), note that Lemma 2.2(a) shows that, for every set A, PQUERY(A) = P(QBF$\oplus A$). Under the hypothesis P = PSPACE, QBF ∈ P so that PQUERY(A) = P(QBF$\oplus A$) ⊆ P(A). The result follows since P(A) ⊆ PQUERY(A). The proof of (b) is similar. □

The characterization of PQUERY(A) as P(QBF$\oplus A$) suggests similar operators. Book, Long and Selman [3] and Long [6] have used the operator P(SAT$\oplus A$). In studying the random oracle hypothesis, Kurtz [5] has shown that PQUERY(A) and P(SAT$\oplus A$) have similar properties, a fact that is not surprising when one sees that PQUERY(A) = P(QBF$\oplus A$).

The polynomial-time hierarchy relative to a set A can be obtained by beginning with NP(A) and iterating the NP( ) operator. That is, $\Sigma_1^{\mathrm{P}}(A) = \mathrm{NP}(A)$ and, for $i > 0$, $\Sigma_{i+1}^{\mathrm{P}}(A) = \mathrm{NP}(\Sigma_i^{\mathrm{P}}(A)) = \bigcup\{\mathrm{NP}(B) \mid B \in \Sigma_i^{\mathrm{P}}(A)\}$. Book and Wrathall [4] introduced the 'polynomial-query hierarchy relative to A' by using the NPQUERY( ) operator.

Definition 2.4. Let A be a set. Define $\Sigma_1^{\mathrm{PQ}}(A) = \mathrm{NPQUERY}(A)$, and for each integer $i > 0$ define $\Sigma_{i+1}^{\mathrm{PQ}}(A) = \mathrm{NPQUERY}(\Sigma_i^{\mathrm{PQ}}(A))$. For each $i > 0$, define $\Pi_i^{\mathrm{PQ}}(A) = \text{co-}\Sigma_i^{\mathrm{PQ}}(A)$ and $\Delta_{i+1}^{\mathrm{PQ}}(A) = \mathrm{PQUERY}(\Sigma_i^{\mathrm{PQ}}(A))$, and define $\Delta_1^{\mathrm{PQ}}(A) = \mathrm{PQUERY}(A)$. The structure $\{(\Delta_i^{\mathrm{PQ}}(A), \Sigma_i^{\mathrm{PQ}}(A), \Pi_i^{\mathrm{PQ}}(A)\}_{i \geq 1}$ is the *polynomial-query hierarchy relative to* A. Define $\mathrm{PQH}(A) = \bigcup_{i \geq 1} \Sigma_i^{\mathrm{PQ}}(A)$.

For every A, $\mathrm{PSPACE}(A) = \bigcup_{k \geq 1} \mathrm{DSPACE}(n^k, A) = \bigcup_{k \geq 1} \mathrm{NSPACE}(n^k, A) = \text{co-}\mathrm{PSPACE}(A)$ [9], so that $\mathrm{PQUERY}(\emptyset) = \mathrm{NPQUERY}(\emptyset) = \mathrm{PQH}(\emptyset) = \mathrm{PSPACE}(\emptyset) = \mathrm{PSPACE}$.

The following result will be useful.

Lemma 2.5. *For every* $i \geq 1$ *and every set* A, $\Sigma_i^{\mathrm{PQ}}(A) = \Sigma_i^{\mathrm{P}}(\mathrm{QBF} \oplus A)$. *Thus, for every set* A, $\mathrm{PQH}(A) = \mathrm{PH}(\mathrm{QBF} \oplus A)$.

The proof of Lemma 2.5 is by induction on i. The initial step is Lemma 2.2. The details are left to the reader.

Lemma 2.5 allows us to prove the next result which was first established in [4]. The proof is just like that of Theorem 2.3.

Theorem 2.6. PH = PSPACE *if and only if, for every set* A, $\mathrm{PH}(A) = \mathrm{PQH}(A)$.

Theorems 2.3 and 2.6 represent the first examples of 'positive relativizations' of questions about the comparison of complexity classes. If there exists a set A such that $\mathrm{P}(A) \neq \mathrm{PQUERY}(A)$ ($\mathrm{NP}(A) \neq \mathrm{NPQUERY}(A)$, $\mathrm{PH}(A) \neq \mathrm{PQH}(A)$), then $\mathrm{P} \neq \mathrm{PSPACE}$ (respectively, $\mathrm{NP} \neq \mathrm{PSPACE}$, $\mathrm{PH} \neq \mathrm{PSPACE}$). If for every set A (in particular, $A = \emptyset$), $\mathrm{P}(A) = \mathrm{PQUERY}(A)$ ($\mathrm{NP}(A) = \mathrm{NPQUERY}(A)$, $\mathrm{PH}(A) = \mathrm{PQH}(A)$), then $\mathrm{P} = \mathrm{PSPACE}$ (respectively, $\mathrm{NP} = \mathrm{PSPACE}$, $\mathrm{PH} = \mathrm{PSPACE}$).

3. PH versus PSPACE

Recall that a set S is *sparse* if there is a polynomial p such that, for all n, $\|\{x \in S \mid |x| \leq n\}\| \leq p(n)$. Long and Selman [7] and, independently, the present authors [1] have shown that PH = PSPACE if and only if, for every sparse set S, $\mathrm{PH}(S) = \mathrm{PSPACE}(S)$. Now, the empty set is sparse so that from Theorem 2.6 we see that PH = PSPACE if and only if, for every sparse set S, $\mathrm{PH}(S) = \mathrm{PQH}(S)$. The relationship between these results becomes clear once we have shown that, for every sparse set S, $\mathrm{PSPACE}(S) = \mathrm{PQH}(S) = \Delta_2^{\mathrm{PQ}}(S)$.

For any set A, let $\text{enum}_A(0^n) = \langle\{x \in A \mid |x| \leq n\}\rangle$ and let $\text{prefix}(A) = \{\langle x, 0^n\rangle \mid$ there exists y such that $|xy| = n$ and $xy \in A\}$.

For any set A, $\text{prefix}(A) \in \text{NP}(A)$. Furthermore, it is clear that if S is sparse, then there is a deterministic polynomial time-bounded oracle transducer that computes the function $0^n \mapsto \text{enum}_S(0^n)$ when the set $\text{prefix}(S)$ is used as the oracle set (see [6] or [8]).

Lemma 3.1. *If S is a sparse set, then* $\text{PSPACE}(S) = \text{PQUERY}(\text{prefix}(S)) = \text{PQH}(S) = \Delta_2^{\text{PQ}}(S)$.

Proof. Let M_1 be a deterministic oracle machine that witnesses $L \in \text{PSPACE}(S)$ and uses workspace at most $p(n)$ for some polynomial p. Let M_2 be a deterministic oracle machine that on input x, first computes $\text{enum}_S(0^{p(|x|)})$ and then simulates M_1's computation on x relative to S by querying the list $\text{enum}_S(0^{p(|x|)})$ instead of the oracle. Clearly, M_2 witnesses $L \in \text{PQUERY}(\text{prefix}(S))$. Thus, $\text{PSPACE}(S) \subseteq \text{PQUERY}(\text{prefix}(S))$.

Since $\text{prefix}(S) \in \text{NP}(S)$ and $S \in \text{NP}(S)$, $\text{PQUERY}(\text{prefix}(S)) \subseteq \text{PQUERY}(\text{NP}(S)) \subseteq \text{PQUERY}(\text{NPQUERY}(S)) = \Delta_2^{\text{PQ}}(S)$. Since $\Delta_2^{\text{PQ}}(S) \subseteq \text{PQH}(S) \subseteq \text{PSPACE}(S)$, we have $\text{PSPACE}(S) = \text{PQUERY}(\text{prefix}(S)) = \text{PQH}(S) = \Delta_2^{\text{PQ}}(S)$. □

The technique used in the proof of Lemma 3.1 has been used by a variety of authors. Of particular interest are the papers by Mahaney [8] and Long [6].

Combining Lemma 3.1 and Proposition 2.6, we have [1, Theorem 3.3] and [7, Proposition 3.13].

Theorem 3.2. *The following equalities are equivalent*:

(a) $\text{PH} = \text{PSPACE}$;

(b) *for every sparse set* S, $\text{PH}(S) = \text{PSPACE}(S)$;

(c) *for every sparse set* S, $\text{PH}(S) = \text{PQH}(S)$;

(d) *for every sparse set* S, $\text{PH}(S) = \Delta_2^{\text{PQ}}(S)$.

It is known [2] that there is a sparse set S such that $\Sigma_1^{\text{PQ}}(S) \neq \Delta_2^{\text{PQ}}(S)$ and so, $\Sigma_1^{\text{PQ}}(S) \neq \text{PQH}(S)$. Thus, Theorem 3.2 cannot be improved by substituting $\Sigma_1^{\text{PQ}}(S)$ for $\Delta_2^{\text{PQ}}(S)$ in part (d).

Assume that QBF is self-reducible and $\leq_T^{\text{P}}$-complete for PSPACE. It is clear that, for every set A, $\Delta_2^{\text{PQ}}(A) = \Delta_2^{\text{P}}(\text{QBF} \oplus A)$. Thus, part (d) of Theorem 3.2 is equivalent to the following statement: for every sparse set S, $\text{QBF} \in \text{PH}(S)$. It is shown in [1] that $\text{PH} = \text{PSPACE}$ if and only if there exists a sparse set S such that $\text{QBF} \in \text{PH}(S)$. Thus, either for every sparse S, $\text{QBF} \notin \text{PH}(S)$, or for every sparse S, $\text{QBF} \in \text{PH}(S)$. Hence, $\text{PH} \neq \text{PSPACE}$ if and only if, for every sparse set S, $\text{PH}(S) \neq \text{PSPACE}(S)$.

4. Restricting NPQUERY( )

Book, Long and Selman [3] have considered restrictions on the NP( )-operator in order to obtain positive relativizations of the "P = ? NP" problem. Here we consider similar restrictions of the NPQUERY( )-operator.

For oracle machine M, oracle set A, and input x, let $Q(M, A, x) = \{y \mid \text{in some computation of } M \text{ on } x \text{ relative to } A, \text{ the oracle is queried about } y\}$.

For any set A, let $\mathrm{NP_B}(A) = \{L \mid$ there is a nondeterministic polynomial time-bounded oracle machine M witnessing $L \in \mathrm{NP}(A)$ and a polynomial q such that, for all x, $\|Q(M, A, x)\| \leq q(|x|)\}$.

It is shown in [3] that P = NP if and only if, for all sets A, $\mathrm{P}(A) = \mathrm{NP_B}(A)$. Long [6] has extensively investigated the $\mathrm{NP_B}(\ )$-operator.

For any set A, let $\mathrm{NPQUERY_B}(A) = \{L \mid$ there is a nondeterministic polynomial query-bounded oracle machine M witnessing $L \in \mathrm{NPQUERY}(A)$ and a polynomial q such that, for all x, $\|Q(M, A, x)\| \leq q(|x|)\}$.

It is noted in [3] that, for every set A, $\mathrm{NPQUERY_B}(A) = \mathrm{PQUERY}(A)$. Also, one can prove this using Savitch's Theorem [8]. Since $\mathrm{NPQUERY}(A) = \mathrm{NP}(\mathrm{QBF} \oplus A)$ by Lemma 2.2, we are led to ask whether $\mathrm{NPQUERY_B}(A) = \mathrm{NP_B}(\mathrm{QBF} \oplus A)$.

It is shown in [3] that, for every set A, $\mathrm{P}(A) \subseteq \mathrm{NP_B}(A) \subseteq \mathrm{P}(\mathrm{SAT} \oplus A)$. Thus, for every set A, $\mathrm{PQUERY}(A) = \mathrm{P}(\mathrm{QBF} \oplus A) \subseteq \mathrm{NP_B}(\mathrm{QBF} \oplus A) \subseteq \mathrm{P}(\mathrm{SAT} \oplus \mathrm{QBF} \oplus A)$. Since $\mathrm{SAT} \in \mathrm{NP} \subseteq \mathrm{PSPACE}$ and QBF is $\leq_T^P$-complete for PSPACE, $\mathrm{P}(\mathrm{SAT} \oplus \mathrm{QBF} \oplus A) = \mathrm{P}(\mathrm{QBF} \oplus A) = \mathrm{PQUERY}(A)$.

Theorem 4.1. *For every set A,* $\mathrm{NPQUERY_B}(A) = \mathrm{NP_B}(\mathrm{QBF} \oplus A) = \mathrm{PQUERY}(A)$.

5. Remarks

The operators PQUERY( ), NPQUERY( ), and PQH( ) were introduced in [2] and [4] in the context of language-theoretic representations of complexity classes. Each of these operators is a restriction of the PSPACE( )-operator that limits the number of queries that a polynomial space-bounded oracle machine can make in a computation and, hence, limits the amount of information that the machine can obtain from the oracle set. The interest in these operators is based on their use in 'positive relativizations' of the questions "P = ? PSPACE", "NP = ? PSPACE", and "PH = ? PSPACE", that is, in Theorems 2.3 and 2.4. In the present paper, the methods used to prove the technical lemmas leading to Theorems 2.3 and 2.4 represent a substantial economy of effort over the methods used in the original papers. The same thing can be said about Lemma 3.1 which, when combined with Theorem 2.4, yields a very simple proof of Theorem 3.2.

In Theorem 3.2 the statement "for every sparse set S, $\mathrm{PH}(S) = \mathrm{PSPACE}(S)$" does not involve restricting the access that an oracle machine has to the oracle set; instead the oracle set is forced to be 'small', that is, to have low density. Theorem 3.2

appears to be the first place where a positive relativization is obtained by either restricting the size of the oracle set (the equivalence of parts (a) and (b) of Theorem 3.2) or restricting access to the oracle (the equivalence of parts (a) and (c) or of parts (a) and (d)). Lemma 3.1 shows that in the case of the PSPACE( )-operator, restricting size implies restricting access.

Theorems 2.3, 2.6, and 3.2 represent major steps in the study of restricted reducibilities and positive relativizations of questions about complexity classes. For the reader whose primary interest is in this general theme, this paper offers easy access to some of the main results. After studying the proofs in the present paper, such a reader may wish to return to [2], [4], and [10] since there is a great deal more information about the operators PQUERY( ), NPQUERY( ), and PQH( ) in those papers.

References

[1] J. Balcázar, R. Book and U. Schöning, The polynomial-time hierarchy and sparse oracles, *J. Assoc. Comput. Mach.*, to appear.

[2] R. Book, Bounded query machines: On NP and PSPACE, *Theoret. Comput. Sci.* **15** (1981) 27–39.

[3] R. Book, T. Long and A. Selman, Quantitative relativizations of complexity classes, *SIAM J. Comput.* **13** (1984) 461–487.

[4] R. Book and C. Wrathall, Bounded query machines: On NP( ) and NPQUERY( ), *Theoret. Comput. Sci.* **15** (1981) 41–50.

[5] S. Kurtz, On the random oracle hypothesis, *Inform. and Control* **57** (1983) 40–47.

[6] T. Long, On restricting the size of oracles compared with restricting access to oracles, *SIAM J. Comput.* **14** (1985).

[7] T. Long and A. Selman, Relativizing complexity classes with sparse oracles, *J. Assoc. Comput. Mach.*, to appear.

[8] S. Mahaney, Sparse complete sets for NP: Solution of a conjecture of Berman and Hartmanis, *J. Comput. System Sci.* **25** (1982) 130–143.

[9] W. Savitch, Relationships between deterministic and nondeterministic space complexities, *J. Comput. System Sci.* **4** (1970) 177–192.

[10] A. Selman, Xu Mei-rui and R. Book, Positive relativizations of complexity classes, *SIAM J. Comput.* **12** (1983) 565–579.

[11] L. Stockmeyer, The polynomial-time hierarchy, *Theoret. Comput. Sci.* **3** (1976) 1–22.

[12] C. Wrathall, Complete sets and the polynomial-time hierarchy, *Theoret. Comput. Sci.* **3** (1976) 23–33.