1400008446
còpia 1

Self-reducibility

J.L. Balcázar

Report LSI–88–18

SELF-REDUCIBILITY

Preliminary version[†]
November 1988

José L. Balcázar

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

Abstract: New self-reducibility structures are proposed to deal with sets outside the class *PSPACE* and with sets in logarithmic space complexity classes. General properties derived from the definition are used to prove known results comparing uniform and nonuniform complexity classes, and to obtain new ones regarding deterministic time classes, nondeterministic space classes, and reducibility to context-free languages.

Resum: Proposem noves estructures de auto-reduïbilitat per a estudiar problemes que no pertanyen a la classe *PSPACE* i problemes en classes definides per espai logarítmic. Trobem propietats generals derivades de les definicions que fem servir per demostrar resultats coneguts referents a la comparació entre classes uniformes i no uniformes, i per obtenir-ne de nous que es refereixen a classes deterministes amb fites de temps, classes indeterministes amb fites d'espai, i reduïbilitat a llenguatges incontextuals.

1. Introduction

The self-reducibility of some *NP*-complete sets plays a crucial role in the research on reductions to sparse sets. See [5], [9], and [16] for results regarding sparse polynomial time m-reductions from *NP*-complete sets. Results for more general reducibilities appear, among others, in [18]. If polynomial time Turing reductions are considered, the reduction class of the sparse sets can be characterized as the nonuniform class P/poly, and other nonuniform classes have similar characterizations. As part of the study of potential nonuniform properties of certain sets, very interesting consequences for uniform complexity classes are shown in [13], clarifying the relationships between nonuniform and

[†] This work has been partially supported by CIRIT. Some of the results in section 3 were announced at Symposium on Theoretical Aspects of Computer Science, Passau, West Germany, 1987. The results in sections 4 to 7 were announced at Structure in Complexity Theory Third Annual Conference, Washington, D.C., U.S.A., 1988.

uniform computational models. These results are obtained by three seemingly different methods, called there the "round robin tournament" method, the "self-reducibility" method, and the "recursive definition" method.

Later research on structural complexity-theoretic properties of sets precised that the second and third methods clearly correspond to particular cases of the general notion of polynomial time self-reducibility. More precisely, the "self-reducibility" method applies to sets exhibiting a disjunctive self-reducibility structure, while the "recursive definition" method applies to sets having a truth-table self-reducibility structure, both as defined in [14]. It is natural to wonder whether the round robin tournament method can be casted as well into a self-reducibility framework. As expressed by S. Mahaney in [17], p. 106, "it is not clear whether it [the round robin tournament method] might be subsumed by the above method of recursive definition."

We are interested in studying the possibility of such a subsumption. This method applies to sets having a "game" structure corresponding to alternating computations, which should be captured by more general self-reducibility structures. The two main problems are in some sense analogous: to "push up" self-reducibility structures out of $PSPACE$, where all sets with standard self-reducibility lie, and to "push them down" below P, whose sets are trivially self-reducible under the standard definition.

We propose here new definitions of self-reducibility structures allowing to use the same techniques to show properties of complexity classes above $PSPACE$ or below P. We obtain several of the results of [13], together with other results, as corollaries of our main theorems. Thus, we partially answer Mahaney's question by presenting a partial subsumption of the round robin tournament method by a self-reducibility method.

The subsumption presented here is not complete, since not all the result of [13] are obtained. Moreover, the round robin technique (playing all advices against each other) is not used in the proofs. This issue is discussed more in depth in [4], where the technique is applied not only to games but to arbitrary self-reducible sets.

In order to present in a more precise way the contributions of this paper, let us present compactly the results of [13]. (Some definitions are provided below.)

1. *Theorem.*

a/ If $NP \subseteq P/\text{poly}$ then the polynomial time hierarchy collapses to $\Sigma_2 \cap \Pi_2$.
b/ If $PSPACE \subseteq P/\text{poly}$ then $PSPACE = \Sigma_2 \cap \Pi_2$.
c/ If $EXPTIME \subseteq PSPACE/\text{poly}$ then $EXPTIME = PSPACE \neq P$.
d/ If $EXPTIME \subseteq P/\text{poly}$ then $EXPTIME = \Sigma_2 \cap \Pi_2$, which implies $P \neq NP$.
e/ If $NLOG \subseteq DLOG/\log$ then $NLOG = DLOG$.
f/ For every k, if $P \subseteq DSPACE(\log^k n)/\log$ then $P \subseteq DSPACE(\log^k n)$.
g/ If $NP \subseteq P/\log$ then $P = NP$.
h/ If $PSPACE \subseteq P/\log$ then $PSPACE = P$.

Considering self-reducible sets, a general property which allows one to prove such kind of results was isolated in [2], where parts a/ and b/ of theorem 1 were obtained from more general principles. Other results were deduced as well from the same principles. However, as indicated above, under the standard definition self-reducible sets are always in $PSPACE$, while every set in P is trivially self-reducible. The relationships to logarithmic advice were not clear either. Therefore, other parts of theorem 1 could not be obtained.

We define here a self-reducibility property which holds for certain *EXPTIME*-complete sets, and prove an analogous general theorem about those sets from which we derive parts c/ and d/ as corollaries, together with other results. We should point out that this theorem subsumes the proof in [2], so that all the parts of theorem 1 refering to polynomial advice follow uniformly from the same fact. This definition is presented in section 2, together with several properties, and the theorem is proved in section 3.

Then we present a definition of logspace self-reducibility appropriate to work with classes possibly smaller than polynomial time, which is done in section 4, where we prove some properties. This concept of self-reducibility is based on oracle machines which are restricted to query only certain words depending on the input. The definition is highly technical, but its intuitive explanation is (we hope) quite clear. We show in section 5 that this definition has also a general property analogous to those of the just mentioned references, from which we obtain as corollaries parts e/ and f/ of theorem 1.

In section 6 we use the same technique to obtain a new result, comparing uniform P with nonuniform $NLOG$. Here the closure under complements of nondeterministic space classes [12, 21] plays a crucial role. Section 7 is devoted to obtaining new results, of very similar flavor, for the classes $LOG(CFL)$ and $LOG(DCFL)$ of sets reducible in logarithmic space to context-free languages, resp. deterministic context-free languages. We close the paper with a short section of conclusions.

All our sets consist of words over the alphabet $\Gamma = \{0, 1\}$. Larger alphabets will be assumed when necessary. We denote by λ the empty word. The set of all words is denoted Γ^*, and the length of a word x is denoted $|x|$. We assume a fixed easily computable pairing function denoted by angular brackets $\langle , \rangle$. The reader is assumed to be familiar with the standard complexity classes $DLOG$, $NLOG$, P, NP, $PSPACE$, the polynomial time hierarchy, and the like; for definitions see [3]. There is no generally agreed notation for exponential time classes; we denote by E the class of sets decided in time $2^{O(n)}$, and by $EXPTIME$ the class of sets decided in time 2^{n^k} for some k. $EXPTIME$ coincides with $APSPACE$ (alternating polynomial space). See [7] for complexity classes defined by alternating machines, which will be used in sections 2 and 3. SAT denotes the well-known NP-complete problem of deciding the satisfiability of boolean formulas, and QBF the $PSPACE$-complete problem of deciding the truth of quantified boolean formulas. See [3] for undefined notions.

The notation C/F for nonuniform classes is used as in [13]; see also [3]. Thus C/F denotes the class of sets decidable by machines as specified by C with the help of an advice function from F. Formally:

2. *Definition.* Let C be a complexity class and F a family of functions from $\mathbb{N}$ into Γ^*. Then C/F denotes the class of all sets A such that for some $B \in C$ and $h \in F$ it holds that

$$\forall x \quad (x \in A \iff \langle x, h(|x|)\rangle \in B)$$

We will use classes such as $DLOG$, P, $PSPACE$, and the classes of the polynomial time hierarchy in place of C, and the functions bounded in length by polynomials or by logarithms in place of F. More precisely, the class *poly* contains all functions h from $\mathbb{N}$ into Γ^* such that $|h(n)|$ is bounded by a polynomial in n, and the class *log* contains all functions h from $\mathbb{N}$ into Γ^* such that $|h(n)|$ is bounded by $c \cdot \log n$ for some constant c.

2. Polynomial time self-reducibility structures

In this section self-reducibility structures defined by polynomial time machines are considered. The notion studied in [2], defined below, is denoted here ldq-self-reducibility, which stands for "length-decreasing queries". We recall next two facts from the mentioned reference, regarding the complexity classes that bound the region where ldq-self-reducible sets may exist, which are P and $PSPACE$. Then we define wdq-self-reducibility, which stands for "word-decreasing queries", and prove analogous facts for this structure. The fact that the region is broader allows us to present a diagonalization result that shows that not all the sets in that region possess this self-reducibility structure; the analogous statement for ldq-self-reducibility is currently open, even under the assumption $P \neq PSPACE$.

3. *Definition.* A set A is *polynomial time ldq-self-reducible* if and only if there is a polynomial time deterministic oracle Turing machine M such that $A = L(M, A)$, and on each input of length n every word queried to the oracle has length less than n.

Essentially, this concept corresponds to the definition of self-reducibility proposed in [18], the difference consisting in that the condition on the queries is expressed there by requiring that the machine only asks queries smaller than the input in a (fixed) partial order satisfying some natural conditions, namely:

a/ If x is smaller than y then $|x| \leq p(|y|)$ for some polynomial p.

b/ It is decidable in polynomial time whether x is smaller than y.

c/ Every decreasing chain is bounded in length by a polynomial of the length of its maximum element.

Our definition captures the main self-reducible sets (such as some NP-complete, co-NP-complete, and $PSPACE$-complete sets). The more general definition, in turn, is invariant under polynomial time isomorphism, and is required in some particular applications of the concept.

The following properties of the ldq-self-reducible sets are known (see [2]):

4. *Proposition.*

a/ Every set in P is ldq-self-reducible.

b/ If A is ldq-self-reducible then $A \in PSPACE$.

c/ If M witnesses the self-reducibility of both A and B then $A = B$.

Part (a) is immediate since a machine which never queries any oracle satisfies the condition. Part (b) can be proved by solving the queries recursively, and keeping track of the recursion in a polynomially high stack. To prove (c), argue inductively as follows: since on the empty word no queries are possible, $\lambda \in A \iff \lambda \in B$. If A equals B up to length $n-1$, then all the queries on inputs of length n are answered in the same way in both computations of M under respectively A and B, and therefore A equals B also at length n. This argument can be seen also in [2], and will appear repeatedly throughout this paper.

As mentioned above, it is open whether there exists a set in *PSPACE* which is not self-reducible, even assuming $P \neq PSPACE$. Thus, it may be the case that part (b) is actually an equivalence.

The first new polynomial time self-reducibility structure we will study is defined as follows.

5. *Definition.* A set A is *polynomial time wdq-self-reducible* if and only if there is a polynomial time deterministic oracle Turing machine such that $A = L(M, A)$, and on each input x of length n every word queried to the oracle either has length less than n, or has length n and is lexicographically smaller than x.

Observe that this definition allows exponentially long decreasing chains to exist, but keeps the running time of the self-reducing machine polynomially bounded. Of course, every ldq-self-reducibility structure is a wdq-self-reducibility structure. Parts (a) and (c) of proposition 4 hold for this definition. Part (b) has to be adjusted:

6. *Proposition.* If A is wdq-self-reducible then $A \in E$.

Proof. This argument is due to Jacobo Torán. We show inductively that given the set $A \cap \Gamma^{\leq n-1}$, the set $A \cap \Gamma^{\leq n}$ can be constructed in time $2^{c \cdot n}$. Then the theorem follows since $|x|$ many iterations of the construction starting at length 0 allow to decide membership of x in linear exponential time.

To construct $A \cap \Gamma^{\leq n}$, assume that we have in the set variable B initially the set $A \cap \Gamma^{\leq n-1}$. Then, for each word w of length n, simulate the self-reducing machine with oracle B and, if it is accepted, add it to B. By the uniqueness of the set defined by a self-reducing machine, w is added to B if and only if it is in A. After checking all the 2^n words, each in polynomial time, we have the whole of $A \cap \Gamma^{\leq n}$ in the set variable B, and we can use it either to go on with the construction, or just to decide membership of a word of length n. ∎

The polynomial time bound on the self-reducing machine could be relaxed to a linear exponential time bound and the result would still hold. Another observation that follow from this theorem is that the ldq-self-reducible sets are in E; from proposition 4, b/, it only follows that they are in *EXPTIME*, which is weaker.

It appears that wdq-self-reducibility corresponds to E much in the same way as ldq-self-reducibility corresponds to *PSPACE*. It is well-known that $P \neq E$, since the gap between P and E is large enough to allow the time hierarchy theorem to apply. In the next result we show that the same fact yields room for a diagonalization over the wdq-self-reducible sets, proving that the converse of proposition 6 does not hold.

7. *Theorem.* There is a set in E which is not wdq-self-reducible.

Proof. Consider the set A defined by the following stage construction. At stage n, membership to A of the n^{th} word of Σ^* (denoted w_n in the construction) is decided, ensuring that some k^{th} polynomial time clocked machine M_k is not a self-reducing machine for A. We denote p_k a polynomial of the form $n^c + c$, for some c, clocking the running time of M_k. The list L contains the indices of machines that are still candidates to reduce A to itself, and the index k is the smallest index that never entered L up to the moment.

```
Stage 0:
   set $L$ to $\emptyset$.
Stage $n$:
   if $p_k(n) < 2^n$ then add $k$ to the list $L$ and increment $k$;
   search for the smallest index $i \in L$ such that the machine $M_i$,
      on input $w_n$, and using the current segment of $A$ as oracle,
      queries only words smaller than $w_n$;
   if such $i$ is found, then add $w_n$ to $A$ if and only if $M_i$
      rejects it with oracle $A$, and drop $i$ from $L$.
```

Now assume that M_i is a candidate for being a wdq-self-reducing machine for A, so that every query to the oracle set A is smaller than the input. At every stage after i enters L, some smaller candidate is taken care of if one exists, and after finitely many stages precisely index i has to be selected. When this happens, A is defined in a way that ensures that M_i does not define a self-reduction for A.

The set A can be decided as follows: on input $w = w_n$, perform the first $n = 2^{|w|}$ stages, each of which requires cycling through at most $n = 2^{|w|}$ machines, running each of them on a word of length at most $|w|$ for at most $2^{|w|}$ steps by the condition imposed on the indices that enter L. Thus $2^{O(n)}$ time is enough. ∎

It is interesting to note that this result connects the open converse of proposition 4(b) with the open problem of whether $E \subset PSPACE$.

8. *Corollary.* If every set in $PSPACE$ is ldq-self-reducible then $E \not\subset PSPACE$.

Proof. The set constructed in the previous theorem is not wdq-self-reducible, and therefore it is not ldq-self-reducible. If every set in $PSPACE$ is ldq-self-reducible then this set witnesses that $E \not\subset PSPACE$. ∎

To end this section, we show how to relate wdq-self-reducibility to computations of alternating Turing machines, obtaining wdq-self-reducible sets in the deterministic time classes below $EXPTIME$.

9. *Definition.* Let f be a time-constructible function majorized by some polynomial. We denote K_f the following set:

$$K_f = \{\langle m, 0^n, M, I\rangle \mid M \text{ is an alternating machine, } I \text{ is a configuration of } M\text{, and } M \text{ starting at } I \text{ accepts in alternating time } m \text{ and alternating space } f(n)\}$$

We assume as a technical detail that the tupling encodes the number m in the most significant bits of the tuple, so that words coding $\langle m, 0^n, M, I\rangle$ are always smaller than words coding $\langle m', 0^{n'}, M', I'\rangle$ when $m < m'$. The set K_f exhibits a self-reducibility structure:

10. *Lemma.* Under the hypothesis of the previous definition, K_f is wdq-self-reducible, and is m-complete for the class $ASPACE(f)$.

Proof. The self-reducibility structure is inherited from the alternating computations corresponding to the machine M; it can be described as follows: on input $\langle m, 0^n, M, I\rangle$, if I is a final accepting configuration then accept; if I is nonaccepting and $m = 0$ then reject; and if I is not final, and m is not 0, compute the two branching configurations I_0, I_1 of M reachable from I, query the oracle about $\langle m-1, 0^n, M, I_0\rangle$ and $\langle m-1, 0^n, M, I_1\rangle$, and accept according to the answers and to the state (existential or universal) of I. Encoding m into the most significative digits of $\langle m, 0^n, M, I\rangle$ ensures that both queries are smaller than the input in the lexicographic order.

The membership to $ASPACE(f)$ is straightforward, since an alternating machine can just set itself up in configuration I and simulate M. Completeness follows from the following reduction: if M is an alternating f-space machine, and $I(x)$ denotes the initial configuration of M on input x of length n, then x is accepted by M if and only if $\langle 2^{f(n)}, 0^n, M, I(x)\rangle$ is in K_f. The tuple can be written down in polynomial time since f is majorized by some polynomial. ∎

To indicate the use we will make of this result, assume that such a function f grows faster than $\log n$, and let F be a family of functions such that $2^f \in \Theta(F)$ and $F \subset O(2^{f^{O(1)}})$. Then, by the results of [7], it can be shown that K_f is complete for $DTIME(F)$, which yields a wdq-self-reducible complete set for this class. The properties of wdq-self-reducible sets we want to exploit are presented in the next section.

3. Polynomial advice classes

This section is devoted to obtain results from the definition of wdq-self-reducibility regarding the interconnection between uniform and nonuniform complexity classes, in particular those defined by polynomially long advices. Our aim is to find a framework in which parts a/, b/, c/, and d/ of theorem 1 appear as natural consequences of more general facts about self-reducible sets. In particular, we want to show that the results about $EXPTIME$ do not depend on this particular class, but on the fact that wdq-self-reducible complete sets exist for the class. We show that the same properties hold for other deterministic time classes.

In [2], the following result is shown:

11. *Theorem.* If $A \in \Sigma_i$/poly and A is ldq-self-reducible, then $\Sigma_2(A)$ is included in Σ_{i+2}. In particular, $A \in \Sigma_{i+2}$.

This property allows one to prove parts a/ and b/ of theorem 1 using the ldq-self-reducibility of SAT and QBF, as well as some other results regarding the collapse of the relativized polynomial time hierarchy:

12. *Corollary.*
  a/ If $NP \subseteq P$/poly then $PH = \Sigma_2$.
  b/ If $PSPACE \subseteq P$/poly then $PSPACE = \Sigma_2$.

Both statements follow from theorem 11 by taking SAT and QBF respectively for A, and using the facts that if $\Sigma_3 = \Sigma_2$ then $PH = \Sigma_2$, and that Σ_2 is closed under m-reducibility.

The key point now is that theorem 11 carries over to wdq-self-reducible sets. Moreover, membership in $PSPACE$/poly is a meaningful condition for wdq-self-reducible sets, and we can extend the result to this nonuniform class.

13. *Theorem.*

a/ If $A \in \Sigma_i$/poly and A is wdq-self-reducible then $\Sigma_2(A)$ is included in Σ_{i+2}. In particular, $A \in \Sigma_{i+2}$.

b/ If $A \in PSPACE$/poly and A is wdq-self-reducible then $A \in PSPACE$.

Proof. a/ Membership to A can be expressed as follows: x is in A if and only if there is a polynomially long advice string y which is correct for A up to length $|x|$, such that $\langle x, y\rangle \in B$, where B is a Σ_i predicate. The fact that y is correct for A up to length n means the following:

$$\forall z \, (|z| \leq n) \quad (z \in A \iff \langle z, y\rangle \in B)$$

This correctness assertion can be expressed as a predicate $Corr(y, n)$ with just an universal quantifier, using the self-reducing machine M of A as follows. Let M' be the machine that on input $\langle z, y\rangle$ simulates M on input z, and when M queries about w it queries about $\langle w, y\rangle$. Then y is correct if and only if

$$\forall z \, (|z| \leq n) \quad (\langle z, y\rangle \in B \iff \langle z, y\rangle \in L(M', B)\}$$

This equivalence follows from the unicity of the self-reducible set defined by M (proposition 4(c)), since the last assertion says that the set of words z of length up to n such that $\langle z, y\rangle \in B$ is coherent with the self-reducing machine M for A.

Now let D be any set in $\Sigma_2(A)$, so that

$$x \in D \iff \exists u \forall v \langle x, u, v\rangle \in L(M'', A)$$

where both quantifiers are polynomially bounded and M'' is a deterministic polynomial time machine. Transform it into M''' that on input $\langle x, u, v, y\rangle$ acts as M'' on $\langle x, u, v\rangle$, but queries $\langle w, y\rangle$ when M'' queries w. Thus, if y is a correct advice up to a certain polynomial $p(|x|)$ the answers are consistent with A and M''' accepts $\langle x, u, v, y\rangle$ with oracle B if and only if M'' accepts $\langle x, u, v\rangle$ with oracle A. Now,

$$x \in D \iff \exists u \exists y \, (Corr(y, p(|x|)) \wedge \forall v \langle x, u, v, y\rangle \in L(M''', B))$$

which is a Σ_{i+2} predicate since B is Σ_i.

b/ It is very similar. Now membership to A can be expressed as: x is in A if and only if there is a polynomially long advice string y which is correct for A up to length $|x|$, such that $\langle x, y\rangle \in B$, where B is now in $PSPACE$. The fact that y is correct for A up to length n can be tested in $PSPACE$ by the same universal predicate as before using M' and the $PSPACE$ algorithm for solving the queries to B. Cycling through all the polynomially long advices until finding a correct one and using it to decide membership to A yields a polynomial space decision procedure for A. ∎

Using this result, we will get results from the existence of wdq-self-reducible sets in deterministic time classes proved in lemma 10. In particular, the following theorem allows us to apply theorem 13 to many classes below $EXPTIME$.

14. *Theorem.* Let f be as in lemma 10, and assume that f grows faster than $\log n$. Let F be a family of functions such that $2^f \in \Theta(F)$ and $F \subset O(2^{f^{O(1)}})$. Then:

a/ $DTIME(F) \subseteq \Sigma_i/\text{poly}$ implies $DTIME(F) \subseteq \Sigma_{i+2}$.

b/ $DTIME(F) \subseteq PSPACE/\text{poly}$ implies $DTIME(F) \subseteq PSPACE$.

Proof. As indicated above, under these hypothesis the set K_f is complete for $DTIME(F)$. Therefore, it must belong to the nonuniform class in the left hand side of each implication. Since it is wdq-self-reducible, by theorem 13 it belongs to the uniform class in the right hand side of each implication. The statements follow from the closure of the uniform classes under polynomial time reductions. ∎

We can state now as a corollary parts c/ and d/ of theorem 1:

15. *Corollary.*

If $EXPTIME \subseteq P/\text{poly}$ then $EXPTIME \subseteq \Sigma_2 \cap \Pi_2$.

If $EXPTIME \subseteq PSPACE/\text{poly}$ then $EXPTIME \subseteq PSPACE$.

Proof. Take $f(n) = n$. Since all hypothesis of theorem 14 hold for $F = \{2^{n^k}\}$, both inclusions follow from it. ∎

Note that right to left inclusions in the right hand sides are immediate, and therefore the corresponding inclusions are actually equalities. Also note that corollary 12, proven in [2] using the same principle for ldq-self-reducibility, can be seen also as a consequence of theorem 14, since ldq-self-reducibility implies wdq-self-reducibility. Thus, parts a/ to d/ of theorem 1 follow uniformly from theorem 14.

We can present now other classes to which theorem 14 applies. An immediate case is E. Indeed, all the classes appearing in the previous corollary are closed under polynomial time m-reducibility, and $EXPTIME$ is the closure of E under the same reducibility. Thus, $E \subset P/poly$ implies $EXPTIME \subseteq P/poly$, and the corollary holds for E as well as for $EXPTIME$.

More classes for which we get such results are presented in the following corollary. The proof amounts to setting f to either $\log^2 n$ or $\log n \cdot \log\log n$.

16. *Corollary.* Let F be any of the following classes, where k is fixed:

$$\{n^{c\cdot\log^k n} \mid c > 0\}, \quad \{n^{\log^c n} \mid c > 0\}, \quad \{n^{c\cdot\log n} \mid c > 0\}, \quad \{n^{c\cdot\log\log n} \mid c > 0\}$$

Then:

If $DTIME(F) \subset P/\text{poly}$ then $DTIME(F) \subset \Sigma_2 \cap \Pi_2$.

If $DTIME(F) \subset PSPACE/\text{poly}$ then $DTIME(F) \subset PSPACE$.

To end, we show how to obtain the consequence $P \neq NP$ as in theorem 1, part d/, from weaker hypothesis.

17. *Corollary.* Let $\varepsilon(n)$ be a slowly growing, unbounded function such that $(\log n)\cdot\varepsilon(n)$ is time-constructible, and such that $DTIME(n^{\varepsilon(n)}) \neq P$ (such functions can be obtained as in the time hierarchy theorem). If $DTIME(n^{\varepsilon(n)})$ is included in P/poly, then $P \neq NP$.

Proof. By theorem 14, under these hypothesis, $DTIME(n^{\varepsilon(n)})$ is included in Σ_2. Therefore $P \neq \Sigma_2$, which implies $P \neq NP$. ∎

Note that $\varepsilon(n)$ is assumed to be as small as desired, given that $DTIME(n^{\varepsilon(n)}) \neq P$. Hence, the hypothesis "$DTIME(n^{\varepsilon(n)})$ is included in P/poly" can be read as "a polynomially long advice saves computation time". The corollary shows that if this is the case, then $P \neq NP$.

4. Logspace self-reducibility

Some technical concepts are required for setting up a concept of self-reducibility in logarithmic space. We are going to present the appropriate model of oracle Turing machine, which is based on a property similar to a characterization given in [19] of certain nondeterministic oracle machines. The property that identifies our model is that all the queries are small variations of the input; more precisely, every query is equal to the input in all but the $\log n$ last symbols, where n is the length of the input.

Let us motivate this machine model, by explaining why more natural definitions present important drawbacks. If we look for a definition of logspace self-reducibility, determinism and log work space are clearly out of discussion; but the way of bounding the oracle tape raises some problems, as can be expected by those readers aware of the discussions regarding these issues.

Indeed, a logspace self-reducing machine in which the oracle tape obeys the space bound does not furnish any structure in the accepted set, since a set defined by such a self-reduction procedure is already in $DLOG$. To see this, consider a machine that simulates the self-reducing machine, and when a query is made pushes the configuration onto a stack and starts solving the query; when a final state is reached with nonempty stack, it pops the stack and continues with oracle answer in accordance with the final state, and when a final state is reached with empty stack it stops in the corresponding state. The space required for the stack decreases very fast, since each query is logarithmically smaller than the previous one and thus the computations on a query are made in a logarithmically smaller space. If the space for the outermost computations is $\log n$, then the space needed for the remaining part of the stack is less than $(\log \log n)^2$, which is asymptotically smaller than $\log n$. Thus a constant factor compression of tape yields a $DLOG$ decision algorithm.

On the other hand, if the oracle tape is not bounded then $PSPACE$-complete sets can be obtained. This is easy to see from the self-reducibility structure of QBF, in which the queries are constructed scanning and copying the input, substituting a constant for one of the variables; to identify which one, its index can be stored in logarithmic space.

The reason behind this problem is that the actual amount of resources used by the self-reduction procedure is in some sense less important than the mathematical structure enforcing its finiteness. Let us informally speak of the "depth" of a preorder on Γ^* as the maximum length of its descending chains, as a function of the size of the maximum word in the chain; and of the "size" of a preorder as the number of elements smaller than a given one, as a function of the size of this one. Then, as the QBF example shows, logarithmic work space self-reducibility is enough for reaching $PSPACE$-complete sets, provided that the query condition gives a "polynomial depth" of the recursion. Comparing with the wdq-self-reducibility, we see that while a polynomial depth preorder corresponds to polynomial space, an "exponential size" recursion tree yields somehow E-complete sets, and therefore it is natural to suggest that a "polynomial size" preorder be used for the sets in P. However, as the first example shows, plainly bounding the length of the queries by a logarithm is not enough. The reason is that in this case the polynomially many elements that can be queried on each input of length n are just the words of length at most $\log n$: the *same set for all* the exponentially many words of that length! A more flexible structure is needed, in which it should be possible that the recursion tree depends substantially on the input. Thus the model to consider next is the model described above, in which queries are equal to the input in all but the $\log n$ last symbols.

Another view of this model, which will become clearer after the examples presented in propositions 25 and 27, is the following: the input consists of two parts, a main data structure and a fixed number of elements of it, given in some sense by logarithmic length pointers; then the queries consist of the same main data structure, which is invariant in the whole self-reduction tree, and some other pointers coded in the logarithmically long varying section. Still another very informal view, which we use just to motivate the name, is that the machine can be thought of as somebody that is given a standard sentence, learnt by heart, which allows him to start speaking, leaving his natural silent state, in order to ask afterwards for a very small piece of information. Thus we call them *shy machines*. The following notation will be useful for the formal definition.

18. *Notation.* Let x and w be words such that $|w| = \log |x|$. We denote $sub(x, w)$ the word resulting from substituting the word w for the last $\log |x|$ symbols of x.

Notice that $sub(x, w)$ is a word of the same length as x, and that they can be compared according to the lexicographic criterion.

19. *Definition.* A *shy machine* is a logspace oracle Turing machine, with no bound on the oracle tape, such that on input x every query is of the form $sub(x, w)$ for some w of length $\log |x|$.

A point that should be observed is that on input $sub(x, w)$ the queries made by M are themselves again of the form $sub(x, v)$, since $sub(sub(x, w), v) = sub(x, v)$.

We define next logarithmic space self-reducibility in terms of shy machines. The self-reducibility structure is enforced to be well-founded via a restriction analogous to that of the "word decreasing queries" self-reducibility proposed in previous sections, in the sense that we use a lexicographical comparison in the condition on the queries.

20. *Definition.* A set A is *self-reducible in logarithmic space* (logspace self-reducible for short) if and only if there is a logarithmic space shy machine M such that $A = L(M, A)$, and on every input x every word queried by M is lexicographically smaller than x.

The following properties correspond to proposition 4 for the new definition. They locate logspace self-reducible sets, and state the uniqueness of the self-reducible set defined by a given shy machine M, respectively.

21. *Proposition.*

a/ Every set in $DLOG$ is logspace self-reducible.

b/ Every logspace self-reducible set is in P.

Proof. Part a/ is immediate, since a logspace machine that does not query is a shy machine and observes whichever condition is required on the queries. To see part b/, a polynomial time algorithm for deciding membership of x is obtained by simulating the self-reducing machine in lexicographic order on all the polynomially many words $sub(x, u)$, recording all the answers for later use; in this way, all the answers to queries have been answered before they are needed. When x itself is reached we know the answer and can stop. Observe the similarity of this "dynamic programming" argument with the proof of proposition 6. ∎

The unicity of the self-reduced set can be shown separately for each length:

22. *Proposition.* Let M be a shy machine which always queries words smaller than the input in the lexicographical order, and let $A = L(M, A)$. Let B be a set of words, all of them of length n, such that for every word x of length n, x is in B if and only if x is in $L(M, B)$. Then B is precisely the subset of all the words of length n in A.

Proof. On the smallest word of length n, M cannot query the oracle. Therefore it cannot show a difference between $L(M, A)$ and $L(M, B)$, and it is either both in A and B or outside both of them. Now, for any word w of length n, suppose that A and B coincide on all smaller words of that length. Then the behavior of M on w is identical for both oracles, and w must be either accepted, and therefore belong to both A and B, or rejected, and therefore belong to neither of them. Thus, inductively, A and B cannot differ at length n. Again, observe the similarity of this argument with proposition 4(c). We will use this inductive argument in the forthcoming sections. ∎

Observe that the argument for each length is independent of the other lengths due to the fact that shy machines always query words of the same length as their input, but the consequence is the unicity of the full self-reduced set of the machine:

23. *Corollary.* If M is a shy machine which always queries words smaller than the input in the lexicographical order, $A = L(M, A)$, and $B = L(M, B)$, then $A = B$.

We show next that logspace self-reducible sets exist; our examples are quite natural encodings of complete sets for certain complexity classes.

24. *Definition.* Let AGAP (standing for Acyclic Graph Accesibility Problem) be the set of all words of the form $G\#s\#t$ where G encodes an acyclic graph, s and t are nodes of G, and there is a path in G leading from s to t. We require further that the nodes are labeled according to a topological sort in such a way that the label of each node is a number of length $\log |G\#s\#t|$.

Note that the requirement of G being an encoding of a graph topologically sorted only means that the numbering of the nodes is such that the source of each edge has a number smaller than its target: this can be tested easily in logspace and implies acyclicity, so AGAP is in $NLOG$. Using standard techniques, it is not difficult to see that AGAP is complete for $NLOG$ under logspace reductions. (In order to obtain an acyclic graph, start from a $NLOG$ machine that counts the number of steps performed during its computation: this guarantees absence of loops.)

25. *Proposition.* AGAP is logspace self-reducible.

Proof. On input $G\#s\#t$, if s is a predecessor of t in G then accept, otherwise query the oracle about all the words $G\#s\#t'$ where t' is a predecessor of t in G. If the alphabet is appropriately chosen, the queries have the required form for a shy machine, and the topological sort implies that the queries are smaller than the input. Clearly logspace suffices. ∎

Our next example is a particular encoding of the circuit value problem.

26. *Definition.* Let CVP (standing for Circuit Value Problem) be the set of all words of the form $u\#C\#g$, where u is a binary string, C is an encoding of a fan-in 2 boolean circuit with $|u|$ inputs, and g is a gate of C, which we designate as output gate, and which takes value 1 on input u. We require that each gate is labeled by a number of length $\log |u\#C\#g|$, and that the label of each gate is greater than the labels of its two input gates.

It can be seen that CVP is complete for P under logspace reductions [15]. Our requirements about the encoding are irrelevant for the proof. We have the following property.

27. *Proposition.* CVP is logspace self-reducible.

Proof. It is similar to the previous one, but the self-reducibility is no longer disjunctive. On input $u\#C\#g$, if g is an input gate then check the corresponding symbol of u; otherwise, let g_1 and g_2 be the gates that are inputs to g, query the oracle about $u\#C\#g_1$ and $u\#C\#g_2$ to obtain their respective values, and apply to the answers the boolean function corresponding to gate g. For an appropriate alphabet the queries have the form needed by a shy machine, and the requirement on the numbering guarantees that they are lexicographically smaller. ∎

The self-reducibility of these sets will be used in the next sections. Other logspace self-reducible sets are presented in section 7.

5. Deterministic logspace with advice

In this section we show a property of logspace self-reducible sets which yields as particular cases parts e/ and f/ of theorem 1. It is very similar to the properties of self-reducible sets presented in previous sections. This property is stated as follows:

28. *Theorem.* Let A be a logspace self-reducible set. If $A \in DLOG/\log$ then $A \in DLOG$.

Proof. We show how to decide A in deterministic logarithmic space. The algorithm just cycles over all possible advices of the appropriate length, searching for a correct one, and when found it uses the $DLOG$ algorithm with this advice. The self-reducibility structure is used to check the correctness of each possible advice.

More formally, let $A = L(M, A)$ where M is a shy machine which witnesses the logspace self-reducibility of A. Further, let M' be a logspace machine and let h be such that

$$\forall x\, (x \in A \iff \langle x, h(|x|)\rangle \in L(M'))$$

given by the fact that A belongs to $DLOG/\log$. Without loss of generality we assume that the alphabet is large enough so that $|h(n)| = \log n$. We say that an advice w of length $\log n$ is *correct for* x where $|x| = n$ if and only if

$$\forall u \ (sub(x,u) \in A \iff \langle sub(x,u), w\rangle \in L(M'))$$

where u ranges over the words of length $\log |x|$; i.e. if w can be used without harm instead of the actual value of h in order to decide x and the words that M could query on x. Consider now the following algorithm.

```
input $x$;
for each word $w$ of length $\log|x|$ do
   check (using the subroutine below) that $w$ is a correct advice for $x$;
   if it is then exit the for loop;
accept if and only if $\langle x, w\rangle \in L(M')$.
```

By the definition of correctness, this program decides A provided that the subroutine works properly, since at least the value $h(|x|)$ will be found (and possibly some other correct one). The correctness of the candidate advice can be tested as follows.

```
for each word $u$ of length $\log|x|$ do
   simulate $M$ on input $sub(x,u)$;
   whenever $M$ queries about $sub(x,v)$, answer YES if and only if
         $\langle sub(x,v), w\rangle \in L(M')$;
   check that $M$ accepts $sub(x,u)$ if and only if $\langle sub(x,u), w\rangle \in L(M')$;
   if so, return YES, else return NO.
```

The correctness of this subroutine is argued by the same argument used in the proof of proposition 22. ∎

Now we can derive easily part e/ of theorem 1 as announced. Just apply theorem 28 to the set AGAP, which was shown in the previous section to be logspace self-reducible and $NLOG$-complete.

29. *Corollary.* If $NLOG \subseteq DLOG/\log$ then $NLOG = DLOG$.

Similarly, theorem 28 can be applied to CVP, yielding the following.

30. *Corollary.* If $P \subseteq DLOG/\log$ then $P = DLOG$.

It is easy to see that if in theorem 28 the class $DSPACE(\log^k n)$ is substituted for $DLOG$ (keeping the advice logarithmically bounded) the proof carries through. This yields as a corollary part f/ of theorem 1.

31. *Corollary.* For every k, if $P \subseteq DSPACE(\log^k n)/\log$ then $P \subseteq DSPACE(\log^k n)$.

6. Nondeterministic logspace with advice

The results in the previous section indicate that for classes having a complete logspace self-reducible set, being included in nonuniform logarithmic space amounts to being included in the corresponding uniform class *DLOG*; i.e. the advice is in some sense useless. It is natural to wonder whether a similar result can be obtained under the hypothesis that P is included in nonuniform *nondeterministic* logarithmic space: is it possible again to "get rid of" the advice and show an equality with the corresponding uniform class?

In this section we prove a theorem that allows one to obtain precisely this result, thus completing in some sense the "map" of implications between the uniform and nonuniform classes P, $NLOG$, and $DLOG$. The proof is similar to that of theorem 28, and requires the use of the complement closure theorem [12,21] and of a consequence of it. More precisely, we need the following property, which is easy to prove using the results of [12] or [21].

32. *Proposition.* $DLOG(NLOG) = NLOG$.

Now we can state the main result of this section.

33. *Theorem.* Let A be a logspace self-reducible set. If $A \in NLOG/\log$ then $A \in NLOG$.

Proof. Let $A = L(M, A)$ where M is a shy machine which witnesses the logspace self-reducibility of A. Further, let M' be a nondeterministic logspace machine, and let h be such that

$$\forall x\, (x \in A \iff \langle x, h(|x|)\rangle \in L(M'))$$

which exist since $A \in NLOG/\log$. Again, we assume that the alphabet is large enough so that $|h(n)| = \log n$. The notion of *correct advice* for a given word x is defined exactly as in the deterministic case:

$$Corr(x, w): \quad \forall u\, (sub(x, u) \in A \iff \langle sub(x, u), w\rangle \in L(M'))$$

We claim that the predicate $Corr(x, w)$ can be tested in nondeterministic log space. We will do this by considering the following deterministic logspace oracle machine M'':

```
input $\langle y, w\rangle$;
simulate $M$ on $y$,
   on query $z$, query $\langle z, w\rangle$.
```

This machine is designed to use $L(M')$ as oracle. We show our claim by proving the following equivalence:

$$(*) \quad Corr(x, w) \iff$$
$$\Big[\forall u\, (|u| = \log |x|)\, \big(\langle sub(x, u), w\rangle \in L(M'', L(M')) \iff \langle sub(x, u), w\rangle \in L(M')\big)\Big]$$

Cycling to test the universal quantifier requires log space; the quantified predicate is in $DLOG(NLOG)$, and therefore in $NLOG$ by proposition 32. Thus, the predicate $Corr(x, w)$ can be decided in $NLOG$. Let us now prove $(*)$.

Assume that $Corr(x, w)$ is true. Then, since M is shy, all queries of M'' on $\langle sub(x,u), w\rangle$ are of the form $\langle sub(x,v), w\rangle$ and therefore, by the correctness of w, correctly answered by $L(M')$. Thus $\langle sub(x,u), w\rangle \in L(M'', L(M'))$ if and only if $sub(x,u) \in A$, and again by the correctness if and only if $\langle sub(x,u), w\rangle \in L(M')$.

Conversely, if the right hand side of $(*)$ holds then the set of words $sub(x,u)$ such that $\langle sub(x,u), w\rangle \in L(M')$ is consistent with the self-reducing machine M. By the inductive argument of the proof of proposition 22, we obtain that $sub(x,u) \in A \iff \langle sub(x,u), w\rangle \in L(M')$, and therefore w is correct for x. This proves the claim that correctness can be decided in $NLOG$.

Now it is immediate to prove the theorem: on input x, guess a correct advice w, check its correctness in $NLOG$, and use it to decide whether $x \in A$ by simulating M' on $\langle x, w\rangle$. ∎

In the same manner as in the preceding section, this theorem can be applied to CVP:

34. *Corollary.* If $P \subseteq NLOG/\log$ then $P = NLOG$.

Once more, the proof carries through if a class of the form $NSPACE(\log^k n)$ is substituted for $NLOG$ (but again keeping the advice logarithmically bounded). We obtain:

35. *Corollary.* For every k, if $P \subseteq NSPACE(\log^k n)/\log$ then $P \subseteq NSPACE(\log^k n)$.

7. Reducibility to context-free languages

An interesting class contained in P is the closure of the class of context-free languages under logspace m-reducibility, denoted $LOG(CFL)$, which has been recently shown to be closed under complements [6]. Its analog class for the deterministic context-free languages is $LOG(DCFL)$. They have been characterized in [20] in terms of multihead pushdown automata and logspace polynomial time auxiliary pushdown automata. Their relationship to the logspace complexity classes is obviously related to the open question of whether context-free languages can be decided in logarithmic space. We show here that languages in these classes can be captured by certain logspace self-reducible sets, and therefore results like corollaries 30 and 34 can be obtained for them.

Our results are based on a smart technique presented in [8], which is based in turn on the decision procedure for context-free languages of [1]. Reference [8] applies this technique to auxiliary pushdown automata. Although we apply it to pushdown automata as in [1], we follow the approach of the former since it is closer to our goal: we want to make apparent the logspace self-reducibility structure underlying the technique. For the proof of our main theorem in this section we will require the following two lemmas.

36. *Lemma.*

a/ There is a pushdown automaton M_1, with no λ-transitions, which accepts by empty store, such that $L(M_1)$ is complete for $LOG(CFL)$ under logspace m-reducibility.

b/ There is a deterministic pushdown automaton M_2, with no λ-transitions, which accepts by empty store, such that $L(M_2)$ is complete for $LOG(DCFL)$ under logspace m-reducibility.

Proof. a/ The hardest context-free language of Greibach [10] does not contain the empty word, and is complete under homomorphism for the class of context-free languages that do not contain the empty word; therefore it is logspace m-complete for $LOG(CFL)$. By a known result of automata theory (see [11], theorem 5.5.1), it is accepted by empty store by a pushdown automaton with no λ-transitions.

b/ In [20], a deterministic context-free language is exhibited that is logspace m-complete for the class of deterministic context-free languages (lemma 8 and proof of lemma 9 of [20], see also footnote in page 413). Also, in the same reference, it is shown (lemma 7) that every deterministic context-free language is logspace m-reducible to a deterministic context-free language recognized by empty store by a deterministic pushdown automaton with no λ-transitions. Our claim follows from the transitivity of the logspace m-reducibility. ∎

Let $\mathrm{AuxPDA}_{pt}(\log)$ denote the class consisting of those sets decidable by nondeterministic logspace auxiliary pushdown automata in polynomial time, and similarly $\mathrm{AuxDPDA}_{pt}(\log)$ for deterministic logspace auxiliary pushdown automata.

37. *Lemma.* The following equalities hold:

$$LOG(CFL) = \mathrm{AuxPDA}_{pt}(\log)$$
$$LOG(DCFL) = \mathrm{AuxDPDA}_{pt}(\log)$$

We omit the proof: it is theorem 1 in [20]. Now we present our main theorem of this section. For closely related material and analogous notation and properties, see the proof of theorem 1 of [8], part (b) implies (c).

38. *Theorem.* Let M be a pda with no λ-transitions which accepts by empty store. There is a set $A \in LOG(CFL)$ which is logspace self-reducible, such that $L(M) \in DLOG(A)$. Furthermore, if M is deterministic then $A \in LOG(DCFL)$.

The proof requires to develop some definitions and notation. Given the pushdown machine M as in the theorem, a *surface configuration* of M on input w is a triple $\langle p, q, Z\rangle$; p is the position of the input tape head, q is a state of M, and Z is the top symbol in the pushdown. A pair of surface configurations P, Q is *realizable* if and only if there is a partial computation of M on input w starting at a configuration c_1 corresponding to surface configuration P, ending at a configuration c_2 corresponding to surface configuration Q, and such that the height of the pushdown is the same in c_1 and in c_2, and during that computation this height never drops below this threshold. Note that realizability depends on the input.

We encode pairs of surface configurations on input w as strings of length $\log |w|$ over a large enough alphabet. We assume that this encoding is such that the following condition holds: if in surface configuration Q_1 the input tape head is scanning a symbol strictly at the left of the symbol scanned in surface configuration Q_2, then the encoding of the pair $\langle P_1, Q_1\rangle$ is smaller in the lexicographic ordering than that of $\langle P_2, Q_2\rangle$ for every P_1, P_2. This is attained by encoding the position of the tape head in component Q into the most significant digits of $\langle P, Q\rangle$.

The key to the self-reducibility structure is given by the following definition [8].

39. *Definition.* Pairs $\langle P_1, Q_1\rangle$ and $\langle P_2, Q_2\rangle$ *yield* pair $\langle P_3, Q_3\rangle$ if and only if $P_1 = P_3$ and either:

(i) $Q_1 = P_2$ and M goes in one step from Q_2 to Q_3 without changing the pushdown, or

(ii) M goes in one step from Q_1 to P_2 pushing a symbol Z, and M goes in one step from Q_2 to Q_3 popping the same symbol Z.

The core of the proof is in our next lemma.

40. *Lemma.* Starting from all identity pairs $\langle P, P\rangle$ and iterating the "yield" relation, exactly the set of all realizable pairs is obtained.

Proof. Identity pairs are clearly realizable, and the definition of the "yield" relation implies that realizable pairs always yield realizable pairs. Conversely, suppose that the pair $\langle P, Q\rangle$ is realizable via a computation $P = P_1, P_2, \ldots, P_t = Q$. If $t = 1$ then $P = Q$ and the pair is a base identity pair. Assume that $t > 1$ and, inductively, that computations shorter than t can be obtained by the "yield" relation. If the transition from P_{t-1} to P_t does not change the pushdown, since we can assume by induction hypothesis that $\langle P_1, P_{t-1}\rangle$ has been obtained from the "yield" relation, part (i) yields $\langle P, Q\rangle$. The definition of realizability prevents the transition from P_{t-1} to P_t from being a pushing move; thus, assume that the pushdown is popped, and consider the first pushing move in the partial computation, say from P_i to P_{i+1}. Inductively, $\langle P_1, P_i\rangle$ and $\langle P_{i+1}, P_{t-1}\rangle$ are realizable and therefore can be obtained from the "yield" relation. Applying part (ii) of the definition of "yield" gives $\langle P, Q\rangle$. ∎

Of course, in order to decide whether two pairs yield another the input must be known. An important point in the previous proof is that every realizable pair (excepting identity pairs, of course) can be obtained by applying the "yield" relation to pairs having strictly smaller encodings, due to the fact that the pushdown machine M has no λ-transitions.

We are now ready to prove theorem 38.

Proof. (Of theorem 38). The set whose existence is asserted in the statement is the set A formed by all the words of the form $w\#\langle P, Q\rangle$, such that on input w the pair $\langle P, Q\rangle$ is realizable.

$$A = \{w\#\langle P, Q\rangle \mid \text{on input } w \text{ the pair } \langle P, Q\rangle \text{ is realizable}\}$$

We show that the theorem holds. To see that $A \in LOG(CFL)$, we argue that A is accepted in linear time by a logspace AuxPDA, which on input $w\#\langle P, Q\rangle$ sets up itself on configuration P and simulates M, keeping in a counter the height of the stack, and checking that Q is reached with no extra symbols left on the stack. By lemma 37, $A \in LOG(CFL)$. Moreover, if M is deterministic then the AuxPDA is deterministic also, and therefore $A \in LOG(DCFL)$.

To see that A is logspace self-reducible, we take advantage of the characterization given by the "yield" relation. Consider a shy machine that on input $w\#\langle P, Q\rangle$ accepts if $P = Q$, else searches for smaller pairs that yield $w\#\langle P, Q\rangle$ and queries the oracle to find whether they are realizable. It is easy to see that the queries have the right form; its correctness follows from lemma 40.

Finally, $L(M)$ is decidable in logarithmic space with oracle A by checking whether a realizable pair exists starting at the initial configuration of M and ending at an accepting configuration. This completes the proof. ∎

As applications of this theorem, we obtain:

41. *Corollary.*
 (a) If $LOG(CFL) \subseteq DLOG/\log$ then $LOG(CFL) = DLOG$.
 (b) If $LOG(DCFL) \subseteq DLOG/\log$ then $LOG(DCFL) = DLOG$.
 (c) If $LOG(CFL) \subseteq NLOG/\log$ then $LOG(CFL) = NLOG$.
 (d) If $LOG(DCFL) \subseteq NLOG/\log$ then $LOG(DCFL) = NLOG$.

Proof. Apply theorem 38 to the pushdown automata described in lemma 36 to obtain logspace self-reducible sets complete respectively for $LOG(CFL)$ and $LOG(DCFL)$. Then the results follow from theorems 28 and 33. ∎

As a final remark, notice that this result does not say that if context-free languages can be decided by nonuniform logspace models then they can be decided by uniform logspace models: the hypothesis required is that the whole class $LOG(CFL)$ is included in $DLOG/\log$. The reason why the proof does not work from a weaker hypothesis, like context-free languages being in $DLOG/\log$, is that this second class is not closed under logspace reducibility, since each advice is valid for only one length and a reducibility may map the words of a given length to words of polynomially many different lengths. Therefore the membership of context-free languages to $DLOG/\log$ does not guarantee membership to the nonuniform class of the logspace self-reducible complete set for $LOG(CFL)$.

8. Conclusions

This work was based on the definition of polynomial time self-reducibility, which we have called here ldq-self-reducibility since it is characterized by queries that decrease in length. This definition was useful as a tool to study structural properties of sets between P and $PSPACE$. We have developed two new notions of self-reducibility. The first one is wdq-self-reducibility, where queries are only required to decrease in the lexicographic ordering, and gives a definition which captures sets in E, thus possibly out of $PSPACE$. The second one corresponds to logspace self-reducibility, and is refined enough to discriminate among sets in P; this definition required to develop an appropriate model of oracle machine. We have demonstrated that the most interesting properties of ldq-self-reducible sets in nonuniform classes hold as well, for appropriate classes, for these two new forms of self-reducibility.

Thus, our definitions allow one to obtain known and new consequences in the comparison between uniform and nonuniform classes. Using wdq-self-reducibility, we obtain a general property from which all the statements of theorem 1 that correspond to polynomial advice classes follow uniformly; among them, two properties that were derived in [2] from the similar property for ldq-self-reducible sets. Other new consequences are obtained for other complexity classes.

Using logspace self-reducibility, we obtain similar but stronger general properties, so that if a nonuniform model corresponding to deterministic (resp. nondeterministic) logarithmic space is able of deciding logspace self-reducible sets then the nonuniformity capability can be "switched off", and equalities such as $P = DLOG$ or $P = NLOG$ follow. Similar results compare $NLOG$ to $DLOG$, $LOG(CFL)$ and $LOG(DCFL)$ to $NLOG$ and $DLOG$, and P to $DSPACE(\log^k n)$ and to $NSPACE(\log^k n)$. Therefore, parts e/ and f/ of theorem 1 are also obtained among other results. Comparing both definitions, we see that we have two properties similar in spirit, one of them appropriate to work between P and E and the other one appropriate to work in classes below P.

It is also interesting to compare the three definitions of self-reducibility, and the complexity classes to which they correspond, from the point of view of the mathematical structure of the preorder than ensures the well-foundedness of the series of recursive calls. Bounded space classes such as $PSPACE$ correspond to (polynomially) bounded depth preorders, while bounded time classes such as P and E correspond to (polynomially resp. exponentially) bounded size preorders, in the sense indicated in section 4. The most peculiar point is that, for the natural uses of these notions, the resource bounds of the self-reducing machine seem to be much less relevant than the well-foundedness structure. The fact that logarithmic work space suffices to self-reduce $PSPACE$-complete sets is curious, but it can be superseded by the fact that, under appropriate encodings, even $EXPTIME$-complete sets such as those of lemma 10 seem to be wdq-self-reducible via logspace machines. It even could make sense to fix some very strong resource bound to the self-reducing machines, such as logarithmic work space, and impose various reasonable bounds on either the depth, the size, or both, of the preorder corresponding to the well-foundedness of the recursive structure, to check whether in this way self-reducibility structures corresponding to each of the natural complexity classes are found. The author has pursued this line of research for some small steps, finding no really interesting new results, but possibly a greater development would be worth.

From all the parts of theorem 1, two of them remain that do not follow from the results reported here. The original proof uses the round-robin tournament method, that as we have shown is not intrinsically necessary for the other uses it had. The remaining parts correspond to logarithmic advice and classes in the ldq-self-reducibility realm. A forthcoming work [4] will show a theorem stating a principle such as those presented here, and relating ldq-self-reducibility to log advice classes. From it, these two parts of theorem 1 will follow, again among new results, and it will be argued that it subsumes the round-robin tournament method, finally answering in the positive Mahaney's question.

9. Acknowledgements

The author is grateful to Gerd Wechsung for suggesting part of this research, to Jacobo Torán for interesting discussions and for suggesting proposition 6, to Birgit Jenner and Bernd Kirsig for calling his attention to Sudborough's work [20], to Mario Rodríguez

Artalejo for providing him reference [1], and to Josep M. Humet for providing him reference [10].

10. References

[1] A.A. Aho, J.E. Hopcroft, J.D. Ullman: "Time and tape complexity of pushdown automaton languages". *Information and Control* 13 (1968), 186–206.

[2] J.L. Balcázar, R.V. Book, U. Schöning: "The polynomial time hierarchy and sparse oracles". *Journal ACM* 33 (1986), 603–617.

[3] J.L. Balcázar, J. Díaz, J. Gabarró: *Structural Complexity I.* EATCS Monographs, vol. 11, Springer-Verlag (1988).

[4] J.L. Balcázar, U. Schöning. Logarithmic advice classes. Report de recerca LSI-88-12, Univ. Politècnica de Catalunya.

[5] P. Berman: "Relationship between density and deterministic complexity of *NP*-complete languages". In: Int. Coll. Automata, Languages, and Programming, Lect. Notes in Comp. Sci. 62 (1978), 63–71.

[6] A. Borodin, S. Cook, P. Dymond, W. Ruzzo, M. Tompa: "Two applications of complementation via inductive counting". In: Third Structure in Complexity Theory Conference (1988), 116–125.

[7] A. Chandra, D. Kozen, L. Stockmeyer: "Alternation". *Journal ACM* 28 (1981), 114–133.

[8] S. Cook: "Characterizations of pushdown machines in terms of time bounded computers". *Journal ACM* 18 (1971), 4–18.

[9] S. Fortune: "A note on sparse complete sets". *SIAM Journal on Computing* 8 (1979), 431–433.

[10] S. Greibach: "The hardest context-free language". *SIAM Journal on Computing* 2 (1973), 304–310.

[11] M.A. Harrison: *Introduction to formal language theory.* Addison-Wesley (1978).

[12] N. Immerman: "Nondeterministic space is closed under complement". In: Third Structure in Complexity Theory Conference (1988), 112–115.

[13] R. Karp, R. Lipton: "Some connections between nonuniform and uniform complexity classes". In: Proc. 12th ACM Symposium on Theory of Computing (1980), 302–309.

[14] K. Ko: "On self-reducibility and weak *p*-selectivity". *J. Comp. Syst. Sci.* 26 (1983), 209–221.

[15] R. Ladner: "The Circuit Value Problem is logspace complete for P". *SIGACT News*, January 1975, 18–20.

[16] S. Mahaney: "Sparse complete sets for NP: solution of a conjecture by Berman and Hartmanis". *J. Comp. Syst. Sci.* 25 (1982), 130–143.

[17] S. Mahaney: "Sparse sets and reducibilities". In: *Studies in complexity theory*, R. Book ed., Pitman 1986.

[18] A. Meyer, M. Paterson: "With what frequency are apparently intractable problems difficult?". M.I.T. Tech. Report TM-126 (1979).

[19] W. Ruzzo, J. Simon, M. Tompa: "Space-bounded hierarchies and probabilistic computations". *J. Comp. Syst. Sci.* 28 (1984), 216–230.

[20] I.H. Sudborough: "On the tape complexity of deterministic context-free languages". *Journal ACM* 25 (1978), 405–414.

[21] R. Szelepcsényi: "The method of forcing for nondeterministic automata". *Bulletin of the EATCS* 33 (1987), 96–99.