LOGSPACE SELF-REDUCIBILITY

Extended Abstract†

José L. Balcázar
Facultad de Informática de Barcelona, U.P.C.
08028 Barcelona, SPAIN
e-mail: balqui@fib.upc.iris.cernvax (bitnet)

Abstract: A definition of self-reducibility is proposed to deal with logarithmic space complexity classes. A general property derived from the definition is used to prove known results comparing uniform and nonuniform complexity classes below polynomial time, and to obtain new ones regarding nondeterministic nonuniform classes and reducibility to context-free languages.

1. Introduction

The present work is a direct sequel of [2], and reports the results of continuing work on the same subject. Familiarity with this reference is not required but is helpful. In particular, motivations for the study of self-reducibility structures and a discussion of their usefulness in dealing with the relationship between nonuniform and uniform complexity classes are presented there.

In [2], definitions of self-reducibility were proposed to capture, under the form of general results, the power of certain techniques used in [10] to obtain some very interesting consequences for uniform complexity classes from hypothesis about nonuniform complexity classes. The purpose of this work was the study of the role played by the self-reducibility structures in their proofs, since sometimes it is apparent but in some cases it is hidden under a "game" structure corresponding to alternating computations. We propose here additional possibilities for defining self-reducibility structures appropriate to use the same techniques to show properties of complexity classes below P.

Throughout this paper, words are finite sequences of symbols from a fixed, finite alphabet Γ. 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 [15] and the like. The class *poly* contains all functions h from $\mathbb{N}$ into Γ^* such that $|h(n)|$ is bounded by a polynomial in n. 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. For notations and basic results see [4].

We recall now the notation for nonuniform complexity classes defined by Karp and Lipton [10].

1. *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

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

Let us review the main results of the paper by Karp and Lipton:

2. *Theorem* [10].

(A) If NP $\subseteq$ P/poly then the polynomial time hierarchy collapses to $\Sigma_2 \cap \Pi_2$.
(B) If PSPACE $\subseteq$ P/poly then PSPACE $= \Sigma_2 \cap \Pi_2$.
(C) If EXPTIME $\subseteq$ PSPACE/poly then EXPTIME = PSPACE $\neq$ P
(D) If EXPTIME $\subseteq$ P/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.

In some of the proofs, a self-reducibility property was used explicitly. The general property of self-reducible sets which allows one to prove such kind of results was isolated in [3], 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, under the standard definition self-reducible sets are always in PSPACE, while every set

† This work has been partially supported by CIRIT.

in P is trivially self-reducible. Therefore other parts of theorem 1 could not be obtained.

A self-reducibility property able to go up to EXPTIME was defined in [2], and an analogous general theorem about those sets was found which gave rise to parts (C) and (D) as corollaries, together with other results along the same line. Thus the known relations among nonuniform *polynomial* advice classes and uniform classes were shown to appear as consequences of the same principle.

The purpose of this paper is to present a definition of logspace self-reducibility appropriate to work with classes possibly smaller than polynomial time, which is done in section 2, and to show that this definition has also a general property analogous to those of the just mentioned references. We present it in section 3, and obtain as corollaries parts (E) and (F) above. In section 4 we use the same principle to obtain a new result, comparing uniform P with nonuniform NLOG. Here the closure under complements of nondeterministic space classes [9] plays a crucial role. Section 5 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.

1. 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 [14] 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. 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 this purpose.

1. *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 lexicographic criteria. Now shy machines are easily introduced.

2. *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 shoud be made 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 [2].

3. *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.

A more general definition, considering the properties to be imposed to an arbitrary partial order to be used in place of the lexicographic order in the preceding definition, has been developed by the author in order to provide a link with the wdq-self-reducibility studied in [2]. It is similar in spirit to the self-reducibility of [13]. It has not yield new results so far, and therefore will be omitted here.

The following property states the uniqueness of the self-reducible set defined by a given shy machine M. The argument will be useful in the next sections.

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

The proof is by the following inductive argument on each length n: on the smallest word of length n, M cannot query the oracle, therefore it is either both in A and B or outside both of them. Now, for any word w, suppose that A and B coincide on all smaller words. 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).

Observe that the argument can be done separately for each length, using the fact that shy machines always query words of the same length as their input. In fact, we can say that if A is accepted by a shy machine as in proposition 6, and B is 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.

To locate these sets, we can state the following property. We omit the proof.

5. *Proposition.* Every logspace self-reducible set is in P.

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

6. *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.)

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

Proof (sketch). On input $G\#s\#t$, if s is a predecessor of t then accept, otherwise query the oracle about all the words $G\#s\#t'$ where t' is a predecessor of t in G. □

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

8. *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 their two input gates.

It can be seen that CVP is complete for P under logspace reductions [11]. Our requirements about the encoding are irrelevant for the proof. We have the following property.

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

Proof (sketch). 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. The number of queries is always 2. □

The self-reducibility of these sets will be used in the next sections. Other logspace self-reducible sets are presented in section 5.

2. 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 properties of self-reducible sets presented in [2] and [3].

This property is stated as follows:

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

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 it 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 can be shown by applying the inductive argument following proposition 6. □

Now we can derive easily part (E) of theorem 1 as announced. Just apply theorem 12 to the set AGAP, which was shown in the previous section to be logspace self-reducible and NLOG-complete.

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

Similarly, theorem 12 can be applied to CVP, yielding the following.

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

It is very easy to see that if in theorem 12 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.

13. *Corollary.* If P $\subseteq$ DSPACE($\log^k n$)/log then P $\subseteq$ DSPACE($\log^k n$) for every k.

3. 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 12, and requires the use of Immerman's theorem [9] and of some consequences of it. More precisely, we need the following property, which is easy to prove using the results of [9].

14. *Proposition.* DLOG(NLOG) = NLOG.

Now we can state the main result of this section.

15. *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:

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

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. Its only purpose is to present in a clear form the NLOG algorithm to decide the correctness of a given advice. Indeed, we show this claim by proving the following equivalence:

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

The universal quantifier can be tested in log space; the quantified predicate is trivially a DLOG(NLOG), and therefore an NLOG predicate by proposition 16. 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 $\langle sub(x,u), w\rangle \in A$, 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 easily seen to be consistent with the self-reducing machine M. By an inductive argument as the one following proposition 6, 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:

16. *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:

17. *Corollary.* If P $\subseteq$ NSPACE($\log^k n$)/log then P $\subseteq$ NSPACE($\log^k n$) for every k.

4. 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). Its analog class for the deterministic context-free languages is LOG(DCFL). They have been characterized in [16] 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 14 and 18 can be obtained for them.

Our results are based on a smart technique presented in [6], which is based in turn on the decision procedure for context-free languages of [1]. Reference [6] 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.

18. *Lemma.*
    1. 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.
    2. 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.

1. The hardest context-free language of Greibach [7] does not contain the empty word, and is complete under homomorphism for the class of CFL's that do not contain the empty word; therefore it is logspace m-complete for LOG(CFL). By a classical result of automata theory (see [8], theorem 5.5.1), it is accepted by empty store by a pushdown automaton with no λ-transitions.
2. In [16], a deterministic CFL is exhibited that is logspace m-complete for the class of deterministic CFL's (lemma 8 and proof of lemma 9 of [16], see also footnote in page 413). Also, in the same reference, it is shown (lemma 7) that every deterministic CFL is logspace m-reducible to a deterministic CFL 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 AuxPDA$_{pt}$(log) denote the class of sets decidable by nondeterministic logspace auxiliary pushdown automata in polynomial time, and similarly AuxDPDA$_{pt}$(log) for deterministic logspace auxiliary pushdown automata.

19. *Lemma.* The following equalities hold:

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

Proof. It is theorem 1 in [16]. □

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 [6], part (b) implies (c).

20. *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 as strings of length $\log w$ over a large enough alphabet. We assume that this encoding is such that the follow-

ing 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.

21. *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.

22. *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 (sketch). It is not difficult to see by induction that every pair obtained by iteration of the "yield" relation from the identity pairs is realizable.

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. If $t > 1$ and the transition from P_{t-1} to P_t does not change the pushdown, and assuming inductively 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 22.

Proof of theorem 22. The set whose existence is asserted in the statement is 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. 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 21, $A \in$ 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, constructing 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 correct form; its correctness follows from lemma 24.

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:

23. *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 22 to the pushdown automata described in lemma 20 to obtain logspace self-reducible sets complete respectively for LOG(CFL) and LOG(DCFL). Then the results follow from theorems 12 and 17. □

As a final remark, notice that this result does *not* say that if CFL 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 CFL included 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 inclusion of CFL in DLOG/log does not guarantee membership to the nonuniform class of the logspace self-reducible complete set for LOG(CFL).

5. Conclusions

We have developed a notion of logarithmic space self-reducibility, adapting to this resource bound a notion

that is becoming very interesting for comparing uniform and nonuniform complexity classes.

Our definition allows one to obtain known and new consequences in the comparison between uniform and nonuniform classes below polynomial time. In particular, we obtain that if a nonuniform model corresponding to deterministic (resp. nondeterministic) logarithmic space is able of deciding the sets in P then the nonuniformity capability can be "switched off", and the equality P = DLOG (resp. P = NLOG) follows. 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($\log^k n$).

The "/poly" counterparts of some of the nonuniform classes considered here (namely DLOG/poly and NLOG/poly) have been studied quite in depth, and can be characterized by sequences of bounded-size two-way automata and by similarly bounded branching programs —see [4] and [12]—. It would be interesting to find analogous characterizations of the "/log" classes, and also to try to obtain results like those presented here from hypothesis regarding the "/poly" classes.

6. Acknowledgements

The author is grateful to Gerd Wechsung for suggesting this research, to Jacobo Torán for interesting discussions and proofreading, to B. Jenner and B. Kirsig for calling his attention to Sudborough's work [16], to Mario Rodríguez Artalejo for providing him reference [1], and to Josep M. Humet for providing him reference [7].

7. References

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

[2] J.L. Balcázar: "Self-reducibility". In: Proc. Symposium on Theoretical Aspects of Computer Science (1987), Lecture Notes in Computer Science 247, Springer-Verlag, 136–147.

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

[4] J.L. Balcázar, J. Diaz, J. Gabarró: *Structural Complexity I.* EATCS Monographs, vol. 11, Springer-Verlag (1987).

[5] J.L. Balcázar, J. Gabarró: "Nonuniform complexity classes specified by lower and upper bounds". Preprint (1987). To appear in *Inf. Theor. et Applications.*

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

[7] S. Greibach: "The hardest context-free language". *SIAM J. Comput.* 2 (1973), 304–310.

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

[9] N. Immerman: "Nondeterministic space is closed under complement". Preprint (1987). See also these proceedings.

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

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

[12] C. Meinel: "p-projection reducibility and the complexity classes L(nonuniform) and NL(nonuniform)". In: Proc. Mathematical Foundations of Computer Science (1986), Lecture Notes in Computer Science 233, Springer-Verlag, 527–535.

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

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

[15] L. Stockmeyer: "The polynomial-time hierarchy". *Theor. Comp. Sci.* 3 (1977), 1–22.

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