Space-Efficient Recognition of Sparse Self-Reducible Languages

Lane A. Hemachandra, Mitsunori Ogiwara,
and Seinosuke Toda

Technical Report 347
May 1990

UNIVERSITY OF
ROCHESTER
COMPUTER SCIENCE
UNIV. OF ROCHESTER
CARLSON LIBRARY

Space-Efficient Recognition of Sparse Self-Reducible Languages

*Lane A. Hemachandra**

Department of Computer Science
University of Rochester
Rochester, NY 14627

Mitsunori Ogiwara

Department of Information Sciences
Tokyo Institute of Technology
Tokyo 152, Japan

Seinosuke Toda†

Department of Computer Science
University of Electro-communications
Tokyo 182, Japan

May, 1990

Abstract

Mahaney and others have shown that *sparse self-reducible sets have time-efficient algorithms*, and have concluded that it is unlikely that NP has sparse complete sets. Mahaney's work, intuition, and a 1978 conjecture of Hartmanis notwithstanding, until now nothing has been known about the density of complete sets for feasible classes. This paper shows that *sparse self-reducible sets have space-efficient algorithms*, and concludes that it is unlikely that NL, NC^k , LOG(DCFL), LOG(CFL), or P have complete (or even Turing-hard) sets of low density.

1 Introduction

Complete sets are the quintessence of their complexity classes; by studying them, we seek answers to the fundamental complexity-theoretic questions about the classes for which they

*Research supported in part by the National Science Foundation under grant CCR-8996198 and a Presidential Young Investigator Award.

†Research supported in part by the National Science Foundation under grant CCR-8913584.

are complete. Thus, during the last fifteen years, a broad and intense research effort has explored the properties—*isomorphism, equivalence, self-reducibility, etc.*—of the complete sets for most familiar complexity classes. One question on which particularly stunning progress has been made is: Must complete sets be dense?

The progress has been made along two lines. The first line seeks to show that if certain complexity classes have sparse complete sets, then nonintuitive class relationships follow as a consequence. Berman originated this line by showing that if NP has $\leq_m^P$ -hard sets over a single letter alphabet, then $P = NP$ [Ber78].¹ A series of results extended Berman’s claim to the more general notion of sparse sets (see [Mah86, Mah89] for surveys of the complexity-theoretic importance of sparse sets).

Definition 1.1 S is $f(n)$ -sparse if $||S \cap \Sigma^{\leq n}|| = O(f(n))$, where $\Sigma^{\leq n}$ indicates the strings of length at most n .² Throughout this paper, we’ll always assume that $f(\cdot)$ is a non-decreasing logspace-constructible function.

Substantially extending the techniques of Berman and related work of Fortune [For79], in the early 1980s Mahaney proved that if NP has an $n^{O(1)}$ -sparse $\leq_m^P$ -hard sets, then $P = NP$ [Mah82]. This line has continued to be active. Mahaney’s work was built upon in papers by Ukkonen [Ukk83], Yesha [Yes83], and Watanabe [Wat88], culminating in the striking result of Ogiwara and Watanabe [OW90] that if NP has an $n^{O(1)}$ -sparse $\leq_{btt}^P$ -hard set, then $P = NP$.

The research line just described—proving unlikely consequences from the existence of $\leq^P$ -complete sets—is complemented by an alternate research approach. This alternate approach has appeared in a paper by Hartmanis; he proves—absolutely and unconditionally, via a set that diagonalizes against reductions to itself—that many classes (in particular the context-sensitive languages, PSPACE, exponential time, and exponential space) lack $\leq_m^L$ -complete $n^{O(1)}$ -sparse sets [Har78].

Hartmanis follows his results with the conjecture that neither NL nor P has $\leq_m^L$ -complete $n^{O(1)}$ -sparse sets [Har78, p. 286]. It is this conjecture that motivated the present paper. First, we note that Hartmanis coyly neglects to mention a relatively germane point.

Proposition 1.2 If either NL or P lack $\leq_m^L$ -complete $n^{O(1)}$ -sparse sets, then $L \neq P$.

¹Throughout the paper, we’ll use $\leq_r^L$ to denote the various standard types of logspace reductions [LL76] and $\leq_r^P$ to denote the various standard types of polynomial-time reductions [LLS75]. Particular values that r will take on include m (many-one), btt (bounded truth-table), tt (truth-table), and T (Turing).

²In the literature, “sparse” is often used to mean $n^{O(1)}$ -sparse. However, the present paper, like other earlier papers [HIS85, All89], discusses various degrees of sparseness. Thus, we’ll always explicitly specify the sparseness bound.

This follows immediately from the facts that (1) NL and P trivially have $\leq_m^p$ -complete sparse sets, and (2) $L = P$ if and only if the class of functions computable in logarithmic space is identical to the class of functions computable in polynomial time.

Proposition 1.2 should not dim our faith in Hartmanis’s intuition that NL and P lack complete sets, as surely L and P differ. Nonetheless, Proposition 1.2 precludes any reasonable probability of soon proving Hartmanis’s conjectures. In light of Proposition 1.2, the present paper seeks to bolster faith in Hartmanis’s conjectures by tarring the opposition; we show that if NL, NC^k , LOG(DCFL), LOG(CFL), or P has logspace-hard sets of low density, then implausible complexity class inclusions follow.

The principal limitation of our results is that they meaningfully apply only to sets of relatively low density. In the body of the paper we discuss in detail the sparseness levels at which our theorems are of interest.

Our results on NL, NC^k , LOG(DCFL), LOG(CFL), and P follow from a general result that we establish.

Theorem If a set A is logspace self-reducible and A is logspace Turing reducible to some $f(n)$ -sparse set, then $A \in DSPACE[f(p(n)) \log n]$ for some polynomial $p(\cdot)$.

From this, we conclude that P is unlikely to have $\leq_m^L$ -complete (or even $\leq_T^L$ -hard) sets that are less than polynomially-sparse, that NC^k is unlikely to have $\leq_T^L$ -hard sets that are less than $(\log^{k-1} n)$ -sparse, and so on.

Intuitively, these results are of the form: *self-reducibility plus sparseness yields space-efficient algorithms*. This stands in contrast with the earlier work stretching from Berman through Ogiwara and Watanabe, which may be characterized as showing: *self-reducibility plus sparseness yields time-efficient algorithms*. However, those time-efficient algorithms are P algorithms, and our paper deals with sets that are *already* in polynomial time; thus, proving such time efficiency is vacuous, but implications yielding space-efficiency results are indeed surprising.

Our proof technique is quite different from those of all preceding related papers. Hartmanis’s methods [Har78, Section 4] fail critically for NL, P, etc., as his methods require a single NL, P, etc., machine to diagonalize against all logspace reductions. The techniques of Berman [Ber78] and his successors are nearer the mark. Indeed, we proved the earliest version of our main theorem via modifications of such techniques. However, not only were the modifications rather complex, but the results obtained were also weaker; the problems center around the fact that we’re asking a sublinear-space machine to prune a self-reduction tree that is vastly beyond its ability to store.

In contrast, the proof technique we include here—*inductive checking*—lets DSPACE do

what it does best: sequentially investigating possibilities (making essential use of both the sparseness and self-reducibility hypotheses). Both intuitive and formal descriptions of the proof are included in the results section. In addition, the results section makes two related observations. Intuitively, if P has sparse $\leq \frac{L}{f}$ -hard sets, then many instances of a P problem should be individually solvable in small nonuniform space; we verify that this is so. Finally, turning briefly to the one-way logspace reductions introduced by Hartmanis, Immerman, and Mahaney [HIM78] and further studied by Hartmanis and Mahaney [HM81], we note that, from a proof of Hartmanis and Mahaney, L , NL , and P provably have no $2^{\text{polylog-sparse}}$ many-one one-way-logspace-hard sets.

2 Consequences of the Existence of Sparse Self-Reducible Sets

Self-reducibility—the ability to test “ $x \in A$?” via queries to certain strings in A other than x —has played a central role in the development of complexity theory (see [JY90]). The most common notion of self-reducibility—polynomial-time Turing self-reducibility—is superfluous for classes at or below polynomial time. So that he and other researchers might be able to study the self-reducibility properties of low-complexity classes, Balcázar defined logspace self-reducibility.

Definition 2.1 [Bal88] Let x and w be words such that $|w| = \log|x|$.³ We denote by $\text{sub}(x, w)$ the word resulting from substituting the word w for the last $\log|x|$ symbols of x . An *s-restricted 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 $\text{sub}(x, w)$ for some w of length $\log|x|$. A set A is *logspace self-reducible* if there is a logspace s -restricted machine M such that $A = L(M^A)$, and on every input x , every word queried by M is lexicographically smaller than x .

The substantial perspicacity of Balcázar’s intuition that logspace self-reducibility is a powerful tool in studying low-complexity classes is in part certified by the present paper. Our main result is that every logspace self-reducible sparse set of low density is space-efficiently recognizable.

Theorem 2.2 If a set A is logspace self-reducible and is $\leq \frac{L}{f}$ -reducible to an $f(n)$ -sparse set, then $A \in \bigcup_{k>0} \text{DSPACE}[f(n^k) \log n]$.

³Sometimes, for example here, we use $\log n$ as a shorthand for $\lceil \log n \rceil$; this has no effect on the formal claims.

The intuitive flavor of the proof is as follows. Ideally, we'd like our DSPACE algorithm to find the $f(n)$ -sparse set S and check that it agrees with the self-reducibility structure of A , and with the A -hardness of S . However, given only $f(p(n)) \log n$ space, the DSPACE algorithm (in all cases of interest) does not even have enough space to store a single string from the posited $f(n)$ -sparse set. Thus, we propose a scheme that allows us to store compressed versions of the names of certain strings in the $f(n)$ -sparse set—perhaps not all strings of the lengths we are interested in, but certainly all strings needed to check the self-reducibility structure of our set. Our compact representation, though used to check the self-reducibility structure, is itself crucially dependent upon the self-reducibility structure. We'll sequentially test every possible compressed set until we find one that “works.” By “working,” we mean that the compressed set agrees with the self-reduction tree implicitly generated by the current input (obtained by repeatedly applying the self-reduction scheme), and with the mappings from this tree to the $f(n)$ -sparse set. We'll check this by, sequentially, working our way back from $sub(x, 0^{\log |x|})$ towards x , checking the correctness of the self-reduction of, and map to the sparse set of, each string under the assumption that all smaller strings are correctly reduced to the currently posited $f(n)$ -sparse set. Informally, one might describe this procedure as *inductive checking*. We now present a formal proof.

Definition 2.3 (Equivalent to the definition of [LL76]) We say that $A \leq_{tt}^L B$ if there exist two logspace computable functions, g and e , such that for every x :

1. $g(x) = y_1 \# \cdots \# y_m$, ($m \geq 1$), and
2. $x \in A \iff e(x, \chi_B(y_1), \dots, \chi_B(y_m)) = 1$.

We simply say that (g, e) *accepts x relative to B* if $e(x, \chi_B(y_1), \dots, \chi_B(y_m)) = 1$ in 2 above.

Lemma 2.4 [LL76] For any sets A and B , $A \leq_T^L B$ if and only if $A \leq_{tt}^L B$.

Proof of Theorem 2.2 We will show that if A is logspace self-reducible and is $\leq_{tt}^L$ -reducible to an $f(n)$ -sparse set, then $A \in \text{DSPACE}[f(p(n)) \cdot \log_2 n]$ for some polynomial p . The theorem follows immediately from this and Lemma 2.4.

Suppose A is logspace self-reducible and is $\leq_{tt}^L$ -reducible to an $f(n)$ -sparse set S . Let M be an s -restricted machine witnessing the self-reducibility of A . Let (g, e) be a $\leq_{tt}^L$ -reduction from A to S . Then we want to design a deterministic machine that accepts A in $O(f(p(n)) \cdot \log_2 n)$ space for some polynomial p . In order to do this, we define some notions and notations.

Let t be a polynomial bounding the number of query strings generated by g ; that is, for all x , if $g(x) = y_1 \# y_2 \# \cdots \# y_m$, then $m \leq t(|x|)$. For any string x , let $W_x = \{w \mid |w| =$

$\log_2 |x|$. For any positive integer m , let $[m] = \{1, \dots, m\}$. For any $x, w \in W_x$, and $j \in [t(|x|)]$, we define $Q_x(w, j) = \{y\}$ if the j th query string generated in $g(\text{sub}(x, w))$ is y , and $Q_x(w, j) = \emptyset$ otherwise. We can easily extend this notion to any set $C \subseteq W_x \times [t(|x|)]$: $Q_x(C) = \bigcup_{(w,j) \in C} Q_x(w, j)$. We observe the following fact.

Fact 1. For every string x , there is a set $C \subseteq W_x \times [t(|x|)]$ such that $Q_x(C) \subseteq S$ and for every w in W_x , (g, e) accepts $\text{sub}(x, w)$ relative to $Q_x(C)$ iff (g, e) accepts $\text{sub}(x, w)$ relative to S iff $\text{sub}(x, w) \in A$.

Proof. We can find a set $C \subseteq W_x \times [t(|x|)]$ so that $Q_x(C) = Q_x(W_x \times [t(|x|)]) \cap S$. Clearly, this set C satisfies the conditions above.

End of Fact 1

Below, we say that a set $C \subseteq W_x \times [t(|x|)]$ is *consistent with S for x* if for every w in W_x , (g, e) accepts $\text{sub}(x, w)$ relative to $Q_x(C)$ iff (g, e) accepts $\text{sub}(x, w)$ relative to S .

Let p be a polynomial such that for every string x and for every w in W_x , the maximum length of query strings generated by $g(\text{sub}(x, w))$ is bounded above by $p(|x|)$. Then it is clear that the number of strings in $Q_x(W_x \times [t(|x|)]) \cap S$ is bounded above by $f(p(|x|))$. Hence, we have the following fact.

Fact 2. For some $C \subseteq W_x \times [t(|x|)]$ that is consistent with S for x , $\|C\| \leq f(p(|x|))$.

The machine that we want to describe is based on Fact 1 and Fact 2, and intuitively operates in the following manner. Given an input x , the machine enumerates the sets $C \subseteq W_x \times [t(|x|)]$ that have at most $f(p(|x|))$ elements. For each C , it checks whether C is consistent with S for x . This check is done inductively from the lexicographically least string in W_x toward the largest string in W_x , by simulating (g, e) relative to $Q_x(C)$ and simulating M . The check is easy for the least string in W_x . After succeeding at the check for the i th string, the machine proceeds to the check for the $(i+1)$ st string. In this check, the machine simulates (g, e) once again relative to $Q_x(C)$ on the strings less than the $(i+1)$ st string if those strings are queried by M on the $(i+1)$ st string.

Now we describe the machine N working on an input x , based on the above intuition.

(The machine N)

Let $w_1, w_2, \dots, w_m$ be the lexicographical list of strings in W_x .

For each $C \subseteq W_x \times [t(|x|)]$ such that $\|C\| \leq f(p(|x|))$, until it finds a C for which (1) is satisfied, N checks (1).

- (1) For each $w \in \{w_1, w_2, \dots, w_m\}$, in turn, starting with $w = w_1$ and proceeding towards $w = w_m$, N simulates (g, e) on $sub(x, w)$ relative to $Q_x(C)$ and checks whether $sub(x, w) \in A$ iff (g, e) accepts $sub(x, w)$ relative to $Q_x(C)$, in the following manner:
 - (a) If $w = w_1$, then N simulates M on input $sub(x, w)$ and simulates (g, e) on $sub(x, w)$ relative to $Q_x(C)$. If both simulations have the same outcome, then N proceeds to the check for the next string w_2 ; otherwise, it proceeds to the next set C .
 - (b) If $w \neq w_1$, then N simulates (g, e) on input $sub(x, w)$ relative to $Q_x(C)$ and gets the outcome—namely, whether $Q_x(C)$ asserts (possibly incorrectly) that $sub(x, w) \in A$. After that, N begins to simulate M on input $sub(x, w)$. Each time that M queries some string $sub(x, v)$, N simulates (g, e) on $sub(x, v)$ relative to $Q_x(C)$, and N continues the simulation of M from the “yes” configuration or the “no” configuration according to the outcome of (g, e) on $sub(x, v)$ relative to $Q_x(C)$. When M completes, if its outcome is the same as (g, e) ’s outcome above, then N goes to the check for the next string in W_x (i.e., continues our “for $w \in \dots$ ” loop); otherwise, it proceeds to the next set C .

After succeeding at the above process for some set C , N simulates (g, e) on input x relative to $Q_x(C)$. If (g, e) accepts x relative to $Q_x(C)$, then N accepts x ; otherwise, it rejects x . **(End of machine specification)**

It is easy to see that all elements in $W_x \times [t(|x|)]$ can be encoded into binary strings of length $O(\log |x|)$. Hence N uses at most $O(f(p(|x|)) \cdot \log |x|)$ work space in order to store the sets C . To see that all computations can be done in space logarithmic in $|x|$ (except for storing the sets C), the most crucial point is how to simulate (g, e) on an input $sub(x, u)$ relative to $Q_x(C)$ in the above machine N . This can be done as follows:

(The simulation of (g, e) on $sub(x, u)$ relative to $Q_x(C)$)

Each time that the machine N has to simulate (g, e) on $sub(x, u)$ relative to $Q_x(C)$, N begins to simulate e . When e needs to read the oracle answer for the i th query string generated by $g(sub(x, u))$, N does the following:

- (1) N computes $Q_x(w, j)$, one after another, for each $(w, j) \in C$.
- (2) N simultaneously computes the i th query string, say z , generated by $g(sub(x, u))$.
- (3) N compares each $Q_x(w, j)$ with z bit by bit in the computations (1) and (2) above.

- (4) If $Q_x(w, j) = z$ for some $(w, j) \in C$, then N knows the oracle answer is “yes”; otherwise, it knows the oracle answer is “no.”

N simulates e in the above manner until e halts.

(End of the simulation)

It is not difficult to see that the above simulation uses only logarithmic space except for storing the set C .

It remains to show that the machine works correctly. Let x be any input to N . From Fact 1, there is a set $C \subseteq W_x \times [t(|x|)]$ that is consistent with S for x ; that is, for every w in W_x , (g, e) accepts $sub(x, w)$ relative to $Q_x(C)$ iff $sub(x, w) \in A$. For this set C , we can observe that N successfully completes the process (1). This observation is done by induction on $w_i, i = 1, \dots, m$. For w_1 , it is clear that N correctly finds in (1a) that M on $sub(x, w_1)$ enters an accepting state iff $sub(x, w_1)$ is in A iff (g, e) accepts $sub(x, w_1)$ relative to $Q_x(C)$. Thus N proceeds to the check for the next string w_2 . Assume that for some $i > 1$, N finds that for each $w_j, 1 \leq j < i$, (g, e) accepts $sub(x, w_j)$ relative to $Q_x(C)$ iff M on $sub(x, w_j)$ enters an accepting state iff $sub(x, w_j) \in A$. Then we inductively see that the simulation of M on $sub(x, w_i)$ gives us the same outcome as M^S on $sub(x, w_i)$; hence, M enters an accepting state in that simulation iff $sub(x, w_i) \in A$. From the consistency of C with S for x , $sub(x, w_i) \in A$ iff (g, e) accepts x relative to $Q_x(C)$. Hence N correctly finds in (1) that M on $sub(x, w_i)$ enters an accepting state iff (g, e) accepts $sub(x, w_i)$ relative to $Q_x(C)$, and it goes to the check for the string w_{i+1} . Thus, N successfully completes the computations in (1). From the consistency of C with S for x , it follows that N accepts x iff $x \in A$. ■

Theorem 2.2 yields corollaries about classes of core interest in complexity theory. Following standard conventions, we use NC^k to denote the class of sets recognized by logspace-uniform bounded fan-in circuits that are polynomial size and $\mathcal{O}(\log^k n)$ depth-bounded, simultaneously [Pip79]. $LOG(DCFL)$ and $LOG(CFL)$ are the classes of languages logspace many-one reducible to, respectively, deterministic context-free languages and context-free languages [Sud78]. All these classes lack low-density $\leq \frac{L}{T}$ -hard sets unless implausible class inclusions hold.

Lemma 2.5 [Bal88]

1. There is a $\leq \frac{L}{m}$ -complete set for P that is logspace self-reducible. There is a $\leq \frac{L}{m}$ -complete set for NL that is logspace self-reducible.
2. There is a $\leq \frac{L}{T}$ -complete set for $LOG(DCFL)$ that is logspace self-reducible. There is a $\leq \frac{L}{T}$ -complete set for $LOG(CFL)$ that is logspace self-reducible.

We'd like a similar lemma for NC^k . Below, in the encoding of each circuit α , we assume that the label of each gate is lexicographically greater than the labels of its two inputs. Furthermore, for an $s(n)$ -size-bounded circuit family $\alpha = \{\alpha_1, \alpha_2, \dots, \alpha_n, \dots\}$, we assume that the label of each gate in α_n is encoded into a binary number of length $\lceil \log s(n) \rceil$. Without loss of generality, we may make these assumptions. For the details of the former assumption, the reader may refer to Theorem 12 of [BCD⁺89] and Theorem 3.2 of [Wil90].

Lemma 2.6 Let $k \geq 2$. For any set $A \in \text{NC}^k$, there is a logspace self-reducible set B such that A is $\leq_m^L$ -reducible to B .

Proof. Let $\alpha = \{\alpha_1, \alpha_2, \dots, \alpha_n, \dots\}$ be a family of logspace uniform circuits that witnesses a set A being in NC^k . Then, for some polynomial p , the size of each α_n is bounded by $p(n)$. We define a set B as follows: B consists of strings of the form $\langle x, g \rangle$, and $\langle x, g \rangle$ is in B if and only if:

- (1) g is the label of a gate in $\alpha_{|x|}$,
- (2) if g is an INPUT gate, then its corresponding input bit of x is 1,
- (3) if g is an OR gate, then $\langle x, g_1 \rangle \in B$ or $\langle x, g_2 \rangle \in B$,
- (4) if g is an AND gate, then $\langle x, g_1 \rangle \in B$ and $\langle x, g_2 \rangle \in B$, and
- (5) if g is a NOT gate, then $\langle x, g_1 \rangle \notin B$,

where g_1 and g_2 are the gates that provide input to gate g . Clearly, B is logspace self-reducible by its definition. It is also clear that $A \leq_m^L B$. Below, we show $B \in \text{NC}^k$. For each gate g in α_n , let $\alpha_n^g(x)$ denote the output of the gate g in α_n on input x . Let us define two boolean functions EQ and SELECT as follows:

$$\text{EQ}(u, v) = 1 \iff u = v.$$

$$\text{SELECT}(a_1 a_2 \dots a_m, b_1 b_2 \dots b_m) = \bigvee_{i=1}^m (a_i \wedge b_i).$$

Then we can describe a circuit $\beta_{n+\lceil \log p(n) \rceil}$ that recognizes $B^{=n+\lceil \log p(n) \rceil}$: For input $\langle x, g \rangle$ where $|g| = \lceil \log p(|x|) \rceil$,

- (1) $a_i \leftarrow \alpha_n^{g_i}(x)$ and $b_i \leftarrow \text{EQ}(g, g_i)$ for $1 \leq i \leq m$, where g_i is the label of the i th gate in α_n and m is the number of gates in α_n .
- (2) Output $\text{SELECT}(a_1 a_2 \dots a_m, b_1 b_2 \dots b_m)$.

It is easy to see that $\langle x, g \rangle \in B$ iff the output of the gates g in $\alpha_{|x|}(x)$ is 1 iff $\beta_{|x|+\lceil \log p(|x|) \rceil}(x, g)$ outputs 1; furthermore, it is easily seen that $\beta_{n+\lceil \log p(n) \rceil}$ is polynomial size in n and $\mathcal{O}(\log^k n)$ depth-bounded, and it is also easy to see that $\{\beta_m\}_{m \geq 1}$ is logspace uniform. ■

From Theorem 2.2, Lemmas 2.5 and 2.6, and the fact that $\leq_T^L$ is transitive [LL76,Lad75], we immediately obtain the following result.

Corollary 2.7 For a class $\mathcal{C}$ chosen from $\{\text{NL}, \text{LOG}(\text{DCFL}), \text{LOG}(\text{CFL}), \text{P}, \text{NC}^2, \text{NC}^3, \text{NC}^4, \dots\}$, if $\mathcal{C}$ has an $f(n)$ -sparse $\leq_T^L$ -hard set, then $\mathcal{C} \subseteq \bigcup_{k \geq 0} \text{DSPACE}[f(n^k) \log n]$.

For which sparseness bounds $f(n)$ is the above corollary meaningful? We discuss each class in turn, and also provide some examples.

NL The best known DSPACE containment of NL is that of Savitch's Theorem: $\text{NL} \subseteq \text{DSPACE}[\log^2 n]$ [Sav70]. Thus, Corollary 2.7 is vacuous for $f(n)$ at least $\Omega(\log n)$. For smaller $f(n)$, Corollary 2.7 implies surprising relationships that would improve Savitch's Theorem. For example:

Corollary 2.8 If NL has a $\sqrt{\log n}$ -sparse $\leq_T^L$ -hard set, then $\text{NL} \subseteq \text{DSPACE}[\log^{3/2} n]$.

LOG(DCFL) and LOG(CFL) The best known space containment for both CFLs is $\text{DSPACE}[\log^2 n]$, and this is also the best known containment for DCFLs [LSH65, BCMV83]. It follows that LOG(DCFL) and LOG(CFL) sets can be recognized in $\text{DSPACE}[\log^2 n]$. Thus, the corollary is vacuous for $f(n)$ at least $\Omega(\log n)$. For smaller $f(n)$, Corollary 2.7 implies surprising relationships that would improve the results of [LSH65, BCMV83].

NC^k For $k \geq 2$, $\text{NC}^k \subseteq \text{DSPACE}[\log^k n]$ [Bor77]. Thus, for $k \geq 2$, Corollary 2.7 is vacuous for $f(n)$ at least $\Omega(\log^{k-1} n)$. For smaller $f(n)$, Corollary 2.7 implies surprising relationships that would improve Borodin's results. For example:

Corollary 2.9 If, for some $k' \geq 2$, $\text{NC}^{k'}$ has a $(\log^{k'-2} n)$ -sparse $\leq_T^L$ -hard set, then $\text{NC}^{k'} \subseteq \text{DSPACE}[\log^{k'-1} n]$.

P Corollary 2.7 is vacuous for $f(n)$ at least $n^{1/j}$, some j . For smaller $f(n)$, Corollary 2.7 implies surprising relationships. For example:

Corollary 2.10 If P has a $\log n$ -sparse $\leq_T^L$ -hard set, then $\text{P} \subseteq \text{DSPACE}[\log^2 n]$.

Theorem 2.2 and its corollaries don't apply to the case of $n^{\mathcal{O}(1)}$ -sparseness. We make two observations that do apply to this case. First, Theorem 2.12 verifies the fact that if a language has a small $\leq \frac{L}{T}$ -hard sparse set, then the language's instances can be solved with a small amount of information (namely, the sparse set) in the Karp-Lipton advice model [KL80]. The second observation, due to Hartmanis and Mahaney, notes that for one-way reductions, we indeed lack complete sparse sets.

The first result follows immediately from a lemma that is based on the equivalence between $P/poly$ and $\{L \mid L \leq_T^p S, \text{ for some sparse set } S\}$ (attributed to A. Meyer in [BH77]). Analogously, we note that it holds that $A \in L/poly$ iff $A \in \{L \mid L \leq_T^L S, \text{ for some sparse set } S\}$. For the purposes of this paper, we prove a variation on this.

Lemma 2.11 If $A \in \{L \mid L \leq_T^L S, \text{ for some } f(n)\text{-sparse set } S\}$, then $A \in L/f(poly)$.

Proof Let M be a logspace-bounded deterministic oracle machine accepting a set A via an $f(n)$ -sparse oracle set S . Let p be a polynomial bounding the length of query strings made by M . Then, for all inputs x of length n , there are at most $f(p(n))$ strings in S possibly queried by M . We define a set B by

$$\begin{aligned} B = \{ & x \# w_1 \# w_2 \# \cdots \# w_m \mid m \leq f(p(n)), \\ & |w_i| \leq p(|x|) \text{ for all } i, \text{ and } M \text{ accepts } x \\ & \text{relative to } \{w_1, \dots, w_m\} \}. \end{aligned}$$

Clearly, B is in L . For every integer $n \geq 0$, we define $g(n)$ to be the lexicographically ordered list of strings in S up to length $p(n)$. Then it is easy to see that for every x , $x \in A$ if and only if $x \# g(|x|) \in B$. Thus A is in $L/f(poly)$. $\blacksquare$

Theorem 2.12

1. If P has an $f(n)$ -sparse $\leq \frac{L}{T}$ -hard set, then $P \subseteq L/f(poly)$.
2. If NL has an $f(n)$ -sparse $\leq \frac{L}{T}$ -hard set, then $NL \subseteq L/f(poly)$.

Theorem 2.13 (Essentially [HM81]) P has no 2^{polylog} -sparse $\leq_m^{1-L}$ -hard sets.

3 Conclusions and Open Problems

We proved that if a set A is logspace self-reducible and logspace Turing reducible to some $f(n)$ -sparse set, then A is in $DSPACE[f(p(n)) \log n]$ for some polynomial $p(\cdot)$. From this, we concluded that if NL , NC^k , $LOG(DCFL)$, $LOG(CFL)$, or P have sufficiently sparse logspace-hard sets, then implausible class inclusions follow that would imply the suboptimality of

many fundamental results. The major open question that remains is: Do our results extend to the case of polynomially-sparse sets?

Acknowledgements

We are very grateful to Ron Book for making our collaboration possible, and to Juris Hartmanis and Joel Seiferas for helpful suggestions on presentation.

References

- [All89] E. Allender. Limitations of the upward separation technique. In *Automata, Languages, and Programming (ICALP 1989)*, pages 18–30. Springer-Verlag *Lecture Notes in Computer Science* #372, July 1989.
- [Bal88] J. Balcázar. Logspace self-reducibility. In *Proceedings 3rd Structure in Complexity Theory Conference*, pages 40–46. IEEE Computer Society Press, June 1988.
- [BCD⁺89] A. Borodin, S. Cook, P. Dymond, W. Ruzzo, and M. Tompa. Two applications of inductive counting for complementation problems. *SIAM Journal on Computing*, 18(3):559–578, 1989.
- [BCM⁺83] B. von Braunmühl, S. Cook, K. Mehlhorn, and R. Verbeek. The recognition of deterministic CFLs in small time and space. *Information and Control*, 56(1):34–51, 1983.
- [Ber78] P. Berman. Relationship between density and deterministic complexity of NP-complete languages. In *Automata, Languages, and Programming (ICALP 1978)*, pages 63–71. Springer-Verlag *Lecture Notes in Computer Science* #62, 1978.
- [BH77] L. Berman and J. Hartmanis. On isomorphisms and density of NP and other complete sets. *SIAM Journal on Computing*, 6(2):305–322, 1977.
- [Bor77] A. Borodin. On relating time and space to size and depth. *SIAM Journal on Computing*, 6(4):733–744, 1977.
- [For79] S. Fortune. A note on sparse complete sets. *SIAM Journal on Computing*, 8(3):431–433, 1979.
- [Har78] J. Hartmanis. On log-tape isomorphisms of complete sets. *Theoretical Computer Science*, 7(3):273–286, 1978.
- [HIM78] J. Hartmanis, N. Immerman, and S. Mahaney. One-way log-tape reductions. In *Proceedings 19th IEEE Symposium on Foundations of Computer Science*, pages 65–71, 1978.

- [HIS85] J. Hartmanis, N. Immerman, and V. Sewelson. Sparse sets in NP-P: EXPTIME versus NEXPTIME. *Information and Control*, 65(2/3):159–181, May/June 1985.
- [HM81] J. Hartmanis and S. Mahaney. Languages simultaneously complete for one-way and two-way log-tape automata. *SIAM Journal on Computing*, 10(2):383–390, 1981.
- [HU79] J. Hopcroft and J. Ullman. *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley, 1979.
- [JY90] D. Joseph and P. Young. Self-reducibility: Effects of internal structure on computational complexity. In A. Selman, editor, *Complexity Theory Retrospective*. Springer-Verlag *Lecture Notes in Computer Science*, 1990. To appear.
- [KL80] R. Karp and R. Lipton. Some connections between nonuniform and uniform complexity classes. In *12th ACM Sym. on Theory of Computing*, pages 302–309, 1980.
- [Lad75] R. Ladner. On the structure of polynomial time reducibility. *Journal of the ACM*, 22(1):155–171, 1975.
- [LL76] R. Ladner and N. Lynch. Relativization of questions about log space computability. *Mathematical Systems Theory*, 10(1):19–32, 1976.
- [LLS75] R. Ladner, N. Lynch, and A. Selman. A comparison of polynomial time reducibilities. *Theoretical Computer Science*, 1(2):103–124, 1975.
- [LSH65] P. Lewis, R. Stearns, and J. Hartmanis. Memory bounds for recognition of context-free and context-sensitive languages. In *Proceedings of 6th IEEE Symposium on Switching Circuit Theory and Logical Design*, pages 191–202, 1965.
- [Mah82] S. Mahaney. Sparse complete sets for NP: Solution of a conjecture of Berman and Hartmanis. *Journal of Computer and System Sciences*, 25(2):130–143, 1982.
- [Mah86] S. Mahaney. Sparse sets and reducibilities. In R. Book, editor, *Studies in Complexity Theory*, pages 63–118. John Wiley and Sons, 1986.
- [Mah89] S. Mahaney. The isomorphism conjecture and sparse sets. In J. Hartmanis, editor, *Computational Complexity Theory*, pages 18–46. American Mathematical Society, 1989. Proceedings of Symposia in Applied Mathematics #38.
- [OW90] M. Ogiwara and O. Watanabe. On polynomial bounded truth-table reducibility of NP sets to sparse sets. In *22nd ACM Symposium on Theory of Computing*, pages 457–467. ACM Press, May 1990.
- [Pip79] N. Pippenger. On simultaneous resource bounds. In *Proceedings 20th IEEE Symposium on Foundations of Computer Science*, pages 307–311, 1979.
- [Sav70] W. Savitch. Relationships between nondeterministic and deterministic tape complexities. *Journal of Computer and System Sciences*, 4(2):177–192, 1970.

- [Sud78] I. Sudborough. On the tape complexity of deterministic context-free languages. *Journal of the ACM*, 25(3):405–414, 1978.
- [Ukk83] E. Ukkonen. Two results on polynomial time truth-table reductions to sparse sets. *SIAM Journal on Computing*, 12(3):580–587, 1983.
- [Wat88] O. Watanabe. On $\leq^p_{1-tt}$ sparseness and nondeterministic complexity classes. In *Automata, Languages, and Programming (ICALP 1988)*, pages 697–709. Springer-Verlag *Lecture Notes in Computer Science* #317, 1988.
- [Wil90] C. Wilson. On the decomposability of NC and AC. *SIAM Journal on Computing*, 19(2):384–396, 1990.
- [Yes83] Y. Yesha. On certain polynomial-time truth-table reducibilities of complete sets to sparse sets. *SIAM Journal on Computing*, 12(3):411–425, 1983.