Some Observations about Relativization of Space Bounded Computations

Juris Hartmanis* Richard Chang† Jim Kadin‡
Stephen G. Mitchell

Department of Computer Science, Cornell University
Ithaca, New York 14853, USA

1 About Relativization

In this column we explore what relativization says about space bounded computations and what recent results about space bounded computations say about relativization.

There is a strong belief in computational complexity circles that problems which can be relativized in two contradictory ways are very hard to solve. We believe that such problems can only be solved by proof techniques that do not relativize. For example, standard diagonalization methods are powerless in these cases [BGS75, HH76, Hop84]. So far only very simple problems with contradictory relativizations have been solved [Har85, Cha90]. Unfortunately, many important problems in complexity theory have contradictory relativizations and have to be viewed as inaccessible to our current proof techniques. The classic example is the $\mathrm{P} \stackrel{?}{=} \mathrm{NP}$ problem for which Baker, Gill and Solovay [BGS75] exhibited recursive oracles A and B such that

$$\mathrm{P}^A = \mathrm{NP}^A \quad \text{and} \quad \mathrm{P}^B \neq \mathrm{NP}^B.$$

Since then, a large number of other computational complexity problems have been shown to have contradictory relativizations and so far none of them have been solved.[1]

Today, a proof that a problem has contradictory relativizations is viewed as strong evidence that the problem cannot be solved by current proof techniques and such problems are generally not attacked vigorously. In an intuitive sense, a contradictory

*This research was supported by NSF Research Grant DCR 85-20597.

†Current Address: Department of Computer Science, University of Maryland, Baltimore County Campus, Baltimore, MD 21228, USA.

‡Current Address: Department of Computer Science, University of Maine, Neville Hall, Orono, ME 04469, USA.

[1]Since the writing of this column, there has been a breakthrough in the study of interactive protocols which resulted in a non-relativizing proof technique [LFKN90, Sha90]

relativization of a problem can be viewed as a weak form of an *independence* result asserting that the problem is "not provable in ..." or "not provable with ...". So far, these concepts have not been made precise. At the same time, we believe that it is important to clarify and make precise what kind of "independence" results are implied by what kind of contradictory relativizations. We hope that this challenge will be taken up and that the "meaning of contradictory relativizations" in complexity theory will be clarified.

The theoretical computer science community has shown a love-hate attitude toward relativization results in complexity theory. Several relativization results have caused great excitement while other sound work in relativization have not been kindly received by program committees. This is partially understandable since quite a few of the relativization results looked like pure recursive function theory without direct relevance to computer science. At the same time, many beautiful relativization results have enriched theoretical computer science, and we believe that a deeper understanding of relativization will yield further important insights into the nature of computing.

In our opinion, relativization work in computational complexity plays a different role than in recursive function theory. In complexity theory, relativization results should be viewed as revealing logical possibilities about computational structures, as a study of the power of different access mechanisms to information, and as a study of how the structure of the accessed information affects this power. Finally, a relativization result can be viewed as a not-yet-fully-understood "independence" result which reveals the difficulty of a problem.

A nice example, which reveals the power of different access mechanisms and the influence of the structure of information, is the result showing that polynomially many parallel queries to SAT can be replaced by $O(\log n)$ many sequential queries [Hem87]:

$$\mathrm{P}^{\mathrm{SAT}[\log n]} = \mathrm{P}^{\mathrm{SAT}||}.$$

Clearly, this result depends on the structure of SAT, and one can easily show that this result fails to hold with probability 1 for random oracles [Kad88].

A surprising version of this result for a constant number of queries has been shown by Beigel [Bei87]. For all $k \geq 1$,

$$\mathrm{P}^{\mathrm{SAT}[k]} = \mathrm{P}^{\mathrm{SAT}||[2^k-1]}$$

Some other interesting results about various access mechanisms to SAT have been observed in [Wag88]. Let $\mathrm{LOG}^{\mathrm{SAT}}$ denote the set of languages accepted by a deterministic $\log n$ tape bounded machine with a one-way oracle tape. Then

$$\mathrm{P}^{\mathrm{SAT}[\log n]} = \mathrm{LOG}^{\mathrm{SAT}} = \mathrm{LOG}^{\mathrm{SAT}[\log n]} = \mathrm{LOG}^{\mathrm{SAT}||}.$$

2 Relativization of Space Bounded Computations

The Immerman-Szelepcsényi result [Imm88, Sze88], which showed that nondeterministic space bounded computations are closed under complementation, came as an unexpected surprise. The elegance and simplicity of this proof is so nice that in the *33rd EATCS Bulletin* the proof was presented three times: once in the original paper by Szelepcsényi, again in the "Formal Language Theory Column" by Salomaa and a third time in the "Structural Complexity Column" by Hartmanis. It seems that the possibility of contradictory relativizations of this problem was not explored, which would have been impossible to obtain in the correct relativization model. This impossibility would have suggested that the problem could be solved by "known techniques". As it turned out, with some ingenuity, it can.

In view of this situation, we review and derive some new results about relativization of space bounded computations. Our main interest is in what relativization says about space bounded computations and vice versa.

Since 1965, when the study of space bounded computations was initiated [SHL65, LSH65], it was clear that the reusability of space yields sharper hierarchy results than those obtainable for time bounded computations [HS65]. Also, Savitch's result [Sav70] showed that nondeterminism for space bounded computations could at best save the square root in resources over deterministic computations. That is, for space constructible bounds $L(n) \geq \log n$:

$$\mathrm{NSPACE}[L(n)] \subseteq \mathrm{SPACE}[L(n)^2].$$

Therefore,

$$\mathrm{PSPACE} = \mathrm{NPSPACE},$$

in contrast to our belief that

$$\mathrm{P} \neq \mathrm{NP}.$$

On the other hand, the relativization of space bounded computations presented a more complex problem than the time bounded case. For space bounded computations, there are several options for the access mechanism to the oracle [LL76, RST84]. For deterministic and nondeterministic $(\log n)$-space bounded computations, we can allow the machines to use an additional one-way write-only oracle tape on which they can write n^k-long queries. We denote the corresponding language classes by LOG^A and NLOG^A respectively.

It is easily shown that

$$\mathrm{LOG} \subseteq \mathrm{NLOG} \subseteq \mathrm{P}.$$

However, using a straightforward Baker-Gill-Solovay oracle construction [BGS75, LL76] one can construct an oracle A such that

$$\mathrm{NLOG}^A \not\subseteq \mathrm{P}^A.$$

Intuitively, NLOG^A can query nondeterministically all strings of length n in A, whereas P^A can query only polynomially many such strings. This permits an A where

$$\{\ 1^n \mid (\exists x)[\ x \in A \text{ and } |x| = n]\ \} \in \mathrm{NLOG}^A - \mathrm{P}^A.$$

Similarly, we can construct oracles B and C which, for this model of relativization, invalidate Savitch's result [LL76],

$$\mathrm{NLOG}^B \not\subseteq \mathrm{SPACE}^B[(\log n)^2],$$

as well as the Immerman-Szelepcsényi result,

$$\mathrm{NLOG}^C \neq \overline{\mathrm{NLOG}^C}.$$

The construction of C is achieved by judiciously placing at most one string of each length in C so that

$$\Delta = \{\ 1^n \mid (\exists x)[\ x \in C \text{ and } |x| = n]\ \} \in \mathrm{NLOG}^C - \overline{\mathrm{NLOG}^C}.$$

The key observation in the construction of C is that if an NLOG^C machine accepts an input 1^n with $C \cap \Sigma^n = \emptyset$, then there is an accepting computation path which makes only polynomially many queries to C. Therefore, there are strings x, $|x| = n$, which were not queried on this accepting path. We add one such string to C to insure that this machine does not accept $\overline{\Delta}$. If this machine does not accept 1^n on any path with $C \cap \Sigma^n = \emptyset$, then it does not accept $\overline{\Delta}$, and we add no strings of length n to C.

All these violations, in particular the $\mathrm{NLOG}^A \not\subseteq \mathrm{P}^A$ violation, suggest that this is not the right model for relativizing space bounded computation.

To avoid these anomalies, another somewhat artificial oracle access mechanism was defined in [RST84]. In this model, there is again an additional one-way write-only oracle tape, but the oracle query must be written deterministically — even for non-deterministic machines. (That is, during the query computation phase the machine computes deterministically). We denote the corresponding language classes by $\mathrm{LOG}^{D(A)}$ and $\mathrm{NLOG}^{D(A)}$. Now, for any given input, both types of machines can query only polynomially many strings in A, and the above anomalies are avoided. On the other hand, when we consider linear tape bounds, this model looks artificial because it permits queries that are exponentially long in the input length.

Finally, the simplest and most natural oracle access mechanism is the one which limits the oracle queries to the length of the work tape. We denote the corresponding language classes by $\mathrm{LOG}^{S(A)}$ and $\mathrm{NLOG}^{S(A)}$. In this model the nondeterministic machines can make nondeterministically computed queries.

The one objection to this model is that there exist oracles A such that

$$A \notin \mathrm{NLOG}^{S(A)},$$

since only $\log n$ long queries are possible on inputs of length n. This anomaly disappears for linear and larger tape bounds.

In the following we will show that, quite surprisingly, the $D(\cdot)$ and $S(\cdot)$ models of relativization behave very similarly when we deal with the possibility of separating deterministic and nondeterministic space bounded classes.

3 Relativized Separation of Space Bounded Computations

In [Wil85] it was shown that

$$\mathrm{LOG} = \mathrm{NLOG} \iff (\forall A)[\ \mathrm{LOG}^{D(A)} = \mathrm{NLOG}^{D(A)}\ ].$$

This is a generalization of a result in [Sim77] and in [RS81]. A later publication of this result can also be found in [KL87]. See also [Wil86].

Our extensions of this result are summarized in the following two theorems. The key ideas for the proofs, which are quite simple, will be given in the next section.

Theorem 1 Consider only space constructible $L(n) \geq \log n$. Then,

$$\begin{aligned} \mathrm{LOG} = \mathrm{NLOG} \quad &\iff \quad (\forall L(n), A)[\ \mathrm{SPACE}^{S(A)}[L(n)] = \mathrm{NSPACE}^{S(A)}[L(n)]\ ] \\ &\iff \quad (\forall L(n), A)[\ \mathrm{SPACE}^{D(A)}[L(n)] = \mathrm{NSPACE}^{D(A)}[L(n)]\ ]. \end{aligned}$$

Since the $D(\cdot)$-relativization models for LOG and NLOG can compute n^k-long queries on inputs of length n, there exist oracles A such that

$$\mathrm{LOG}^{D(A)} = \mathrm{NLOG}^{D(A)},$$

(regardless of whether LOG = NLOG or LOG $\neq$ NLOG). We believe that this is not the case for the $S(\cdot)$ relativization when only $\log n$ long queries can be used. We expect that

$$\mathrm{LOG} \neq \mathrm{NLOG} \iff (\forall A)[\ \mathrm{LOG}^{S(A)} \neq \mathrm{NLOG}^{S(A)}\ ],$$

but have not been able to prove it. It seems that proving for some A

$$\mathrm{LOG}^{S(A)} \neq \mathrm{NLOG}^{S(A)}$$

is as hard as proving LOG $\neq$ NLOG.

If LOG $\neq$ NLOG, then for both relativization models we can construct "natural" oracles which collapse and separate the higher complexity classes.

Theorem 2 If LOG $\neq$ NLOG then for all space constructible $L(n) \geq n$ there exist oracles A and B such that

$$\begin{aligned} \mathrm{SPACE}^{D(A)}[L(n)] &= \mathrm{NSPACE}^{D(A)}[L(n)], \\ \mathrm{SPACE}^{S(A)}[L(n)] &= \mathrm{NSPACE}^{S(A)}[L(n)], \\ \mathrm{SPACE}^{D(B)}[L(n)] &\neq \mathrm{NSPACE}^{D(B)}[L(n)], \\ \mathrm{SPACE}^{S(B)}[L(n)] &\neq \mathrm{NSPACE}^{S(B)}[L(n)]. \end{aligned}$$

These results show very clearly that if LOG = NLOG then we cannot obtain contradictory relativizations of these classes and of higher deterministic and nondeterministic space bounded classes. In other words, the existence of an oracle A such that

$$\mathrm{SPACE}^{S(A)}[L(n)] \neq \mathrm{NLOG}^{S(A)}[L(n)]$$

or alternatively

$$\mathrm{SPACE}^{D(A)}[L(n)] \neq \mathrm{NLOG}^{D(A)}[L(n)]$$

implies that

$$\mathrm{LOG} \neq \mathrm{NLOG}.$$

This is quite different from the corresponding situation for time bounded computations and suggests that the

$$\mathrm{LOG} \stackrel{?}{=} \mathrm{NLOG}$$

problem may be solvable by known techniques. The recent Immerman-Szelepcsényi success, showing NLOG = $\overline{\mathrm{NLOG}}$, similarly suggests that the space bounded computations may be easier to understand than the corresponding time bounded computations. We believe that it is an opportune time to renew an attack on the $\mathrm{LOG} \stackrel{?}{=} \mathrm{NLOG}$ problem.

On the other hand, if LOG $\neq$ NLOG, then there exist oracles which collapse and oracles which separate the higher ($L(n) \geq n$) deterministic and nondeterministic space bounded classes. This leads to the strange possibility that the proof of

$$\mathrm{LOG} \stackrel{?}{=} \mathrm{NLOG}$$

could be achieved by known proof techniques, but because of the contradictory relativization (even in the $S(\cdot)$-model) the

$$\mathrm{SPACE}[n] \stackrel{?}{=} \mathrm{NSPACE}[n]$$

problem could not be solvable by the same technique. We return to this problem in the next section.

4 Proof of Results

To outline proofs of our theorems we recall a few definitions and results.

Let GAP represent the set of directed graphs with an **in** and an **out** node such that there is a directed path from **in** to **out**. The following was observed in [Sav70].

Lemma 3 GAP is complete for NLOG under $(\log n)$-space bounded reductions.

Proof: Clearly, GAP is in NLOG. To see that A in NLOG can be reduced to GAP, let $L(N) = A$, where N is an NLOG machine. For any x, we will construct a directed graph, $G_{x,N}$, polynomially large in the size of x, such that $x \in A$ iff $G_{x,N} \in$ GAP. The nodes of the graph are the configurations of N: the state of the machine, the content of the work tape and the head positions on the input tape and work tape. There is a directed edge from node a to node b iff there is a legal move from configuration a to configuration b when N is working on input x. Note that only the symbol read by the input head, not the entire input string x, is represented in the configurations.

Clearly, the number of configurations of N on input x is polynomial in the length of x, and the graph $G_{x,N}$ can be computed by a deterministic $(\log n)$-space bounded machine from input x (and printed on a one-way output tape). Also, x is in $L(N)$ iff there is a directed path in $G_{x,N}$ from the "starting" node to the unique "accepting" node (or from the **in** to the **out** node). □

We now show that in the $S(\cdot)$ model and $D(\cdot)$ model an oracle cannot separate LOG from NLOG if LOG = NLOG.

Lemma 4 LOG = NLOG $\iff$ $(\forall A)[\,\mathrm{LOG}^{S(A)} = \mathrm{NLOG}^{S(A)}\,]$.

Proof:

($\Leftarrow$) This part of the proof is obvious when we let A be the empty oracle.

($\Rightarrow$) If LOG = NLOG, then GAP is in LOG. We now show that an $S(A)$ oracle computation cannot do any harm. For an $N^{S(A)}(x)$ computation, we can again construct a directed graph of configurations. Here, the configurations are augmented by the query tape (of length $\log |x|$), and the graph includes two transition edges, labelled **yes** or **no**, after each configuration in the oracle query state. Again, $N^{S(A)}$ accepts iff there is a path in this graph from the **in** to the **out** node using only the correct **yes** and **no** edges, as determined by A.

Since LOG = NLOG, a $\mathrm{LOG}^{S(A)}$ machine can recognize such graphs. The deterministic machine proceeds as a deterministic GAP recognizer and consults its oracle on the query edges to decide which edges are legally present in the graph, as the computation demands. Therefore,

$$\mathrm{LOG}^{S(A)} = \mathrm{NLOG}^{S(A)}.$$

□

A similar proof shows that the same result holds for the $D(\cdot)$ oracle access model. The crucial observation here is that a $D(\cdot)$ query is computed deterministically and is specified completely by the machine configuration Q at the start of the deterministic query computation. This query computation is not encoded in the graph. Instead, the node Q is followed by the two appropriate **yes** and **no** edges. Clearly, a $\mathrm{LOG}^{D(A)}$

machine can recognize if such a graph is in GAP by recomputing the oracle query and deciding which edge (**yes** or **no**) is consistent with A.

This proof fails for the unrestricted oracle access mechanism in which an NLOG^A machine can query exponentially many strings in A and the LOG^A machine can query only polynomially many strings.

Next we extend this result to higher deterministic and nondeterministic tape bounded classes. So we may highlight the simplicity (and elegance) of these ideas, we discuss only the linearly bounded tape classes. With a few technical details, the same proof can be used for all other space constructible bounds $L(n) \geq \log n$.

Lemma 5 $\mathrm{LOG} = \mathrm{NLOG} \Longrightarrow (\forall A)[\ \mathrm{SPACE}^{S(A)}[n] = \mathrm{NSPACE}^{S(A)}[n]\ ]$.

Proof: Again, we let the linear-space bounded machines have a read-only input tape, a linearly bounded read-write work tape and a separate linearly bounded oracle tape.

Given a nondeterministic n-space bounded machine $N_i^{S(A)}$ and input x, we construct an $\mathrm{NLOG}^{S(A)}$ machine $N_{\sigma(i)}^{S(A)}$ such that $N_i^{S(A)}$ accepts x iff $N_{\sigma(i)}^{S(A)}$ accepts $x\#^{2^{|x|}-|x|}$. By the previous lemma we know that for $N_{\sigma(i)}^{S(A)}$ there exists an equivalent deterministic $(\log n)$-space bounded machine $M_{\rho(i)}^{S(A)}$. However, there exists a deterministic linear-space bounded machine $M_{\delta(i)}^{S(A)}$ equivalent to $M_{\rho(i)}^{S(A)}$. Thus, $\mathrm{SPACE}^{S(A)}[n] = \mathrm{NSPACE}^{S(A)}[n]$. $\Box$

The same proof extends to all space constructible $L(n) \geq \log n$ and to the $D(\cdot)$ model, but not to the unrestricted oracle access model.

Lemma 6 $\mathrm{LOG} \neq \mathrm{NLOG} \Longrightarrow (\exists A)[\ \mathrm{SPACE}^{S(A)}[n] \neq \mathrm{NSPACE}^{S(A)}[n]\ ]$.

Proof: Assuming $\mathrm{LOG} \neq \mathrm{NLOG}$, for each LOG machine M_i there exist arbitrarily large counterexample graphs G_i such that

$$G_i \in \mathrm{GAP} \iff G_i \notin L(M_i).$$

For each M_i we choose a counterexample graph G_i with edge descriptions of size $n_i \geq 2^{n_{i-1}}$. We define A to be the set consisting of the edges in the graphs G_i, $i \geq 1$. Note that

$$A^{=n_i} = \{\ e \in A \mid |e| = n_i\ \}$$

precisely encodes the graph G_i. (We leave it to the reader to verify that A can be recognized in $\mathrm{SPACE}[2^n]$ by an algorithm which diagonalizes over the M_i's.)

Consider the language

$$\Delta = \{\ 1^n \mid A^{=n} \text{ encodes some graph } G \text{ and } G \in \mathrm{GAP}\ \}.$$

We show that

$$\Delta \in \mathrm{NSPACE}^{S(A)}[n] - \mathrm{SPACE}^{S(A)}[n].$$

Clearly, $\Delta \in \mathrm{NSPACE}^{S(A)}[n]$. Suppose $\Delta \in \mathrm{SPACE}^{S(A)}[n]$ and is recognized by some linear space machine $M^{S(A)}$. From M we construct a logspace machine M_j which on input G with edges of size $n = \log(|G|)$ simulates $M^{S(A)}(1^n)$.

- When $M^{S(A)}(1^n)$ queries an edge e of size n, $M_j(G)$ looks for e in G.
- When $M^{S(A)}(1^n)$ queries a string x of size $> n$, $M_j(G)$ assumes the oracle answers "no".
- When $M^{S(A)}(1^n)$ queries a string x, $\log n < |x| < n$, $M_j(G)$ also assumes the oracle answers "no".
- When $M^{S(A)}(1^n)$ queries a string x, $|x| \leq \log n$, $M_j(G)$ computes directly whether $x \in A$. (Recall that $A \in \mathrm{SPACE}[2^n]$.)

Note that $M^{S(A)}(1^n)$ and $M_j(G)$ each have $O(n)$ workspace. Also, for each graph G_i encoded in A, $M_j(G_i)$ agrees with $M^{S(A)}(1^{n_i})$. In particular,

$$\begin{aligned} M_j \text{ accepts } G_j &\iff M \text{ accepts } 1^{n_j} \\ &\iff G_j \in \mathrm{GAP}, \end{aligned}$$

contrary to the choice of G_j as a counterexample for M_j. □

Observe that the same oracle also separates $\mathrm{NSPACE}^{D(A)}[n]$ from $\mathrm{SPACE}^{D(A)}[n]$ on the same language Δ provided that we made the gaps between n_i and n_{i+1} sufficiently large so that on input 1^{n_i} the graph G_{i+1} cannot be accessed (even though the $D(A)$-oracle access mechanism allows 2^{n_i} long queries).

Finally, for a PSPACE complete set R and space constructible $L(n) \geq n$,

$$\mathrm{SPACE}^{D(R)}[L(n)] = \mathrm{NSPACE}^{D(R)}[L(n)]$$

and

$$\mathrm{SPACE}^{S(R)}[L(n)] = \mathrm{NSPACE}^{S(R)}[L(n)].$$

This sketch completes the proofs for Theorems 1 and 2.

We now discuss the possible difference in proving $\mathrm{LOG} \neq \mathrm{NLOG}$ and $\mathrm{SPACE}[n] \neq \mathrm{NSPACE}[n]$. Recall that

$$\mathrm{LOG} \neq \mathrm{NLOG} \iff \mathrm{GAP} \notin \mathrm{LOG}.$$

On the other hand, we will show that $\mathrm{NSPACE}[n]$ and $\mathrm{SPACE}[n]$ are different if and only if there exists a sparse, easily computable subset of GAP in $\mathrm{NLOG} - \mathrm{LOG}$.

To make "easily computable" precise, let M_u be a standard universal TM (with a one-way output tape) and define [Har83]

$$\begin{aligned}\mathrm{KS}[\log m, \log m] = \{\ w \mid |w| = m \text{ and } (\exists y)[\ |y| \le \log m \text{ and } \\ M_u(y) = w \text{ is computed on } \log m \text{ tape }]\ \}.\end{aligned}$$

Lemma 7 SPACE$[n]$ = NSPACE$[n]$ $\iff$ GAP $\cap$ KS$[\log m, \log m] \in$ LOG.

Proof:
$(\Leftarrow)$ Observe that, like NLOG, any NSPACE$[n]$ computation can also be described by a graph, except this graph has edges with description size n (as specified by the configurations of n-space bounded machine). In addition, the graph is computable from the input x, $|x| = n$, and the machine description. Since the graph is exponentially larger than x and is computable in space linear in $|x|$ (not counting output tape), the graph itself is in KS$[\log m, \log m]$. So, if GAP $\cap$ KS$[\log m, \log m] \in$ LOG, a SPACE$[n]$ machine can determine if any graph describing an NSPACE$[n]$ computation is in GAP. Thus, NSPACE$[n]$ = SPACE$[n]$.

$(\Rightarrow)$ Consider the following set of short descriptions:

$$C = \{\ y \mid M_u(y) = G \text{ using } |y| \text{ tape, } |y| \le \log(|G|) \text{ and } G \text{ is a graph in GAP }\}.$$

Clearly, $C \in$ NSPACE$[n]$. By assumption, NSPACE$[n]$ = SPACE$[n]$, so $C \in$ SPACE$[n]$. Now, a LOG machine can determine if $G \in$ GAP $\cap$ KS$[\log m, \log m]$, by checking if any string y, $|y| \le \log(|G|)$, is in C. (Such a y exists iff $G \in$ GAP $\cap$ KS$[\log m, \log m]$.) $\square$

Thus, SPACE$[n] \neq$ NSPACE$[n]$ if and only if NLOG and LOG are separated on the sparse set of graphs in GAP which can easily be computed from exponentially shorter descriptions.

As stated earlier, it may be the case that NLOG $\neq$ LOG can be proved by known techniques, but the proof does not extend to linear-space bounded computations. That is, the separation of LOG from NLOG may not be as easily provable on the sparse set

$$\mathrm{GAP} \cap \mathrm{KS}[\log m, \log m].$$

Such a situation could arise, for example, if one could prove that NLOG and LOG differ on random graphs in GAP. Since random graphs (with high probability) are not computable from shorter strings, the separation of NLOG from LOG may not be extendable to a separation of NSPACE$[n]$ from SPACE$[n]$.

References

[Bei87] R. Beigel. Bounded queries to SAT and the Boolean hierarchy. Technical Report 7, Department of Computer Science, The Johns Hopkins University, 1987. To appear in *Theoretical Computer Science*.

[BGS75] T. Baker, J. Gill, and R. Solovay. Relativizations of the P =? NP question. *SIAM Journal on Computing*, 4(4):431–442, December 1975.

[Cha90] R. Chang. An example of a theorem that has contradictory relativizations and a diagonalization proof. *Bulletin of the European Association for Theoretical Computer Science*, 42:172–173, October 1990.

[Har83] J. Hartmanis. Generalized Kolmogorov complexity and the structure of feasible computations. In *Proceedings of the IEEE Symposium on Foundations of Computer Science*, pages 439–445, 1983.

[Har85] J. Hartmanis. Solvable problems with conflicting relativizations. *Bulletin of the European Association for Theoretical Computer Science*, 27:40–49, Oct 1985.

[Hem87] L. A. Hemachandra. The strong exponential hierarchy collapses. In *ACM Symposium on Theory of Computing*, pages 110–122, 1987.

[HH76] J. Hartmanis and J. E. Hopcroft. Independence results in computer science. *ACM SIGACT News*, pages 13–24, October-December 1976.

[Hop84] J. E. Hopcroft. Turing machines. *Scientific American*, pages 86–98, May 1984.

[HS65] J. Hartmanis and R. E. Stearns. On the computational complexity of algorithms. *Transactions of the AMS*, 117:285–306, 1965.

[Imm88] N. Immerman. Nondeterministic space is closed under complementation. *SIAM Journal on Computing*, 17(5):935, October 1988.

[Kad88] J. Kadin. *Restricted Turing Reducibilities and the Structure of the Polynomial Time Hierarchy*. PhD thesis, Cornell University, February 1988.

[KL87] B. Kirsig and K. J. Lange. Separation with the Ruzzo, Simon, and Tompa relativization implies DSPACE[$\log n$] $\neq$ NSPACE[$\log n$]. *Information Processing Letters*, 25:13–15, 1987.

[LFKN90] C. Lund, L. Fortnow, H. Karloff, and N. Nisan. Algebraic methods for interactive proof systems. In *Proceedings of the IEEE Symposium on Foundations of Computer Science*, pages 2–10, 1990.

[LL76] R. E. Ladner and N. A. Lynch. Relativization of questions about log space computability. *Mathematical Systems Theory*, 10:19–32, 1976.

[LSH65] P. M. Lewis II, R. E. Stearns, and J. Hartmanis. Memory bounds for recognition of context-free and context-sensitive languages. In *Proceedings of the Sixth Annual Symposium on Switching Circuit Theory and Logical Design*, pages 191–202, 1965.

[RS81] C. W. Rackoff and J. I. Seiferas. Limitations on separating nondeterministic complexity classes. *SIAM Journal on Computing*, 10:742–745, 1981.

[RST84] W. L. Ruzzo, J. Simon, and M. Tompa. Space-bounded hierarchies and probabilistic computations. *Journal of Computer and System Sciences*, 28(4):216–230, November 1984.

[Sav70] W. J. Savitch. Relationships between nondeterministic and deterministic tape complexities. *Journal of Computer and System Sciences*, 4(2):177–192, 1970.

[Sha90] A. Shamir. IP = PSPACE. In *Proceedings of the IEEE Symposium on Foundations of Computer Science*, pages 11–15, 1990.

[SHL65] R. E. Stearns, J. Hartmanis, and P. M. Lewis II. Hierarchies of memory limited computations. In *Proceedings of the Sixth Annual Symposium on Switching Circuit Theory and Logical Design*, pages 179–190, 1965.

[Sim77] I. Simon. *On Some Subrecursive Reducibilities.* PhD thesis, Stanford University, March 1977.

[Sze88] R. Szelepcsényi. The method of forced enumeration for nondeterministic automata. *Acta Informatica*, 26:279–284, 1988.

[Wag88] K. Wagner. Bounded query computations. In *Proceedings of the 3rd Structure in Complexity Theory Conference*, pages 260–277, June 1988.

[Wil85] C. Wilson. Relativized circuit complexity. *Journal of Computer and System Sciences*, 31(2):169–181, October 1985.

[Wil86] C. Wilson. Parallel computation and the NC hierarchy relativized. In *Structures in Complexity Theory*, volume 223 of *Lecture Notes in Computer Science*, pages 362–382. Springer-Verlag, 1986.