Structural Complexity Column

Some Observations about Relativization of Space Bounded Computations

Juris Hartmanis*

in collaboration with
R. Chang, J. Kadin, and S. Mitchell

88-912
May 1988

Department of Computer Science
Cornell University
Ithaca, NY 14853-7501

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

Structural Complexity Column

Some Observations about Relativization of Space Bounded Computations

by
J. Hartmanis*

in collaboration with
R. Chang, J. Kadin and S. Mitchell

Cornell University
Department of Computer Science
Ithaca, NY 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 that can be relativized in two contradictory ways are very hard to solve. We know that such problems can only be solved by proof techniques that do not relativize and thus, 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]. Unfortunately, very many important problems in complexity theory have contradictory relativizations and thus have to be viewed

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

as inaccessible to our current proof techniques. The classic example is the $P \stackrel{?}{=} NP$ problem for which Baker, Gill and Solovay [BGS75] exhibited recursive oracles A and B such that

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

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

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 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 some program committees have been unkind to other sound work in relativization. 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, there are quite a few beautiful relativization results that 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 well as a study of how the structure of the accessed information affects this power. Finally, relativization results can be viewed as not yet fully understood "independence" results, pinpointing the difficulty of problems.

A nice example revealing 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]:

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

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 $k \geq 1$,

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

Some other interesting results about various access mechanisms to SAT have been observed in [Wag87]. Let LOG^{SAT} denote the set of languages accepted by a deterministic $\log(n)$ tape bounded machine with a one-way oracle tape. Then

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

2 Relativization of Space Bounded Computations

The recent result by Immerman and Szelepcsényi [Imm87,Sze87] 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 and also in the "Formal Language Theory Column" and "The Structural Complexity Column" by Salomaa and Hartmanis, respectively. It seems that the possibility of contradictory relativizations of this problem was not explored, which for the right relativization model would have been impossible to obtain. This could have suggested that this problem can be, with ingenuity, solved by "known techniques", as it turned out.

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 yielded 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 $L(n) \geq \log(n)$:

$$NSPACE[L(n)] \subseteq SPACE[L(n)^2],$$

and therefore

$$PSPACE = NPSPACE\ ,$$

in contrast to our belief that

$$P \neq NP.$$

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

It is easily shown that

$$LOG \subseteq NLOG \subseteq P.$$

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

$$NLOG^A \not\subseteq P^A.$$

Intuitively, we can see that nondeterministically $NLOG^A$ can query all strings of length n in A, and P^A can query only polynomially many such strings. This permits an A such that

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

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

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

as well as the Immerman-Szelepcsényi result,

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

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

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

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 with 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 is not accepting $\overline{\Delta}$, and we add no strings of length n to C.

All these violations, and in particular the $NLOG^A \not\subseteq P^A$ 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 it is required that the computation of the oracle query must be done deterministically by both types of machines (i.e., during the query computation phase the machine computes deterministically). Denote the corresponding language classes by $LOG^{D(A)}$ and $NLOG^{D(A)}$. Now both types of machines can, for any given input, 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 permits queries exponentially long in the input length and looks artificial.

Finally, the simplest and most natural oracle access mechanism is to limit the oracle queries to the length of the work tape. Denote the corresponding language classes by $LOG^{S(A)}$ and $NLOG^{S(A)}$. In this model the nondeterministic machines can use nondeterministically computed queries.

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

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

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

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

This is a generalization of a result in [Sim77] and later 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,

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

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

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

(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

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

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

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

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

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

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

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

or alternatively

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

implies that

$$LOG \neq NLOG.$$

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

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

problem may be solvable by known techniques. The recent Immerman-Szelepcsényi success, showing $NLOG = \overline{NLOG}$, 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 $LOG \stackrel{?}{=} NLOG$ 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

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

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

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

problem would not be so easily solvable. 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 1 *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 x is not represented in the configurations, only the single symbol read by the input head .

Clearly, the number of configurations for 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 2 $LOG = NLOG \iff (\forall A)\ [\ LOG^{S(A)} = NLOG^{S(A)}\ ]$.

Proof

($\Leftarrow$) This part of the proof is obvious when we set $A = \emptyset$.

($\Rightarrow$) If *LOG* = *NLOG* then *GAP* is in *LOG*. We now show that an $S(A)$ oracle computation cannot do any harm. For an $N^{S(A)}(\mathrm{x})$ computation we can again compute 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 $LOG^{S(A)}$ 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,

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

□

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 therefore determined 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 $LOG^{D(A)}$ machine can recognize if such a graph is in GAP by recomputing the oracle query and determining which of the *YES* or *NO* edge in the graph 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. Not to obscure the simplicity (or 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 3 $LOG = NLOG \Rightarrow (\forall A)\ [\ SPACE^{S(A)}[n] = NSPACE^{S(A)}[n]]$.

Proof

Let the linearly space bounded machines again 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 $NLOG^{S(A)}$ 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, $SPACE^{S(A)}[n] = NSPACE^{S(A)}[n]$. □

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

Lemma 4 $LOG \neq NLOG \Rightarrow (\exists A)\ [\ SPACE^{S(A)}[n] \neq NSPACE^{S(A)}[n]]$.

Proof

Assuming $LOG \neq NLOG$, for each LOG machine M_i there are arbitrarily large counterexample graphs G_i such that

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

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 of edges of the graphs G_i, $i \geq 1$. Note that $A^{=n_i} = \{e \in A |\ |e| = n_i\}$ precisely encodes the graph G_i. (We leave it to the reader to verify that A can be recognized in $SPACE[2^n]$ by an algorithm which diagonalizes over the M_i's.)

Consider the language

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

We show that

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

Clearly, $\Delta \in NSPACE^{S(A)}[n]$. Suppose $\Delta \in SPACE^{S(A)}[n]$ 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 $A \in SPACE[2^n]$.)

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,

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

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

Observe that the same oracle also separates $NSPACE^{D(A)}[n]$ from $SPACE^{D(A)}[n]$ 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, it is easily shown that for a $PSPACE$ complete set R and space constructible $L(n) \geq n$,

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

and

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

This completes the sketch of the proofs for Theorems 1 and 2.

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

$$LOG \neq NLOG \iff GAP \notin LOG.$$

On the other hand, we will show that $NSPACE[n]$ and $SPACE[n]$ are different if and only if there exists a sparse, easily computable subset of GAP in $NLOG - LOG$. To make "easily computable" precise, let M_u be a standard universal TM (with a one-way output tape) and define [Har83]

$$KS[\log(m), \log(m)] = \{w| \quad |w| = m \text{ and } (\exists y)[|y| \leq \log(m) \text{ and } M_u(y) = w \text{ is computed on } \log(m) \text{ tape}]\}.$$

Lemma 5 $SPACE[n] = NSPACE[n] \iff GAP \cap KS[\log(m), \log(m)] \in LOG$.

Proof

($\Leftarrow$) Observe that, like $NLOG$, any $NSPACE[n]$ computation can also be described by a graph, except that this graph has edges with description size n (as determined by the configurations of n-space bounded machine) and that 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:

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

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, $|y| \leq \log(|G|)$, 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 that $NLOG \neq LOG$ and that this can be proved by known techniques, but the proof does not extend to linearly space bounded computations. That is, the separation of LOG from $NLOG$ may not be easily provable on the sparse set

$$GAP \cap KS[\log(m), \log(m)].$$

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. Manuscript, June 1987.

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

[Har83] J. Hartmanis. Generalized Kolmogorov complexity and the structure of feasible computations. In *Proceedings 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, October 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. Hartmains and R. E. Stearns. On the computational complexity of algorithms. *Transactions of the AMS*, 117:285–306, 1965.

[Imm87] N. Immerman. Nondeterministic space is closed under complement. Technical Report DCS-TR-552, Yale University, July 1987. To appear in *SIAM Journal on Computing*.

[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.

[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.

[SHL65] R. E. Stearns, 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.

[Sze87] R. Szelepcsényi. The method of forcing for nondeterministic automata. *The Bulletin of the European Association for Theoretical Computer Science*, 33:96–100, October 1987.

[Wag87] K. Wagner. Bounded query classes. Technical Report TR157, University of Augsburg, October 1987.

[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*, Lecture Notes in Computer Science #223, pages 362–382. Springer-Verlag, 1986.