Space Bounded Computations: Review And New Separation Results*

Desh Ranjan Richard Chang Juris Hartmanis

Computer Science Department
Cornell University

December 20, 1989

Abstract

In this paper we review the key results about space bounded complexity classes, discuss the central open problems and outline the prominent proof techniques. We show that, for a slightly modified Turing machine model, low level deterministic and nondeterministic space bounded complexity classes are different. Furthermore, for this computation model, we show that Savitch's theorem and the Immerman-Szelepcsényi theorem do not hold in the range $\lg\lg n$ to $\lg n$. We also present other changes in the computation model which bring out and clarify the importance of space constructibility. We conclude by enumerating open problems which arise out of the discussion.

1 Introduction

Computational complexity theory is the study of the quantitative laws governing computing. The two most important complexity measures in this study are time and space (or memory) needed for the computation.

The central structural concept in complexity theory is the complexity class, which consists of all the languages recognizable within a given resource bound. Some of the hardest open problems in computer science are questions about containments between various complexity classes defined by different resource bounds. Among these problems, the most notorious are the open questions about the differences between deterministic and nondeterministic time and space bounded computations. The classic open problems are

$$SPACE[\lg n] \stackrel{?}{=} NSPACE[\lg n] \stackrel{?}{=} P \stackrel{?}{=} NP \stackrel{?}{=} PSPACE$$

*This research was supported by NSF Research Grants DCR-85-20597 and CCR-88-23053.

among which , clearly $P \stackrel{?}{=} NP$ is the most famous and important. More recently, similar questions have been posed about the relationship between sequential and parallel computational complexity classes. The most well-known of these are

$$NSPACE[\lg n] \stackrel{?}{=} NC \stackrel{?}{=} P.$$

Finally, we point out that the oldest problem of this type is the classic problem about linear-bounded automata [10, 6].

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

In this paper we concentrate on space bounded computations for two main reasons. First, there have been some exciting recent developments in the study of space bounded computations, to which we add new separation results. Second, we are encouraged by the recent developments and believe that a much deeper understanding of space bounded computations can be obtained with a heroic attack on these problems. So, this paper should be viewed, partially, as a call to arms for an all-out attack on these classic open problems.

2 Space-Bounded Computations

In this section, we review what is known about space bounded computations. In particular we will show how the structure of low-level space bounded computations relates to the structure of higher-level space bounded computations.

We consider the Turing machine model with a two-way, read-only input tape and a separate two-way, read-write worktape. This model was introduced in 1965 [15, 7] to study the rich set of computations requiring less than linear space. Let $SPACE[S(n)]$ and $NSPACE[S(n)]$ denote respectively the classes of languages recognizable by deterministic and nondeterministic Turing machines using no more than $S(n)$ worktape on any input of length n.

From early work [15], we know that the recognition of non-regular sets requires at least $\lg \lg n$ space, and that all context-free languages can be recognized in space $(\lg n)^2$ [11].

Theorem 1

1. *There exist non-regular sets in* $SPACE[\lg \lg n]$.
2. *If a non-regular set* A *is in* $SPACE[S(n)]$, *then*

$$\sup_{n \to \infty} S(n)/\lg \lg n > 0.$$

The proof of this theorem shows that our Turing machine model is "physically" incapable of using an unbounded amount of space if the space bound does not exceed $c \lg \lg n$ for some $c > 0$. For example, no machine can mark off $\sqrt{\lg \lg n}$ space on its worktape, using no more than $\sqrt{\lg \lg n}$ space.

Definition 1 *A function $S(n)$ is fully space constructible if there exists a Turing machine which for all inputs of length n marks off $S(n)$ space, using no more than $S(n)$ space.*

It has been shown that no monotone unbounded function below $\lg n$ is fully space constructible by deterministic Turing machines [2, 13]. For example, $\lg \lg n$ cannot be so constructed, because given any $SPACE[\lg \lg n]$ machine M and sufficiently large n, M on input $\$1^n\$$ does not have enough configurations to traverse the input tape from left to right without going into a cycle—i.e., repeating the same worktape configuration and machine state. Note that the machine's configuration at the end of the traversal depends only on the length of input modulo the length of this cycle. Since the length of the cycle must be less than n, the machine reaches the same configuration at the right end for inputs $\$1^n\$$ and $\$1^{n+n!}\$$. So, in a left-to-right traversal of the input tape, M cannot distinguish $\$1^n\$$ from $\$1^{n+n!}\$$. By repeating this argument, one can show that the machine's behaviour is the same on both inputs. In particular, M uses the same number of tape cells. Thus, M fails to mark off $\lg \lg(n + n!)$ space on input $\$1^{n+n!}\$$, and so $\lg \lg n$ is not fully space constructible.

We will refer to this argument as the $n \to n + n!$ trick [15]. This same technique can be used to show that no monotone unbounded function below $\lg n$ is fully-space constructible by deterministic machines. In Section 4 we consider constructibility by nondeterministic machines.

Another reason for the study of space bounded computations is that it offers a classic example of a resource bounded hierarchy [15].

Theorem 2 (Space Hierarchy Theorem)
For fully space constructible $S(n)$,

$$\inf_{n \to \infty} R(n)/S(n) = 0 \Longrightarrow SPACE[R(n)] \subset SPACE[S(n)].$$

The theorem above shows that every additional amount of space allows Turing machines to recognize more languages. On the other hand, we do not know if the addition of nondeterminism is as useful a resource as space. The best result relating nondeterminism and space as resources was discovered in 1970 [12]. It showed that the difference between deterministic and nondeterministic space is quadratically bounded. In contrast, an analogous result for time bounded computations would imply that $P = NP$.

Theorem 3 (Savitch)
For $S(n) \geq \lg n$, $NSPACE[S(n)] \subseteq SPACE[S(n)^2]$.

We do not know whether this relationship holds for space bounds below $\lg n$. In fact, we show in Section 4 that Savitch's theorem fails below $\lg n$ for certain TM models.

Recently, Immerman and Szelepcsényi showed independently and by a very elegant proof that nondeterministic space is closed under complementation [8, 16].

Theorem 4 (Immerman-Szelepcsényi)
For $S(n) \geq \lg n$, $NSPACE[S(n)] = co\text{-}NSPACE[S(n)]$.

Again, an analogous result for time bounded computations would imply that $NP = co\text{-}NP$ and that the Polynomial Hierarchy collapses. In Section 4, we will investigate whether the Immerman-Szelepcsényi theorem holds below $\lg n$.

As an additional bonus, the Immerman-Szelepcsényi result permits an easy proof of the hierarchy theorem for nondeterministic space bounded computations. (Closure under complementation implies that a larger nondeterministic machine can simulate a smaller nondeterministic machine and decide if the smaller machine *rejects*. Hence the larger machine can diagonalize against the smaller one.) Without the Immerman-Szelepcsényi theorem, the proof of Theorem 5 can be very cumbersome. See [7] for an example.

Theorem 5 *For fully space constructible $S(n) \geq \lg n$,*

$$\inf_{n\to\infty} R(n)/S(n) = 0 \Longrightarrow NSPACE[R(n)] \subset NSPACE[S(n)].$$

Finally, we want to point out that under relativization space bounded computations behave radically differently from time bounded computations [19, 20] .

Theorem 6 *If $SPACE[\lg n] = NSPACE[\lg n]$, then for all oracles A*

$$SPACE^A[\lg n] = NSPACE^A[\lg n].$$

On the other hand, even if $P = NP$, we know that there exists A such that $P^A \neq NP^A$ [1]. In general, we believe that problems with contradictory relativizations, such as $P \stackrel{?}{=} NP$, $P \stackrel{?}{=} PSPACE$ etc. are inherently hard and may require new proof techniques for their solution. In this light, the above result may indicate that the $SPACE[\lg n] \stackrel{?}{=} NSPACE[\lg n]$ problem, which does not have contradictory relativizations (if they are equal), might be solved using known techniques. For a detailed discussion of relativization of space bounded computations see [5].

3 Low Level Tape Bounded Computations

The unsuccessful struggle to solve the classic separation problems has convinced many that we do not yet understand computation well enough for a direct attack on these problems and should therefore concentrate on simpler models of computation. In this vein, the study of circuit complexity is vigorously pursued with the hope that insights gained from understanding these apparently simpler models can be used to solve the classic separation problems. In the same spirit, we will now concentrate on $\lg \lg n$ space bounded computations—the lowest level of interesting space bounded computations.

Before we review the results on $\lg \lg n$ space computations, we prove the following lemma about the distribution of primes. This lemma will be used throughout the rest of the paper.

Lemma 1
$(\forall m)\ 2^{2^{m-1}} \leq \prod_{p_i \leq 2^m} p_i \leq 2^{2^{m+1}}$, *where* p_i *denotes the i-th prime.*

Proof:

1. $\prod_{p_i \leq 2^m} p_i \leq 2^{2^{m+1}}$. The inequality is easy to check for $m = 1, 2, 3, 4$. Let $\pi(x)$ denote the number of primes less than or equal to x. Then, by a theorem due to Rosser and Schoenfield,

   $$(\forall x)\ (x \geq 17)\ x/\ln x\ \leq \pi(x)\ \leq 1.23x/\ln x.$$

   Then, $\prod_{p_i \leq 2^m} p_i \leq (2^m)^{\pi(2^m)}$. However, $m > 4$, so

   $$\pi(2^m) \leq 1.23(2^m)\lg e/m \quad \text{or} \quad \pi(2^m) \leq 2^{m+1}/m.$$

   Therefore $\prod_{p_i \leq 2^m} p_i \leq (2^m)^{2^{m+1}/m} \leq 2^{2^{m+1}}$.

2. $\prod_{p_i \leq 2^m} p_i \geq 2^{2^{m-1}}$. This is a well-known inequality which also follows trivially from the Rosser-Schoenfield theorem by induction. ∎

Very roughly, the above result asserts that the product of the first k primes is about 2^{p_k}. We can use this lemma to show that there is a $SPACE[\lg \lg n]$ machine M which marks off $\Theta(\lg \lg n)$ tape cells infinitely often on unary input. On input 1^n, M looks for the first prime number p_k which does not divide n. M simply checks each successive prime number p_i to see if p_i divides n and stops when it finds p_k. Since it takes only $|p_i|$ space to check for primality, M uses $|p_k|$ tape cells. The lemma says that $|p_k|$ will always be bounded by $\lg \lg n$, because if $p_1, p_2, \ldots, p_{k-1}$ divide n, then the product $p_1 p_2 \cdots p_{k-1}$ divides n. So, n must be larger than this product, which

is in turn bounded below by 2^{p_k}. Moreover, for infinitely many n's (for example, $n = p_1 p_2 \cdots p_{k-1}$ for any k) $|p_k|$ is $\Theta(\lg \lg n)$.

We can use this same routine to prove that if $SPACE[\lg \lg n] = NSPACE[\lg \lg n]$, then all higher deterministic and nondeterministic space classes are equal. This result has been part of the complexity theory folklore for some time, but the first formal proof appeared in [18]. This result establishes the importance of even the lowest level of space bounded computations.

Theorem 7
If $SPACE[\lg \lg n] = NSPACE[\lg \lg n]$ *then* $SPACE[\lg n] = NSPACE[\lg n]$.

Proof: Suppose $SPACE[\lg \lg n] = NSPACE[\lg \lg n]$. Let A be any language in $NSPACE[\lg n]$. We will construct a $SPACE[\lg n]$ algorithm for A. Define A', the padded version of A, by

$$A' = \{x\#^N \mid N = p_1 p_2 \cdots p_k t,\ p_i \text{ is the } ith \text{ prime},\ |p_k| \geq \lg |x|,\\ p_{k+1} \text{ does not divide } t, \text{ and } x \in A\}.$$

$A' \in NSPACE[\lg \lg n]$, because an $NSPACE[\lg \lg n]$ machine can use the routine described above to mark off $\lg \lg N$ space. If N is sufficiently large, $|p_k| \geq \lg |x|$. Then, the $NSPACE[\lg \lg n]$ machine has enough space to simulate the $NSPACE[\lg n]$ acceptor for A on input x.

By hypothesis, $SPACE[\lg \lg n] = NSPACE[\lg \lg n]$, so some $SPACE[\lg \lg n]$ machine M must accept A'. We can use M to construct a $SPACE[\lg n]$ machine M' that recognizes A. M' simulates M on input x until M tries to leave the input and enter the (non-existent) $\#^N$ part. Then, M' determines which configuration M will be in when it returns to the x region of the input. This yields a deterministic $\lg n$ space algorithm for A, and hence $SPACE[\lg n] = NSPACE[\lg n]$. ∎

For any $S(n) \geq \lg n$, it is trivial to show that

$$SPACE[S(n)] = \text{co-}SPACE[S(n)].$$

However, the trivial proof does not extend to space bounds below $S(n)$, because it is possible for a machine to reject by cycling—i.e., loop forever through configurations that will never reach any accepting state. For $S(n) \geq \lg n$, this cycling does not create any problems because we can always force the machine to halt by making it count the number of configurations. For $S(n) < \lg n$, the number of machine configurations will still be at least n, because the input head can be in n different positions. However, with less than $\lg n$ bits, the machine cannot count up to n. So, the standard configuration counting argument does not work. Nevertheless, Sipser showed by an elegant argument that all deterministic space bounded classes are indeed closed under complementation [14]. We state a special case of Sipser's theorem.

Theorem 8 $SPACE[\lg \lg n] = co\text{-}SPACE[\lg \lg n]$.

Proof: (Sketch). For a detailed proof see [14] . The proof of this theorem is based on the observation that the $SPACE[\lg \lg n]$ machine accepts if and only if there is a "backwards" path from the unique accepting configuration to the unique initial configuration. (By "backwards", we mean that the path begins with the accepting configuration and follows the transition table in reverse.) In addition, no "backward" path containing the accepting configuration can cycle because the accepting configuration itself cannot be in a cycle and because the "fork" in the backwards path entering a cycle would constitute a forward nondeterministic move by the $SPACE[\lg \lg n]$ machine. Thus, a depth first search algorithm can detect if there is a backwards path from the accepting configuration to the initial configuration. Carefully implemented, the depth-first-search algorithm needs only as much space as the original machine. (The algorithm must determine how much space the original machine uses without cycling itself. This is accomplished by a similar depth first search algorithm. This algorithm looks for a "backwards" path that leads from a configuration where the machine is adding an extra worktape cell to the initial configuration.) So, every $SPACE[\lg \lg n]$ machine can be replaced by an equivalent machine that always halts. Finally, these equivalent machines can be easily complemented by exchanging the accepting and rejecting configurations. ∎

4 The Dot Model

We have shown that $\lg \lg n$ is not fully space constructible by deterministic machines. In this section, we consider constructibility by nondeterministic machines.

Definition 2 *A function $S(n)$ is fully space constructible nondeterministically, if there is a nondeterministic Turing machine which is $S(n)$ space bounded and uses exactly $S(n)$ space on at least one computation path on every input of length n.*

We do not know if $\lg \lg n$ is fully space constructible by a nondeterministic machine. If so, then

$$SPACE[\lg \lg n] \neq NSPACE[\lg \lg n],$$

because the language $A = \{a^n b^m \mid n \neq m\}$ would be in $NSPACE[\lg \lg n]$ but not $SPACE[\lg \lg n]$. First, $A \in NSPACE[\lg \lg n]$ because given any two numbers n and m, $n \neq m$ if and only if there is a prime p_i, $|p_i| \leq \lg \lg(n+m)$, such that $n \neq m \pmod{p_i}$. (See [3] for a related theorem proved with this technique.) On the other hand, $A \notin SPACE[\lg \lg n]$ by the $n \rightarrow n + n!$ trick. For sufficiently large n, if a $SPACE[\lg \lg n]$ machine correctly rejects $a^n b^n$, then it must also reject $a^n b^{n+n!}$. However, $a^n b^{n+n!} \in A$, so A is not recognized by any $SPACE[\lg \lg n]$ machine.

We now introduce a slightly modified computational model for which we will show a strong separation of deterministic and nondeterministic complexity classes. The separation result will be based on the difference in tape bound constructibility by deterministic and nondeterministic versions of our model.

Definition 3 *A 1-inkdot Turing machine is a standard Turing machine with the additional power of marking 1 tape-cell on the input (with an inkdot!). This tape-cell is marked once and for all (no erasing !!) and no more than one dot of ink is available. The action of the machine depends on the current state, the currently scanned input and worktape symbols and the presence of the inkdot on the currently scanned tape-cell. The action consists of moving the heads and making appropriate changes on worktape cells (using the finite control). In addition, the inkdot may be used to mark the currently scanned cell on the input tape if it has not been used already.*

We now establish the following theorem.

Theorem 9 *There is a 1-inkdot $S(n)$ space bounded nondeterministic Turing machine N, which on input 1^n marks off $S(n)$ worktape cells on some computation path, where*

$$\lfloor \lg \lfloor \lg n \rfloor \rfloor \leq S(n) \leq \lfloor \lg \lfloor \lg n \rfloor \rfloor + 2.$$

Hence, $\lg \lg n$ is fully space constructible by nondeterministic 1-inkdot Turing machines.

Proof: N on any input x first checks if x is of the form 1^n. If so, N, nondeterministically places the inkdot somewhere on the input. Now N pretends that the inkdot is the right endmarker of the input tape. In other words, if N places the inkdot on the k^{th} position on the input, N pretends it is reading 1^k. From this point on N behaves deterministically. N repeatedly checks if k is divisible by each prime $2, 3, 5 \ldots$ and halts when it finds the first prime which does not divide k.

We claim that N is the desired machine. Let $\$1^n\$$ be any input to N. Choose m so $2^{2^m} \leq n < 2^{2^{m+1}}$. First, we show that there is a computation path on which N marks off $m = \lfloor \lg \lfloor \lg n \rfloor \rfloor$ worktape cells. Let $k = \prod_{p_i \leq 2^{m-1}} p_i$. By Lemma 1, $k \leq 2^{2^m} < n$, so some computation path will place the inkdot at position k. But the first prime which does not divide k has size m. (Recall that for any c there is a prime between c and $2c$). Thus, N uses exactly m worktape cells on this computation path.

Next, we show that N uses no more than $m + 2 = \lfloor \lg \lfloor \lg n \rfloor \rfloor + 2$ worktape cells on any computation path. Suppose N places the inkdot at position k of the input. Then the first prime that does not divide k must have size less than or equal to $m+2$, because by Lemma 1, $\prod_{p_i \leq 2^{m+2}} p_i \geq 2^{2^{m+1}} > n \geq k$. Thus, some prime of size up to $m+2$ does not divide k. So, N uses no more than $\lfloor \lg \lfloor \lg n \rfloor \rfloor + 2$ worktape cells. ∎

We can now exploit this space constructibility result to obtain a separation result.

Definition 4

$$SPACE^*[S(n)] = \{L \mid L \text{ is accepted by an } S(n) \text{ space bounded deterministic 1-inkdot Turing machine } \},$$

$$NSPACE^*[S(n)] = \{L \mid L \text{ is accepted by an } S(n) \text{ space bounded nondeterministic 1-inkdot Turing machine } \}.$$

Theorem 10 $SPACE^*[\lg\lg n] \subset NSPACE^*[\lg\lg n]$.

Proof: Since $\lg\lg n$ is fully space constructible by a nondeterministic dot machine, we know that $\{a^n b^m \mid n \neq m\} \in NSPACE^*[\lg\lg n]$. But $\{a^n b^m \mid n \neq m\}$ is not in $SPACE^*[\lg\lg n]$. To see this, let n be sufficiently large and $a^n b^n$ be rejected by M^*. Then M^* will place the dot in either the a^n or the b^n part. Without loss of generality, assume that the dot is placed in the a^n part of the input. Then by the standard $n \to n + n!$ trick, we know that M^* will also reject $a^n b^{n+n!}$ which should be accepted. Thus $\{a^n b^m \mid n \neq m\} \in NSPACE^*[\lg\lg n] - SPACE^*[\lg\lg n]$. ∎

The proof actually shows that $\{a^n b^m \mid n \neq m\}$ is not in any $SPACE^*[S(n)]$ with $\lim_{n\to\infty} S(n)/\lg n = 0$. This implies that Savitch's theorem does not hold for the dot computation model in the range below $\sqrt{\lg n}$ space bounds. (Recall that it is not known if Savitch's theorem holds in this range for the standard space bounded models.)

Similarly, the Immerman-Szelepcsényi theorem does not hold for the dot model in the low space bound range. To see this, simply verify that $\{a^n b^n \mid n \geq 1\} \notin NSPACE^*[\lg\lg n]$ because, for sufficiently large n, if $a^n b^n$ is accepted then there is an accepting computation for $a^n b^{n+n!}$. Again, it is not known if the Immerman-Szelepcsényi result hold for the standard space bounded models below $\lg n$. The results established here are similar to those proved in [9].

Next, we show that deterministic machines do not gain any additional computational power from the dot capability. We state just a special case of this result.

Theorem 11 $SPACE^*[\lg\lg n] = SPACE[\lg\lg n]$.

Proof (outline): Let M be a $\lg\lg n$ dot machine. M' is a standard $\lg\lg n$ machine which will simulate M without using the inkdot. The simulation of M by M' is straightforward until the dot is placed on the input tape by M. At this time, M' records the configuration of the worktape, state of M and symbol being scanned on the input tape. Note that M' cannot remember where the dot was placed on the input tape, if it is forced to move away from the dot. In the following simulation the dot's position will be recomputed again and again.

After the placement of the dot by M, the forward simulation by M' continues until M scans an input tape symbol that could have a dot mark. In this case, M' records the current machine and worktape configuration of M and retrieves the configuration it saved when M used the inkdot. Then, M' runs a depth-first, backwards search to determine if it is possible to back M up from this configuration to the initial configuration. If this is possible, then M' simulates M forward until M reaches the dot writing operation—now the input head is back on the cell with the dot. Then M' switches to the stored configuration of M and continues the forward simulation of M. If it is not possible to back up to the initial configuration, then the depth first search

will halt with head in the original position. So, M' can continue the simulation of M knowing that M did not place the inkdot on the current cell.

Thus, M' can simulate M without using any inkdots. Therefore, $L(M) = L(M')$ and $SPACE^*[\lg\lg n] = SPACE[\lg\lg n]$. ∎

Corollary 1 $SPACE^*[\lg\lg n] = co\text{-}SPACE^*[\lg\lg n]$.

The above result leaves us with a fascinating question :

$$NSPACE^*[\lg\lg n] \stackrel{?}{=} NSPACE[\lg\lg n].$$

Should this be the case, we have a major separation result,

$$NSPACE[\lg\lg n] \neq SPACE[\lg\lg n].$$

5 Space Constructibility And Demon Machines

To emphasize the importance of the constructibility of tape bounds, we consider Turing machine models where the worktape is automatically marked off by a demon.

Definition 5 *A demon $S(n)$ machine is a Turing machine with a two-way, read-only input tape and a two-way, read-write worktape enclosed in endmarkers $S(n)$ apart for inputs of length n.*

Even though a $\lg\lg n$ demon machine cannot count up to the n, $\{a^n b^n \mid n \geq 1\}$ can be accepted by a $\lg\lg n$ demon machine. Furthermore, this much space is required.

Theorem 12

1. $\{a^n b^n \mid n \geq 1\} \in DEMONSPACE[\lg\lg n]$.
2. *if $S(n)$ is monotone increasing and $\sup_{n\to\infty} S(n)/\lg\lg n = 0$ then*

$$\{a^n b^n \mid n \geq 1\} \notin DEMONSPACE[S(n)].$$

Proof:

1. To see this, recall that $m \neq n \iff (\exists p_i)\ m \neq n \pmod{p_i}$, p_i a prime, and $|p_i| \leq \lg\lg(m+n)$. Once the $\lg\lg(m+n)$ worktape is marked off (automatically), the demon machine can test if $m \neq n \pmod{p_i}$ for some prime that can be written on the available tape. If no such prime is found, then $m = n$.

2. For each input tape position k, we define the crossing sequence of a machine M to be the sequence of configurations that M reaches when the input head is at position k. We use a counting argument on the number of crossing sequences to show that for any $S(n)$ spaced bounded demon machine M, where $S(n)$ is $o(\lg \lg n)$, there exist r and n, such that $r < n$ and M accepts $a^r b^n$. The crossing sequence argument will show that not only will M repeat configurations (as in the $n \rightarrow n + n!$ trick) but entire sequences of configurations as well.

   Choose a large n where M accepts $a^n b^n$. For some constant c, the total number of unique configurations that M can reach on inputs of length $2n$ is less than $c^{S(n)}$. Since M accepts $a^n b^n$, it cannot repeat any configuration at any input position (otherwise M will loop and reject). So, every crossing sequence is at most $c^{S(n)}$ long. Thus, the total number of unique crossing sequences is bounded by $(c^{S(n)})^{c^{S(n)}}$. However, $S(n)$ is $o(\lg \lg n)$, so for sufficiently large n, this upper bound is less than n. Thus, there must be two input positions, k_1 and k_2, $k_1 < k_2 < n$, where M repeats the same crossing sequence. If we delete the symbols in the input string between positions k_1 and k_2, we would not change M's accepting behaviour. Thus, M must accept $a^{n-(k_2-k_1)}b^n$. Therefore, M does not recognize $\{a^n b^n \mid n \geq 1\}$. ∎

$\{a^n b^n \mid n \geq 1\}$ is a rather curious language—especially when it is compared to $\{w\#w \mid w \in (a+b)^*\}$. In the standard Turing machine model, the space complexity of both languages is $\Theta(\lg n)$. However, $w\#w$ is harder than $a^n b^n$ in the sense that a one-tape Turing machine requires $\Omega(n^2)$ time to recognize $w\#w$, but can recognize $a^n b^n$ in $O(n \lg n)$ time. We prove a similar result using demon machines. We show that $\lg \lg n$ space bounded demon machines can recognize $\{a^n b^n \mid n \geq 1\}$, but not $\{w\#w \mid w \in (a+b)^*\}$. The latter requires $\Omega(\lg n)$ space. Thus, the $\Omega(\lg n)$ lower bound (for standard Turing machines) for the space complexity of $\{a^n b^n \mid n \geq 1\}$ is due to space constructibility properties, whereas the same lower bound for $\{w\#w \mid w \in (a+b)^*\}$ is independent of any constructibility property.

Theorem 13 *If $S(n)$ is monotone increasing and $\sup_{n\rightarrow\infty} S(n)/\lg n = 0$ then*

$\{w\#w \mid w \in (a+b)^*\} \notin DEMONSPACE[S(n)]$.

Proof: There are 2^n strings of form $w\#w$, $|w| = n$. For any $S(n)$ space bounded demon machine M, the number of configurations that M can reach on inputs of length $2n$ is bounded by $c^{S(n)}$, for some constant c. As before, choose an n where M accepts $w\#w$, for all w, $|w| = n$. Again, the number of possible crossing sequences is bounded by $(c^{S(n)})^{c^{S(n)}}$. However, $S(n)$ is $o(\lg n)$, so, for sufficiently large n, the number of crossing sequences is less than 2^n. So, there must be two strings w_1 and w_2, $w_1 \neq w_2$, such that, when M accepts $w_1\#w_1$ and $w_2\#w_2$, M has the same crossing sequence at the # symbol. This implies that M must also accept $w_1\#w_2$. So, M does not recognize $\{w\#w \mid w \in (a+b)^*\}$. ∎

6 Pebble Machines

In the previous section we showed the importance of space constructibility for machines restricted to $o(\lg n)$ space. We know that, if $\sup_{n\to\infty} S(n)/\lg n = 0$, then $S(n)$ can not be constructed fully by deterministic Turing machines. In this section we introduce a natural model which *can* construct such functions. Similar models have been studied before [7] .

Theorem 11 showed that deterministic $\lg\lg n$ machines do not gain any additional power from the use of one inkdot. This theorem is similar to the well-known result [7] that two-way finite automata do not gain any computing power from the use of a "pebble"—a movable marker placed on the input tape. The situation changes, however, if the pebble machine is given a worktape.

Definition 6 *A pebble machine is a Turing machine with a two-way, read-only input tape, a two-way, read-write worktape and one pebble which can be placed on and removed from the input tape. The action of the Turing machine depends on the current state, the currently scanned input and worktape symbols, and the presence of the pebble on the currently scanned input tape cell. The action consists of changing the symbol on the worktape, moving the input and worktape heads, and picking up or placing (or neither) the pebble on the currently scanned input tape cell according to its finite control.*

We assume that the machine aborts if it ever tries to use more than one pebble.

Definition 7

$$PEBBLESPACE[S(n)] = \{L \mid L \text{ is accepted by a pebble machine which is } S(n) \text{ space bounded}\},$$
$$NPEBBLESPACE[S(n)] = \{L \mid L \text{ is accepted by a nondeterministic pebble machine which is } S(n) \text{ space bounded}\}.$$

It is easy to see that if $S(n) \geq \lg n$ then $PEBBLESPACE[S(n)] = SPACE[S(n)]$. We now show that the pebble does give the $\lg\lg n$ space bounded machine additional computing power.

Theorem 14 $SPACE[\lg\lg n] \subset PEBBLESPACE[\lg\lg n]$.

Proof: We show that $\{a^n b^n \mid n \geq 1\} \in PEBBLESPACE[S(n)]$ by showing that $\lg\lg n$ is constructible by the pebble machines. To construct $\lg\lg n$, M on 1^n places the pebble at position k and finds the first prime which does not divide k. It repeats this procedure for $k = 1, 2, \ldots, n$. Then, by the proof of Theorem 11, M uses $\Theta(\lg\lg n)$ space. Thus, $\{a^n b^n \mid n \geq 1\} \in PEBBLESPACE[\lg\lg n]\}$. ∎

Finally, as a side note, we point out that deterministic pebble machines are closed under complementation. The proof uses Sipser's trick and follows the same lines as the proof of Theorem 11.

$PEBBLESPACE[S(n)] = \text{co-}PEBBLESPACE[S(n)]$.

7 Open Problems

The preceding discussion leaves us with a rich set of open problems. We list some of them here.

1. Is $\lg\lg n$ (or any monotone unbounded $S(n)$ with $\sup_{n\to\infty} S(n)/\lg n = 0$) fully space constructible by nondeterministic Turing machines ? A positive answer to this question would imply that $SPACE[\lg\lg n] \subset NSPACE[\lg\lg n]$. This was observed by [17] and the proof is similar to the proof of Theorem 10.

2. Is $NSPACE^*[\lg\lg n] = NSPACE[\lg\lg n]$?

   This would similarly separate $NSPACE[\lg\lg n]$ and $SPACE[\lg\lg n]$.

3. $DEMONSPACE[\lg\lg n] \stackrel{?}{=} NDEMONSPACE[\lg\lg n]$.

   $PEBBLESPACE[\lg\lg n] \stackrel{?}{=} NPEBBLESPACE[\lg\lg n]$.

4. Are there any space bounds below $\lg\lg n$ that are fully space constructible by a pebble machine ?

5. What are the relationships between the various deterministic and nondeterministic complexity classes defined by space, dot, demon and pebble machines?

6. For which of the above classes are Immerman-Szelepcsényi and Savitch's theorems valid ?

8 Conclusion

In this paper we discussed space bounded computations and showed that for the dot model of space bounded computation $\lg\lg n$ is strongly space constructible by nondeterministic Turing machines but not by deterministic Turing machines. This was achieved by exploiting the $n \to n + n!$ method. This lead to the separation of $SPACE^*[\lg\lg n]$ and $NSPACE^*[\lg\lg n]$. We also showed that Savitch's theorem and the Immerman-Szelepcényi theorem do not hold for this computation model in the low complexity range. We then discussed other useful models for space classes below $SPACE[\lg n]$ and proved some results for these models which demonstrate the importance of space constructibility for low-level complexity classes. These results

suggest new open problems and focus some attention on the old open problems. We hope that this work will encourage a systematic attack on the open problems about space bounded computations. We believe that considerable progress can be made at these problems and that there is hope for solving the general deterministic and nondeterministic space problem, especially in the low complexity range.

9 Acknowledgements

We would like to thank Ken Regan, Wei Li and Alessandro Panconesi for active participation in interesting discussions and providing several bits of oracular advice. The first author would like to thank the students of the Spring 1989 class of CS782 at Cornell University for listening to some of the ideas presented in this paper and providing constructive criticism about them.

10 Update

In a paper submitted to *ICALP '90*, Viliam Geffert from University of P. J. Šafárik, Czechoslovakia has a proof which shows that no monotone unbounded function below $\lg n$ is space constructible even via nondeterministic Turing machines [4]. This settles the first two questions posed in the section on Open Problems. Note that this does not rule out the possibility of separating nondeterministic and deterministic space classes using space constructibility results. In fact, if $S(n)$ is any space bound constructible by nondeterministic machines but not by deterministic Turing machines then $NSPACE[S(n)] \neq SPACE[S(n)]$. The language $\{0^k 1^n \mid k \leq S(n)\}$ is the desired witness in this case. In fact, whether there are functions that are space constructible by nondeterministic Turing machines but not by deterministic Turing machines is an interesting question by itself.

References

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

[2] A. R. Freedman and R. E. Ladner. Space bounds for processing counterless inputs. *Journal of Computer and System Sciences*, 11:118–128, 1975.

[3] R. Freivalds. On the worktime of deterministic and non-deterministic turing machines. *Latvijskij Matematiceskij Eshegodnik*, 23:158–165, 1979.

[4] V. Geffert. Nondeterministic computations in sublogarithmic space and space constructibility. To appear in *ICALP '90*.

[5] J. Hartmanis, R. Chang, J. Kadin, and S. Mitchell. Some observations about space bounded computations. *Bulletin of the EATCS*, 35:82–92, June 1988.

[6] J. Hartmanis and H. H. Hunt. On the LBA problem and its importance in the theory of computation. *SIAM-AMS*, 7:1–26, 1974.

[7] J. E. Hopcroft and J. D. Ullman. *Introduction to Automats Theory, Languages and Computation.* Addison-Wesley Publishing Company, 1979.

[8] Neil Immerman. Nondeterministic space is closed under complement. In *Proceedings of Structure in Complexity Theory Third Annual Conference*, pages 112–115. Computer Society of IEEE, 1988.

[9] R. Kannan. Alternation and the power of nondeterminism. In *ACM Symposium on Theory of Computing*, pages 344–346, 1983.

[10] S. Y. Kuroda. Classes of languages and linearly-bounded automata. *Information and Control*, 7:207–223, 1964.

[11] P. M. Lewis II, R. E. Stearns, and J. Hartmanis. Memory bounds for recognition of context-free and context-sensitive languages. In *IEEE Conference Record on Switching Circuit Theory and Logic Design*, pages 191–202, 1965.

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

[13] J. Seiferas. A note on notions of tape constructibility. Technical Report CSD-TR 187, Pennsylvania State University, 1976.

[14] M. Sipser. Halting space-bounded computations. *Theoretical Computer Science*, 10:335–338, 1980.

[15] R. E. Stearns, J. Hartmanis, and P. M. Lewis II. Hierarchies of memory limited computations. In *1965 IEEE Conference Record on Switching Circuit Theory and Logical Design*, pages 179–190, 1965.

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

[17] A. Szepietowski. Some notes on strong and weak $\log\log n$ space complexity. Technical report, Mathematical Department, Technical University of Gdańsk, Majakowskiego 11/12, PL-80-952 Gdańsk, Poland, 1988.

[18] A. Szepietowski. If deterministic and nondeterministic space complexity are equal for $\log\log n$ then they are equal for $\log n$. In *STACS '89*, pages 251–255, 1989. Springer-Verlag *Lecture Notes in Computer Science #349*.

[19] C. B. Wilson. Relativized circuit complexity. *Journal of Computer and System Sciences*, 31:169–181, 1985.

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