Space Bounded Computations: Review And New Separation Results

Juris Hartmanis*
Desh Ranjan**

TR 89-1002
May 1989

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

---

*This research was supported by the National Science Foundation research grant DCR 85-20597.

*This research was supported by the National Science Foundation research grant DCR 85-20597.

Space Bounded Computations : Review And New Separation Results

J. Hartmanis ^{*}   Desh Ranjan [†]
Computer Science Department
Cornell University

May 7, 1989

Abstract

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

---

^{*}Supported by NSF Research Grant DCR 85-20597

[†]Supported by NSF Research Grant DCR 85-20597

The central structural concept in complexity theory is the complexity class, which consists of all the languages recognizable within a given resource bound. The hardest open problems in computer science seem to be 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

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

among which, clearly $P \stackrel{?}{=} NP$ is the most famous and important. More recent interesting open problems are about the relations between sequential and parallel computational complexity classes. Most well-known of these are

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

The oldest problem of this type is the classic linear-bounded automata problem [7,4]

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

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, encouraged by the recent developments we 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 allout attack on the classic open problems in this area.

2 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 [12,5] 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 the early work [12] 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 $(\lg n)^2$ [8].

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*

$$\sup_{n \rightarrow \infty} S(n) / \lg \lg n = 0.$$

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 can be shown that no unbounded function below $\lg n$ is fully space-constructible by deterministic Turing machines [10,1]. To see this, consider, for example, $\lg \lg n$. For any deterministic $\lg \lg n$ space-bounded Turing machine M , for sufficiently large n , M on 1^n , if it traverses the whole input, cycles (repeating the worktape configuration and machine state). Therefore, M reaches the input end-marker in the same configuration on 1^n and $1^{n+n!}$. The same argument repeats if M comes back to the input start-marker. As the length of the cycle is less than n each time, it divides $n!$ and hence behaviour of M on 1^n and $1^{n+kn!}$ is exactly the same. In particular the space used is the same, which shows $\lg \lg n$ is not fully space-constructible.

We will refer to this as the $n \rightarrow n + n!$ proof method. It was first used in [12]. It is not known if there are functions below $\lg n$ that are fully

space-constructible by nondeterministic machines. We will return to these problems.

For space-bounded computations there are very sharp heirarchy results [12] .

Theorem 2 *For fully space-constructible $S(n)$,*

$$\inf_{n \rightarrow \infty} R(n)/S(n) = 0 \implies SPACE[R(n)] \subsetneq SPACE[S(n)].$$

We know, since 1970, that the difference between deterministic and nondeterministic space is quadratically bounded [9]. We do not believe that this is true for time-bounded computations, as such a result would imply that $P = NP$, which would have drastic and unexpected consequences.

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

It is not known if this relationship holds for space-bounds below $\lg n$ and we will show later that for certain TM models it does not hold below $\lg n$.

More recently, it was shown, to a great surprise of many of us, with a very elegant proof, that nondeterministic space is closed under complement [6,13] .

Theorem 4 *For $S(n) \geq \lg n$, $NSPACE[S(n)] = coNSPACE[S(n)]$.*

Again we do not believe that this relation holds for nondeterministic time-bounded computations as such a relation would imply $NP = coNP$ with drastic and unexpected consequences. Similarly, we do not yet know if this result holds for $S(n) \geq \lg n$ and we will show that for some Turing machine models it doesn't. The Immerman-Szelepcsényi result permits an easy derivation of sharp heirarchy results for nondeterministic space-bounded computations. Since nondeterministic space complexity classes are closed under complement we can use direct diagonalization, which was not possible before.

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

$$\inf_{n \rightarrow \infty} R(n)/S(n) = 0 \implies NSPACE[R(n)] \subsetneq NSPACE[S(n)].$$

To see how much more cumbersome the old proofs were, consult [5] .

Finally, it is interesting to observe that space bounded computations behave radically differently from the time-bounded computations under relativization [16,17] .

Theorem 6 $SPACE[\lg n] = NSPACE[\lg n] \implies (\forall A)SPACE^A[\lg n] = NSPACE^A[\lg n]$.

On the other hand, even if $P = NP$, we know that there exists A such that $P^A \neq NP^A$.

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), may be susceptible to the presently known techniques. For a detailed discussion of relativization of tape-bounded computations see [3] .

Clearly, if $SPACE[\lg n] = NSPACE[\lg n]$ then by a simple padding argument we can show that for all well-behaved $S(n) \leq \lg n$, $SPACE[S(n)] = NSPACE[S(n)]$.

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 therefore we should concentrate on simpler models of computation. For example, the study of circuit complexity is pursued with vigour in hope to understand these, apparently simpler models, and then use the insights to solve the classic separation problems. Clearly, if SAT is not recognizable by any family of polynomial-size circuits then $P \neq NP$.

In the same spirit, we will now concentrate on the lowest space-bounded computations of all space-bounded computations. The distribution of the prime numbers plays a dominant role in the following considerations, and the following lemma is used time and again later.

Lemma 1 $(\forall m) \prod_{p_i \leq 2^m} p_i \leq 2^{2^{m+1}}$, 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,

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

$$\text{Then, } \prod_{p_i \leq 2^m} p_i \leq (2^m)^{\pi(2^m)}$$

$$\text{But as } m > 4, \pi(2^m) \leq 1.23(2^m) \lg e / m$$

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

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

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

The above result very roughly speaking, asserts that the product of the first k primes is about 2^{p_k} . Therefore we will never use more than $\lg \lg n$ space to write down (one after the other) successive primes $2, 3, \dots$, checking for divisibility of the length of the input by each of these, and stop as soon as we find one which does not divide the length of the input. Thus, the non-regular set

$$A = \{1^n \mid (\exists k) \ n = p_1 p_2 \dots p_k p_{k+2} t \text{ such that } p_{k+1} \text{ does not divide } t\}$$

is in $SPACE[\lg \lg n]$. The deterministic machine recognizing this set tests if the length of the input 1^n is divisible by $p_1 = 2, p_2 = 3, \dots$ until it finds p_k not dividing n . Then, it checks that p_{k+1} divides n . By the lemma, we know that the machine will use at most $\lfloor p_{k+1} \rfloor$ space which is less than or equal to $\lg \lg n + 1$. Note that we can check if a binary string represents a prime within the size of the string.

The same phenomenon can be used to prove that the collapse of the lowest deterministic and nondeterministic space-bounded classes implies the collapse of higher deterministic and nondeterministic space classes. This result has been in the folklore and a formal proof appears in [15]. Clearly, this result establishes the relevance of the lowest level space-bounded computations in general.

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

Proof: Let $A \in NSPACE[\lg n]$. Let $\#$ be a new symbol and let $A' = \{x\#^N \mid N = p_1 p_2 \dots p_k, p_i \text{ is the } i\text{th prime and } p_i \leq \lg |x|\}$

Then, because of the lemma, we see that $A' \in NSPACE[\lg \lg n]$. But then by the hypothesis of the theorem, $A' \in SPACE[\lg \lg n]$.

Let M be the deterministic $\lg \lg n$ machine accepting A' . We can use M to construct a deterministic $\lg n$ space machine M' to accept A . M' simulates M on input x , if M tries to leave the input and enter the (non-existent) $\#^n$ part M' uses its $\lg |x|$ tape to simulate this computation by counting to determine if and how M will re-enter the input x . This yields a deterministic $\lg n$ space algorithm for A , and hence $SPACE[\lg n] = NSPACE[\lg n]$. ■

It is easily seen that for all fully space-constructible $S(n) \leq \lg n$,

$$SPACE[S(n)] = coSPACE[S(n)]$$

since we can eliminate cycling for any $S(n)$ space bounded M by counting (on separate worktape tracks) the number of steps M is using and halting if it exceeds the number of possible machine configurations for M . But the situation is much more complicated for space classes below $\lg n$, because we do not have enough space to count upto the number of possible configurations and hence to prevent cycling. Still, Sipser showed by a very elegant argument that all deterministic space-bounded classes are indeed closed under complement [11]. We state a special case of this result.

Theorem 8 $SPACE[\lg \lg n] = coSPACE[\lg \lg n]$.

For a proof see [11]. We state the basic ideas of the proof here as they are used later. The proof is based on the observation that such devices can get into cycles when run forward, but not when they are “going backward” starting from a configuration which we know is not in a cycle. Therefore, if we consider the graph where the vertices are configurations and the edges are valid transitions of the machine, the connected component of the graph containing the accept configuration is a tree. Thus we start from the (unique) accepting configuration and search backwards (using depth-first search) if we can reach the start configuration. If the machine uses space

k , then this search can be carried out in space k because the depth-first search does not require any extra space as previous node can be computed from machine transitions. But a problem arises as we do not know *a priori* how much space the machine is going to use. This is overcome by doing the search for $k = 1, 2, \dots$ with the search for $k = n + 1$ being carried out only if we first ensure that the machine uses at least that much space. This can be done by cycling over all configurations which have used space n and are going to use space $n + 1$ on the next step and checking if from any of these the start configuration can be reached by the depth-first search.

4 The Dot Model

It is known, as shown earlier, that $\lg \lg n$ is not fully space-constructible by deterministic 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 .*

At present, it is not known if $\lg \lg n$ is fully space-constructible by a nondeterministic machine. Should this be the case, then

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

To see this observe that if $\lg \lg n$ is fully space constructible nondeterministically then $\{a^n b^m \mid n \neq m\} \in NSPACE[\lg \lg n]$. Since $n \neq m$ iff there is a prime p_i , $(p_i \mid \lg \lg(m + n))$ and $n \not\equiv m \pmod{p_i}$, this can be verified in $\lg \lg n$ space. This proof technique has been used before in [2] to prove a related result. On the other hand, the same language is not in $SPACE[\lg \lg n]$. To see this, observe that if $a^n b^n$ is rejected and n is sufficiently large, then any $\lg \lg n$ machine must reject $a^n b^{n+n!}$, which should be accepted.

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 version of our new computational model.

Definition 3 A 1-inkdot Turing machine is a usual Turing machine with the additional power of marking 1 tape-square on the input (with an inkdot!). This tape-square 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, currently scanned input and worktape symbols and presence of the inkdot on the currently scanned tape-square and consists of moving the heads and making appropriate changes on worktape squares (using the finite control) ; in addition the inkdot may be used to mark the currently scanned square on the input tape if it has not already been used.

We now establish the following theorem.

Theorem 9 There is a 1-inkdot $S(n)$ space-bounded non-deterministic Turing machine N , which on input 1^n scans $S(n)$ worktape squares 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 N uses $O(\lg \lg n)$ space on each input 1^n on some computation path, and no more on the others.

Proof: N on any input x first checks if x is of the form 1^n . If yes, N , nondeterministically, places the inkdot somewhere on the input. Now N treats as its effective input the string from the start to the place where it put the inkdot. In other words, if N places the inkdot on the k -th position on the input, effective input for N is 1^k . Now onwards N behaves deterministically. N repeatedly checks if k is divisible by each of the primes $2, 3, 5, \dots$. It halts when it finds the first prime which does not divide k .

We now claim that N is the desired machine.

Let 1^n be any input to N . Let m be such that $2^{2^m} \leq n < 2^{2^{m+1}}$. We first show that there is a computation path on which N scans $m = \lfloor \lg \lfloor \lg n \rfloor \rfloor$ work tape squares. If N places the inkdot non-deterministically at position $k = \prod_{p_i \leq 2^{m-1}} p_i$, then the first prime which does not divide k has size m . (As $\forall x$ there is a prime between x and $2x$). Note that by the lemma $k \leq 2^{2^m} \leq n$.

Hence N uses m worktape squares.

Again we show that N uses no more than $m+2 = \lfloor \lg \lfloor \lg n \rfloor \rfloor + 2$ worktape squares on any computation path.

Suppose N places the inkdot at position k on the input. Then the first prime that does not divide k has size less than or equal to $m + 2$, as by the lemma, $\prod_{p_i \leq m+2} p_i \leq 2^{2^{m+1}} > n \geq k$, and hence not all primes of size upto $m + 2$ can divide k . Hence N uses no more than $\lfloor \lg \lfloor \lg n \rfloor \rfloor + 2$ worktape squares. ■

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

Definition 4 $SPACE^*[S(n)] = \{L \mid L \text{ is accepted by a } S(n) \text{ space-bounded deterministic 1-inkdot Turing machine}\}$.

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

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 not place the dot in the a^n or b^n part. Without loss of generality, assume that the dot is not placed in the b^n part of the input. Then by the standard $n \rightarrow n + n!$ method, we know that M^* will also reject $a^n b^{n+n!}$ which should be accepted. Thus $\{a^n b^m \mid n \neq m\} \in NSPACE^*[\lg \lg n] - SPACE^*[\lg \lg n]$. ■

From the proof, we can actually see that $\{a^n b^m \mid n \neq m\}$ is not contained in any $SPACE^*[S(n)]$ with $\lim_{n \rightarrow \infty} S(n)/\lg n = 0$.

From this we see 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 classic space bounded models. Similarly, the Immerman-Szelepcsényi theorem does not hold for the dot model in the low space bound range. To see this, recall 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 classic space bounded models below $\lg n$.

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

case of this result, the proof of which was pointed out by Richard Chang of Cornell University.

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

Proof:

(outline) : Let M be a $lg\ lg\ n$ dot machine. M' is a regular $lg\ lg\ n$ machine which will simulate M without the use of the inkdot. The simulation of M by M' is straight forward until (if at all) the dot is placed on the inpt 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 an input tape symbol is scanned that could have a dot mark. In this case, the machine and worktape configuration is recorded and a depth-first, backwards search is initiated to determine if we can back M up from this configuration to the initial configuration. If this is possible, then M' simulates M forward until M reaches the dot operation; now we are back on the square with the dot on it; 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 we know that the dot was not placed on this tape square and in this case the depth first search will halt with head in the position from where it started. We now continue the simulation of M , knowing that there is no inkdot on this square.

From this outline of the simulation process we see that $L(M) = L(M')$ and therefore $SPACE^*[lg\ lg\ n] = SPACE[lg\ lg\ n]$. ■

From the above it follows

Corollary 1 $SPACE^*[lg\ lg\ n] = coSPACE^*[lg\ lg\ n]$.

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 machines with marked off work tapes.

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 work tape enclosed in end-markers $S(n)$ apart for inputs of length n .*

The $\lg \lg n$ demon machines cannot count upto the length n , but, quite surprisingly they can accept $\{a^n b^n | n \geq 1\}$. Furthermore, this much space is required.

Theorem 12 1. $\{a^n b^n | n \geq 1\} \in DEMONSPACE[\lg \lg n]$.

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

Proof:

1. To see this, recall that $m \neq n \iff (\exists p_i) m \neq n \pmod{p_i}$ for a prime p_i , $|p_i| \leq \lg \lg(m+n)$. Clearly, once we have $\lg \lg(m+n)$ tape marked-off (automatically) we 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. Because $S(n)$ is a slow growing function, it is easy to see that for every m , there is an $n \leq m$ such that $S(2n) = S(n)$.

We shall use a counting argument on the number of crossing sequences to show that, for sufficiently large n , any demon machine with $o(\lg \lg n)$ space, trying to accept $\{a^n b^n | n \geq 0\}$ makes an error. The basic idea will be to show that, because of extremely small amount of space available, we just don't have enough different crossing sequences. Hence, crossing sequences repeat. A crossing sequence here is a sequence of configurations (q, W) , where q is a state of the machine and W its worktape-configuration.

Let M be a demon $S(n)$ machine accepting $\{a^n b^n \mid n \geq 1\}$. Let Q_M be the set of states of M and let Γ be its worktape alphabet. Let $f(n)$ denote the number of different possible configurations for M on an input of length n . Then $f(n) \leq |Q_M| |\Gamma|^{S(n)} S(n) \leq c^{S(n)}$ for suitable c and all large enough n .

Now, choose n large enough and such that $S(2n) = S(n)$. Then consider the crossing sequences induced by the accepting computation of M on $a^n b^n$. Since, there is no more than $S(n)$ space available, there are no more than $c^{S(n)}$ possible configurations of M on $a^n b^n$. As no crossing sequence induced by an accepting path can have a configuration repeated (M , being deterministic, will cycle otherwise), no crossing sequence has length more than $c^{S(n)}$. Therefore, total number of possible crossing sequences of M on $a^n b^n$ are no more than

$$c^{S(n)} + (c^{S(n)})^2 + \dots + (c^{S(n)})^{c^{S(n)}}.$$

But this is less than $2(c^{S(n)})^{c^{S(n)}}$ which is less than n , for sufficiently large n , as $S(n)$ is $o(\lg \lg n)$. Hence, we have at least two identical crossing sequences on a^n part of the input. Then, if we remove the part of the input between these two crossing sequences, we get a string $a^k b^n$, with $k < n$, which M accepts. Hence, M makes an error. ■

It is interesting to note that with the standard Turing machine model the same language requires, and can be recognized in, $O(\lg n)$ space. A very similar language $\{w \# w \mid w \in (a + b)^*\}$ also requires, and can be recognized in, $O(\lg n)$ space. Still, intuitively, $\{w \# w \mid w \in (a + b)^*\}$ seems harder than $\{a^n b^n \mid n \geq 1\}$. This has been justified previously, by showing that on a single tape Turing machine, we require at least $O(n^2)$ time to recognize $\{w \# w \mid w \in (a + b)^*\}$ whereas $\{a^n b^n \mid n \geq 0\}$ can be recognized in $O(n \lg n)$ time. We now prove a result of similar nature. We show that even with the demon machines $\{w \# w \mid w \in (a + b)^*\}$ requires at least $O(\lg n)$ space.

Theorem 13 *if $S(n)$ is monotone increasing and $\sup_{n \rightarrow \infty} S(n)/\lg n = 0$ then $\{w \# w \mid w \in (a + b)^*\} \notin \text{DEMONSPACE}[S(n)]$.*

Proof: There are 2^n strings of form $w \# w$, $|w| = n$. On the other hand, since $S(n)$ is $o(\lg n)$, again we have arbitrarily large n such that

$S(2n + 1) = S(n)$. Then for any demon $S(n)$ machine, M , there is an n such that there are at most $2(c^{S(n)})^{c^{S(n)}}$ crossing sequences on inputs of length $2n + 1$, for suitable c . Then, we have $w\#w$ and $x\#x$, $w \neq x$ such that crossing sequence of M at the $\#$ is same in the accepting computations for $w\#w$ and $x\#x$. But then M accepts $w\#x$, which it should not. ■

This very clearly shows that the lower bound, $O(\lg n)$, on the amount of space required to recognize $\{a^n b^n | n \geq 1\}$ is purely due to space constructibility reasons, whereas $\{w\#w | w \in (a + b)^*\}$ requires $O(\lg n)$ space independent of any space constructibility properties of the standard Turing machine model.

6 Pebble Machines

In the previous section we showed the importance of constructibility of space bounds for machines having less than $\lg n$ space. We know that, if $\sup_{n \rightarrow \infty} S(n)/\lg n = 0$, then $S(n)$ can not be constructed fully by deterministic Turing machines. In this section we introduce a very natural model, which can construct such functions. Similar models have been studied before [5].

Our Theorem 11 showed that deterministic $\lg \lg n$ machines do not gain any additional power from the use of the dot mechanism. This result is similar in spirit to the well-known result [5] that two-way finite automata do not gain any additional computing power from the use of a “pebble” which can be placed on the input tape as a movable marker. The situation however is quite different if we allow an additional 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 a pebble which can be placed on and removed from the input tape. The action of the Turing machine depends on the current state, currently scanned input and worktape symbols, and the presence of the pebble on the currently scanned input square and consists of changing the symbol on the worktape, moving the input and worktape heads and picking up or placing (or none) the pebble on the currently scanned input square 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 $PEBBLESPACE[S(n)] = DSPACE[S(n)]$. We now establish

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

Proof: We show that $\{a^n b^n \mid n \geq 1\} \in PEBBLESPACE[S(n)]$. To see this, observe that $\lg \lg n$ is constructible by the pebble machines. To do this M on 1^n places the pebble at position k and finds out the first prime which does not divide k . It repeats this procedure for $k = 1, 2, \dots, n$. Then, by the proof of Theorem 11, M uses $O(\lg \lg n)$ space and no more. Then clearly $\{a^n b^n \mid n \geq 1\} \in PEBBLESPACE[\lg \lg n]$. ■

Again, using Sipser's method, Richard Chang has shown that

$$PEBBLESPACE[S(n)] = coPEBBLESPACE[S(n)].$$

7 Open Problems

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

1. Is $\lg \lg n$ (or any $S(n)$ with $\sup_{n \rightarrow \infty} S(n)/\lg n = 0$) 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 [14] 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 relations between the various deterministic and nondeterministic space, dot, demon and pebble machine defined complexity classes ?
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 and not by deterministic Turing machines. This was achieved by exploiting the $n - n - n!$ method. This lead to the separation of $SPACE^*[\lg \lg n]$ and $NSPACE^*[\lg \lg n]$. We also showed that Savitch and Immerman-Szelepcsényi theorems do not hold for this computation model in the low complexity range. We also discussed other useful models for space classes below $DSPACE[\lg n]$ 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 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 Richard Chang, Ken Regan, Wei Li and Alessandro Panconesi for active participation in interesting discussions and providing several bits of oracular advice. The second author would like to thank the students of the Spring '89 class of CS782 at Cornell University for listening

to some of the ideas presented in this paper and providing constructive criticism about them.

References

- [1] A.R. Freedman and R.E. Ladner. Space bounds for processing counterless inputs. *Journal of Computer and System Sciences*, 11:118–128, 1975.
- [2] R. Freivalds. On the worktime of deterministic and non-deterministic turing machines. *Latvijskij Matematiseskij Eshgodnik*, 23:158–165, 1979.
- [3] J. Hartmanis, R. Chang, J. Kadin, and S. Mitchell. Some observations about space bounded computations. *Bulletin of the EATCS*, 35:82–92, June 1988.
- [4] J. Hartmanis and H.H Hunt. On the *LBA* problem and its importance in the theory of computation. *SIAM-AMS*, 7:1–26, 1974.
- [5] J.E. Hopcroft and J.D. Ullman. *Introduction to Automats Theory, Languages and Computation*. Addison-Wesley Publishing Company, 1979.
- [6] 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.
- [7] S.Y. Kuroda. Classes of languages and linearly-bounded automata. *Information and Control*, 7:207–223, 1964.
- [8] 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.
- [9] W.J. Savitch. Relationships between nondeterministic and deterministic tape complexities. *Journal of Computer and System Sciences*, 4:177–192, 1970.

- [10] Seiferas. A note on notions of tape constructibility. Technical Report CSD-TR 187, Pennsylvania State University, 1976.
- [11] M. Sipser. Halting space-bounded computations. *Theoretical Computer Science*, 10:335–338, 1980.
- [12] 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.
- [13] R. Szelepcsényi. The method of forcing for nondeterministic automata. *The Bulletin of the European Association for Theoretical Computer Science*, 33:96–100, October 1987.
- [14] 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.
- [15] A. Szepietowski. If deterministic and nondeterministic space complexity are equal for $\log \log n$ then they are equal for $\log n$. In *Lecture Notes in Computer Science*, volume 349, pages 251–255. Springer-Verlag, 1989. STACS '89.
- [16] C.B. Wilson. Relativized circuit complexity. *Journal of Computer and System Sciences*, 31:169–181, 1985.
- [17] C.B. Wilson. Parallel computation and the NC hierarchy relativized. In *Lecture Notes in Computer Science*, volume 223, pages 362–382. Springer-Verlag, 1986. Structure in Complexity Theory.