JOURNAL OF COMPUTER AND SYSTEM SCIENCES **28**, 216–230 (1984)

Space-Bounded Hierarchies and Probabilistic Computations*

WALTER L. RUZZO

Department of Computer Science, FR–35,
University of Washington, Seattle, Washington 98195

JANOS SIMON

Department of Computer Science,
The Pennsylvania State University, University Park, Pennsylvania 16802

AND

MARTIN TOMPA

Department of Computer Science, FR–35,
University of Washington, Seattle, Washington 98195

Received September 13, 1982

We study three aspects of the power of space-bounded probabilistic Turing machines. First, we give a simple alternative proof of Simon's result that space-bounded probabilistic complexity classes are closed under complement. Second, we demonstrate that any language recognizable by an alternating Turing machine in $\log n$ space with a constant number of alternations (the $\log n$ space "alternation hierarchy") also can be recognized by a $\log n$ space-bounded probabilistic Turing machine with small error probability; this is a generalization of Gill's result that any language in NSPACE ($\log n$) can be recognized by such a machine. Third, we give a new definition of space-bounded oracle machines, and use it to define a space-bounded "oracle hierarchy" analogous to the original definition of the polynomial time hierarchy. Unlike its polynomial time analogue, the entire $\log n$ space "alternation hierarchy" is contained in the second level of the $\log n$ space "oracle hierarchy." However, the entire $\log n$ space "oracle hierarchy" is still contained in bounded-error probabilistic space $\log n$.

1. INTRODUCTION

This paper studies the power of space-bounded probabilistic Turing machines. Section 2 presents a simple alternative proof of Simon's result [19] that space-bounded probabilistic complexity classes are closed under complement. Section 3 demonstrates that any language in the $\log n$ space hierarchy can be recognized by a

* This material is based upon work supported by the National Science Foundation under Grants ECS–8007428, MCS–7905006, MCS–8104876, MCS–8003337, and MCS–8110089.

0022-0000/84 $3.00

$\log n$ space-bounded probabilistic Turing machine with small error; this is a generalization of Gill's result that any language in NSPACE($\log n$) can be recognized by such a machine [7, Proposition 6.8].

The second result raises interesting questions about space hierarchies, which are considered in section 4. The usual definition is in terms of space-bounded alternating Turing machines with a constant number of alternations [6]. We show that an alternative definition in terms of space-bounded oracle machines (analogous to the original definition of the polynomial time hierarchy [21]) has the following interesting properties:

1. Unlike its polynomial time analogue, the entire $\log n$ space "alternation hierarchy" is contained in the second level of the $\log n$ space "oracle hierarchy."

2. The entire $\log n$ space "oracle hierarchy" is nonetheless still contained in bounded-error probabilistic space $\log n$.

This new definition of space-bounded oracle machines is of interest in its own right; there does not seem to be a generally accepted definition in the literature. Ours appears to be a reasonable compromise between two previously proposed definitions [10, 17], and is somewhat better behaved than either.

There are enough subtle differences among various proposed definitions of probabilistic computations that it is worthwhile discussing them here. A *probabilistic Turing machine* is a multitape Turing machine that at each step can choose its next state equiprobably from two states. If M is a probabilistic Turing machine and x an input, let $p_M(x)$ denote the probability that M reaches an accepting configuration on input x. By the most general definition, the language L accepted by M is exactly the set of inputs x such that $p_M(x) > \frac{1}{2}$. Such machines might be said to exhibit *2-sided error*, since the probability of an erroneous answer in a given trial may be greater than zero both for inputs in L and for inputs not in L. Computations with 2-sided error can be further classified according to whether or not $p_M(x)$ is *bounded* away from $\frac{1}{2}$ by a constant for all x. An algorithm is said to exhibit *1-sided error* if $p_M(x) = 0$ for all $x \notin L$, and *0-sided error* if in addition $p_M(x) = 1$ for all $x \in L$. The *error probability* of M is the probability that a given trial of M has the wrong outcome, i.e., the least upper bound of $(\{p_M(x) \mid x \notin L\} \cup \{1 - p_M(x) \mid x \in L\})$. A final consideration when defining probabilistic complexity classes is whether worst-case or expected resource usage is measured. As an example for readers familiar with the notation of Gill [7], the polynomial time-bounded complexity classes arising from these definitions are PP (2-sided error, worst-case time), BPP (bounded 2-sided error, worst-case time), VPP (1-sided error, worst-case time), and ZPP (0-sided error, expected time). The class VPP is sometimes called "random P," RP, or R.

All complexity bounds in this paper are assumed to be worst-case unless explicitly stated otherwise.

Previously known results about the power of space-bounded probabilistic Turing machines include the following: Gill [7, Proposition 6.8] showed that every language recognizable in nondeterministic space $S(n)$ is also recognizable in one-sided error

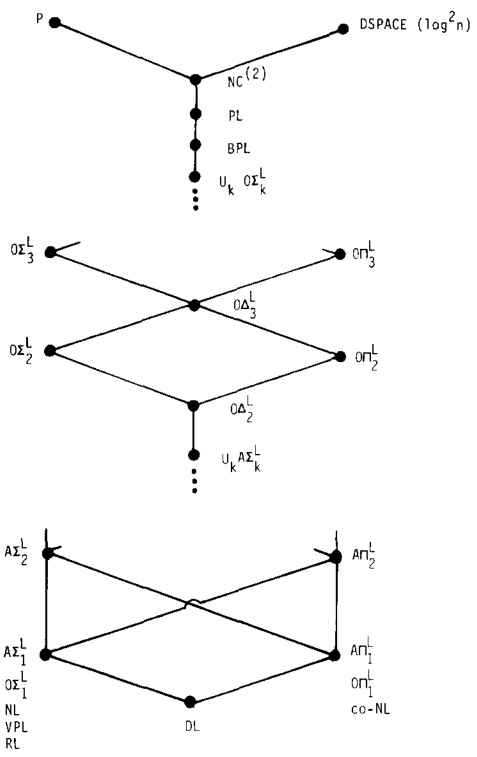

FIG. 1. Some space-bounded complexity classes.

probabilistic space $S(n)$. (In fact, these two classes are equal.) Further, languages probabilistically recognizable under the most general definition (unrestricted 2-sided error) in space $S(n)$ are also recognizable in alternating time $S^2(n)$, hence deterministic space $S^2(n)$. This was recently shown by Borodin *et al.* [5], improving an earlier result of Simon [18]. Similar results have been shown for probabilistic transducers [8, 5]. Figure 1 summarizes these and other complexity relationships discussed in this paper.

The model of space-bounded probabilistic Turing machine considered in these results, and throughout this paper, places no restriction on the time used by the machine. In fact, the time used may be as large as double exponential in the space bound. Other authors have considered probabilistic Turing machines whose time bound may not exceed a single exponential in its space bound [2, 14].

2. Space-Bounded Computations That Halt with Probability 1

J. Simon [19] proved that probabilistic 2-sided error space-bounded complexity classes are closed under complement. This result follows easily from his main lemma:

THEOREM 1. *Let $S(n)$ be space constructible, and let M be an $S(n)$ space-bounded probabilistic Turing machine with 2-sided error (bounded 2-sided error, 1-sided error). Then there is an $S(n)$ space-bounded probabilistic Turing machine M' with 2-sided error (respectively, bounded 2-sided error, 1-sided error) that accepts the same language as M, and on all inputs halts with probability 1, doing so in expected time $2^{2^{O(S(n))}}$.*

Simon's proof of this lemma is quite involved, and the purpose of this section is to provide a simple alternative proof. Notice that it is not sufficient to attach a "clock" to M as might be done for other varieties of machines, as it is possible for $S(n)$ space-bounded probabilistic Turing machines to run usefully for $2^{2^{\theta(S(n))}}$ steps [7, Propositions 6.7 and 6.8].

Both Simon's proof and the one presented here depend on a fundamental result of Gill concerning the possible values of the probability of acceptance $p_M(x)$ in a space-bounded machine.

LEMMA 2 (Gill [7, Lemma 6.6]). *Let M be an $S(n)$ space-bounded probabilistic Turing machine. Then (even in the absence of any assumption about bounded error) there is a constant c such that for every input x, if $p_M(x) > \frac{1}{2}$ then*

$$p_M(x) > \tfrac{1}{2} + 2^{-c^{S(|x|)}}.$$

We now outline the proof of Theorem 1. First we consider the general case of unrestricted 2-sided error. Let M be an $S(n)$ space-bounded probabilistic Turing machine. Let c be the constant given in Lemma 2, and let d be a constant such that the number of configurations of M on inputs of length n is at most $d^{S(n)}$. M' will be constructed to simulate M for $\mathbf{t}$ steps, where $\mathbf{t}$ is a random variable whose value is $2^{2^{\theta(S(n))}}$ with overwhelming probability, and then to accept if and only if M has accepted within $\mathbf{t}$ steps. The aim in so choosing $\mathbf{t}$ is to render the probability of M accepting after $\mathbf{t}$ steps negligible, that is, less than the bound $2^{-c^{S(n)}}$ given in Lemma 2.

Specifically, on input x of length n, M' does the following:

```
repeat forever
    1. Simulate $M$ on $x$ for $d^{S(n)}$ further steps. If $M$ enters an accepting configuration at any step, halt and accept.
    2. Toss $(c+d)^{S(n)}$ coins. If all come up tails, halt and reject.
end.
```

It is easy to see that M' uses $O(S(n))$ space, halts with probability 1, runs in expected time $2^{2^{O(S(n))}}$, and does not accept any input that M does not accept. It remains only to show that M' does not reject any input that M accepts; that is, the probability that M' rejects an input x in step 2 of the loop that M would accept later in the simulation is at most $2^{-c^{S(n)}}$, so that

$$p_{M'}(x) \geqslant p_M(x) - 2^{-c^{s(n)}} > \tfrac{1}{2}.$$

The probability that M' rejects x in step 2 of the kth iteration of the loop (assuming the simulation in step 1 proceeds this far) is

$$(1 - 2^{-(c+d)^{S(n)}})^{k-1} \cdot 2^{-(c+d)^{S(n)}}.$$

The probability that M accepts x after the kth iteration of the loop (assuming the coin-tossing in step 2 proceeds this far) is at most $(1 - 2^{-d^{S(n)}})^k$: Since M eventually accepts x, whenever M' begins step 1 M is in a configuration from which an accepting configuration is reachable, hence reachable within $d^{S(n)}$ steps, and hence reachable with probability at least $2^{-d^{S(n)}}$. Therefore the probability that M fails to accept x for k iterations (but does accept x eventually) is at most $(1 - 2^{-d^{S(n)}})^k$.

The probability that M' rejects x prematurely (that is, M would have accepted x during some later iteration) is then at most

$$\begin{aligned}
&\sum_{k=1}^{\infty} (1 - 2^{-d^{S(n)}})^k \cdot (1 - 2^{-(c+d)^{S(n)}})^{k-1} \cdot 2^{-(c+d)^{S(n)}} \\
&\quad \leqslant 2^{-(c+d)^{S(n)}} \sum_{k=1}^{\infty} (1 - 2^{-d^{S(n)}})^k \\
&\quad \leqslant 2^{-(c+d)^{S(n)}} \cdot 2^{d^{S(n)}} \\
&\quad \leqslant 2^{-c^{S(n)}},
\end{aligned}$$

as was claimed.

Finally, note that M' operates with 1-sided error if M does. If M operates with bounded 2-sided error, i.e., $p_M(x)$ is bounded away from $\frac{1}{2}$ by a constant for all x, then so is $p_M(x) - 2^{-c^{S(n)}}$ for some sufficiently large c, so M' also operates with bounded error. This completes the proof. ▌

Closure under complementation follows easily from this:

THEOREM 3 (Simon [19]). *If L is accepted by an $S(n)$ space-bounded probabilistic Turing machine with 2-sided error (or bounded 2-sided error), then so is the complement of L.*

Proof. Let M be the machine accepting L, and let $\bar{M}$ be the same as M except that accepting and rejecting configurations have been reversed. For $\bar{M}$ to accept the complement $\bar{L}$ of L, it suffices that the following two conditions be satisfied.

1. M halts with probability 1. This ensures that $p_{\bar{M}}(x) = 1 - p_M(x)$.

2. $p_M(x) \neq \frac{1}{2}$ for all x. This, together with the previous condition, ensures that $p_{\bar{M}}(x) > \frac{1}{2}$ iff $p_M(x) \leqslant \frac{1}{2}$, hence $\bar{M}$ accepts $\bar{L}$.

By Theorem 1, we may assume that M satisfies condition 1. We may also assume that condition 2 is satisfied by using the following technique of Gill [7, Proposition 5.3]. Let c be the constant from Lemma 2 for M. Let M' be a probabilistic Turing machine that always halts and accepts each input with probability exactly $(\frac{1}{2}) - 2^{-c^{S(n)}}$. Let M'' run M or M' with equal probability. By Lemma 2, $p_M(x) > \frac{1}{2}$ implies $p_M(x) > (\frac{1}{2}) + 2^{-c^{S(n)}}$, so $p_{M''}(x) > \frac{1}{2}$. Further, $p_M(x) \leqslant \frac{1}{2}$ implies $p_{M''}(x) < \frac{1}{2}$. Thus M'' accepts the same language as M, and $p_{M''}(x) \neq \frac{1}{2}$ for all x. ▮

3. The log n Space Alternation Hierarchy Is in Probabilistic Space log n

Gill has shown that any language in NSPACE(log n) can be recognied by a log n space-bounded probabilistic Turing machine with bounded error probability [7, Proposition 6.8]. One corollary of Theorem 3 is that co–NSPACE(log n) is also contained in this class. It is natural to ask whether more of the log n space hierarchy [6] (which is analogous to the polynomial time hierarchy [21]) can be recognized by such machines. In this section and the next we provide a strong positive answer to this question by showing that any language in this entire hierarchy (and in fact in a hierarchy that contains this one) can be recognized by such probabilistic machines.

One cost of our extension of Gill's simulation is that our construction uses the bounded 2-sided error model, whereas a 1-sided error algorithm suffices for Gill's construction. Indeed, one of the interesting aspects of our algorithm is that it is one of two natural examples known to us of a probabilistic algorithm with bounded, 2-sided error. The other was discovered by Reif [14]. His is an algorithm for a somewhat similar problem, namely, simulating the symmetric space hierarchy. Symmetric space, introduced by Lewis and Papadimitriou [11], is a restriction of nondeterministic space for which there is a particularly efficient probabilistic simulation using the technique of Aleliunas *et al.* [2]. Reif's work generalizes the Lewis and Papadimitriou simulation of symmetric space in a way that is analogous to this section's generalization of Gill's simulation of nondeterministic space.

Another interesting aspect of our result is that it provides evidence that algorithms with 2-sided error are more powerful than those with 1-sided error. Languages recognized in space log n by probabilistic machines with 1-sided error are precisely the languages in NSPACE(log n) [7, Proposition 6.8]. Thus, if our simulation of the log n space hierarchy, or any fixed level of it, could be improved to 1-sided error, then the hierarchy would collapse to NSPACE(log n). Similar statements apply to the oracle hierarchy discussed in Section 4.

We now turn to the proof of our results. In this section and the next, the discussion will be confined to space bounds that are $O(\log n)$. At the end of this section and in Section 5, we discuss generalizations of the results to arbitrary space bounds. Before beginning, it is convenient to establish some notation:

DL = DSPACE(log n)

NL = NSPACE(log n)

BPL = $\{B \mid$ the language B is accepted within log n space by a bounded 2-sided error probabilistic Turing machine$\}$

$\mathrm{A}\sum_k^L = \{B \mid$ the language B is accepted by a $\log n$ space-bounded alternating Turing machine starting in an existential state and making exactly $k-1$ alternations between existential and universal states$\}$

$\mathrm{A}\prod_k^L = \text{co-A}\sum_k^L$

$\text{co-}X = \{B \mid \bar{B} \in X\}$, for any class X of languages.

Notice for example that $\mathrm{A}\sum_1^L = \mathrm{NL}$. We use the notation $\mathrm{A}\sum$ [6] rather than the more customary $\sum$ to emphasize that this is the "alternation" hierarchy, as opposed to the "oracle" hierarchy $\mathrm{O}\sum$ to be introduced in the next section.

In this section we will adopt Ladner and Lynch's notion of space-bounded oracle machines [10]: the query tape of such a machine is write-only, is not subject to the machine's space bound, is erased after each oracle call, and the oracle machine may only run for time exponential in its space bound. If M and A are space-bounded complexity classes, let M^A represent the class of languages recognizable by machines of class M using oracles from class A.

The main result promised for this section is Theorem 4 below. Both it and Theorem 6 will be subsumed by more general results in the next section, but they are included here since they are interesting in their own right, and since they serve to motivate the key definitions and proofs needed for the generalization.

THEOREM 4. $\bigcup_k \mathrm{A}\sum_k^L \subseteq \mathrm{BPL}$.

This follows from the next two theorems. Notice that the first is in sharp contrast to the situation for the polynomial time hierarchy, where the analogous result $(\bigcup_k \sum_k^P \subseteq \Delta_2^P)$ would be very surprising.

THEOREM 5. $\bigcup_k \mathrm{A}\sum_k^L \subseteq \mathrm{DL}^{\mathrm{NL}}$.

Proof. The proof is similar to [6, Theorem 4.2], attributed there to Borodin. Let L be recognized by a $c\log_2 n$ space- and k alternation-bounded alternating Turing machine M. Without loss of generality, assume M doesn't loop within $c\log_2 n$ space. Let $\mathrm{REACH}(x, P, Q)$ be true if and only if M on input x can reach configuration Q from configuration P by a computation whose last move, and only the last move, is an alternation. (Entering an accepting configuration is considered an alternation for the purposes of this proof.) Note that REACH is in NL. Then the following procedure will return true if and only if P is the root of an accepting subtree using space $\leqslant c\log_2 n$, and with at most k alternations.

```
ACC($P$, $k$):
    if $P$ uses space $= c\log_2 n + 1$ then return false;
    if $P$ is an accepting configuration then return true iff $k \geqslant 0$;
    if $P$ is existential then return $\bigvee_Q$ ($\mathrm{REACH}(x, P, Q) \wedge \mathrm{ACC}(Q, k-1)$);
    if $P$ is universal then return $\bigwedge_Q$ ($\mathrm{REACH}(x, P, Q) \Rightarrow \mathrm{ACC}(Q, k-1)$).
```

The required set of all configurations Q using space $\leqslant c \log_2 n + 1$ can be enumerated deterministically in $O(\log n)$ space. REACH is tested by invoking the oracle, and the recursive calls on ACC are tested by maintaining a stack of k configurations. ∎

THEOREM 6. $\mathrm{DL}^{\mathrm{NL}} \subseteq \mathrm{BPL}$.

Proof. The BPL machine will simulate the DL machine deterministically between oracle calls. Whenever an oracle call is made, we use Gill's simulation of NL by BPL [7, Proposition 6.8] to answer the oracle query. (In order to avoid the problem that the query tape may exceed the space bound, we use the standard technique of reconstructing its symbols as needed by the oracle.) Since Gill's simulation is by a 1-sided error algorithm, we can repeat it $\log_2(4t(n))$ times to reduce its error probability to $\varepsilon \leqslant 1/(4t(n))$, where $t(n)$ is the running time of the DL machine. Then the probability that the BPL machine gives the correct answer is at least the probability that all the oracle calls are correctly answered, which is

$$\geqslant (1-\varepsilon)^{t(n)} \geqslant 1 - t(n)\,\varepsilon \geqslant 1 - \tfrac{1}{4} = \tfrac{3}{4} > \tfrac{1}{2}. \quad ∎$$

Theorem 4 generalizes from $\log n$ space to an arbitrary constructible space bound $S(n) = \Omega(\log n)$ as follows. Let

$\mathrm{BPSPACE}(S(n)) = \{B \mid$ the language B is accepted within space $S(n)$ by a bounded 2-sided error probabilistic Turing machine$\}$

$\mathrm{A}\sum_k^{S(n)} = \{B \mid$ the language B is accepted by an $S(n)$ space-bounded alternating Turing machine starting in an existential state and making exactly $k-1$ alternations between existential and universal states$\}$.

Then $\bigcup_k \mathrm{A}\sum_k^{S(n)} \subseteq (\mathrm{DSPACE}(S(n)))^{\mathrm{NL}}$ as in Theorem 5, provided the oracle machine pads each query with $2^{S(n)}$ padding symbols, and $(\mathrm{DSPACE}(S(n)))^{\mathrm{NL}} \subseteq \mathrm{BPSPACE}(S(n))$ as in Theorem 6.

4. THE LOG n SPACE ORACLE HIERARCHY IS IN PROBABILISTIC SPACE LOG n

Theorem 5 suggests an interesting problem: How should one define a hierarchy of relativized space-bounded complexity classes, analogous to the oracle definition of the polynomial time hierarchy [21], with $\mathrm{DL}^{\mathrm{NL}}$ as "Δ_2"? The problem with doing so is that the notion of relativized space-bounded computation, or equivalently of space-bounded Turing reducibility, seems to be much more delicate than the time-bounded version. There is no generally accepted definition in the literature. Ladner and Lynch [10] first exposed some of the delicacy. For example, they showed that the simple relationship $\mathrm{NL} \subseteq \mathrm{P}$ may be false in the relativized setting for the natural definition of relativized computation given in the previous section. (See also Savitch [16].) Similarly unexpected results have since been demonstrated for other models and other definitions of relativized computation [3, 17]. One of our contributions is a new notion of space-bounded relativized computation that seems to be somewhat better behaved than those considered in the past. We will use this to refine our understanding of probabilistic space-bounded computation. In particular, in this section we will define a space-bounded oracle hierarchy that contains the entire space-bounded alternation hierarchy within its second level, and is in turn contained in BPL.

The next example illustrates one unexpected fact about oracle machines as defined in the previous section, and will help motivate our definition.

EXAMPLE 1. The natural next step after $\mathrm{DL}^{\mathrm{NL}}$ in the proposed oracle hierarchy would be the $\sum_2$-analog $\mathrm{NL}^{\mathrm{NL}}$. However, $\mathrm{NL}^{\mathrm{NL}} = \mathrm{NL}^{\mathrm{DL}} = \mathrm{NP}$: An $\mathrm{NL}^{\mathrm{DL}}$ machine can recognize satisfiable formulas by copying its input formula onto its query tape, guessing and writing a truth-assignment onto the query tape, then calling an oracle for the formula value problem, a problem known to be in DL [12]. The converse is straightforward. ∎

Obviously a hierarchy defined in this way won't be of much interest, since it will essentially duplicate the polynomial time hierarchy. The problem with the definition seems to be that the nondeterminism and the long query tape have interacted to give the machine the effect of a nondeterministic write-once/read-many-times tape of polynomial length, greatly increasing the machine's power. We avoid this by insisting that the query tape be written *deterministically*.

DEFINITION. Let $M^{\langle A\rangle}$ denote the language recognized by machine M using oracle A in the following controlled manner: M has a write-only query tape not subject to a space bound, and operates *deterministically* from the time some symbol is written onto the query tape until the time the next oracle call is made, after which the query tape is erased. Similarly, for complexity classes M and/or A, let $M^{\langle A\rangle}$ denote the class of languages recognized by machines of class M using some oracle from class A in this controlled manner. Also let

$$\begin{aligned} \mathrm{O}\textstyle\sum_0^L &= \mathrm{DL} \\ \mathrm{O}\textstyle\sum_{k+1}^L &= \mathrm{NL}^{\langle \mathrm{O}\Sigma_k^L\rangle} \\ \mathrm{O}\Delta_{k+1}^L &= \mathrm{DL}^{\langle \mathrm{O}\Sigma_k^L\rangle} \\ \mathrm{O}\textstyle\prod_k^L &= \text{co-}O\textstyle\sum_k^L. \end{aligned}$$

There is a useful alternative view of oracle computations $M^{\langle A\rangle}$: M can be placed in a "normal form" having a query tape that *is* subject to the bound, provided that the oracle also has access to M's input, and provided that the structure of the oracle set is slightly modified, technically by a deterministic space-bounded many-one reduction. The proof is simple: the short query tape encodes a configuration of M as it starts to write a long query, and the deterministic space-bound computation that M

uses to generate the query from that configuration is incorporated into the definition of the oracle set. This observation is made precise in Lemma 7 below.

Notice that relativized deterministic space is the same as Ladner and Lynch's definition [10]. Also, using the observation in the preceding paragraph, notice that the low levels of the oracle hierarchy are as desired, namely, $\mathrm{O}\sum_1^L = \mathrm{NL}$, and $\mathrm{O}\Delta_2^L = \mathrm{DL}^{\langle \mathrm{NL} \rangle} = \mathrm{DL}^{\mathrm{NL}}$ hence $\bigcup_k \mathrm{A}\sum_k^L \subseteq \mathrm{O}\Delta_2^L$.

Lemma 7. *For any L and A there exists a* $\log n$ *space-bounded oracle machine M such that $L = M^{\langle A \rangle}$ if and only if there exists a set B and a $\log n$ space-bounded oracle machine N of the same type as M (deterministic, nondeterministic, probabilistic) such that*

1. $L = N^{\langle B \rangle}$,

2. *all of N's oracle calls are for strings of the form $x\$y$ where x is N's input, and* $|y| \leqslant \log_2 |x|$, *and*

3. $B \leqslant_m^{\log} A$.

In less precise terms, Lemma 7 says that L is Turing reducible to A if and only if L is Turing reducible to some B using short queries and B is many-one reducible to A.

As one final motivation for our definition, note that the number of configurations reachable during the computation of an $S(n)$ space-bounded deterministic or nondeterministic Turing machine is $2^{O(S(n))}$. Under our definition, oracle machines share this behavior. This is also true of deterministic machines under Ladner and Lynch's definition [10], but not of nondeterministic ones. The example above shows that under their definition, nondeterministic machines may reach exponentially more configurations. This seems unconventional for a space-bounded machine, and probably helps explain the unexpected behavior of relativized nondeterministic classes elucidated in [10].

Oracle machines with space-bounded query tapes were among the models considered by Istvan Simon [17], but his definition did not allow access to the input tape as we have done. One of the drawbacks of his definition is that there are sets A for which $A \notin \mathrm{DL}^A$ (i.e., A is not log space Turing reducible to itself), since all queries must be much shorter than the input. For instance, if $A \in \mathrm{DSPACE}(n) - \mathrm{DL}$, then $\mathrm{DL}^A \subseteq \mathrm{DL}$, so $A \notin \mathrm{DL}^A$.

Our notion of relativized computation seems to be a well-behaved compromise between the previous definitions [10, 17]. For example, like [17, Theorem 4.1] but unlike [10, Theorem 3.2], Savitch's theorem relativizes: for any A,

$$\mathrm{NL}^{\langle A \rangle} \subseteq (\mathrm{DSPACE}(\log^2 n))^{\langle A \rangle}.$$

The promised generalizations of Theorem 4 and 6 are Theorems 8 and 10 below.

Theorem 8. $\bigcup_k \mathrm{O}\sum_k^L \subseteq \mathrm{BPL}$.

Proof. By induction on k using Theorems 9 and 10. ∎

THEOREM 9. $\mathrm{NL}^{\langle A\rangle} \subseteq \mathrm{BPL}^{\langle A\rangle}$.

Proof. Straightforward relativization of Gill's result that $\mathrm{NL} \subseteq \mathrm{BPL}$ [7, Proposition 6.8]. ∎

THEOREM 10. $\mathrm{BPL}^{\langle \mathrm{BPL}\rangle} \subseteq \mathrm{BPL}$.

Proof. We wish to simulate some $M^{\langle A\rangle}$. Without loss of generality it suffices to simulate some BPL oracle machine $N^{\langle B\rangle}$ satisfying the following conditions. First, by Lemma 7, N's query tape can be taken to be of length $\log n$, as are all the other work tapes, with $B \in \mathrm{BPL}$ derived from A appropriately. Second, by a relativized version of Theorem 1, assume that $N^{\langle B\rangle}$ has expected time $T(n) = 2^{2^{O(\log n)}}$ on all inputs. Third, by a relativized version of Lemma 11 below, we can assume $N^{\langle B\rangle}$ has error probability $\varepsilon < \frac{1}{8}$.

Because $N^{\langle B\rangle}$ runs in expected time $T(n)$, the probability that it runs much longer than $T(n)$ is small. Specifically, let

$$p = \Pr\{N^{\langle B\rangle} \text{ halts with the correct answer within } 8T(n) \text{ steps}\}.$$

Then

$$\begin{aligned} p &\geqslant (1-\varepsilon) - \Pr\{N^{\langle B\rangle} \text{ runs for more than } 8T(n) \text{ steps}\} \\ &> \tfrac{7}{8} - \tfrac{1}{8} = \tfrac{3}{4}. \end{aligned}$$

The simulation is similar to that in Theorem 6: simulate the probabilistic machine N directly, pausing to simulate some probabilistic recognizer $\tilde{B}$ for B as needed. The key issue is to show that the error probability arising from the many calls to $\tilde{B}$ isn't too great. The point of the analysis in the preceding paragraph is to bound the number of calls: $N^{\langle B\rangle}$ with high probability halts correctly within $8T(n)$ steps. Thus, if we can with high probability correctly answer $8T(n)$ consecutive oracles queries with our (unreliable) recognizer $\tilde{B}$, then with high probability we will have faithfully simulated a correct computation of $N^{\langle B\rangle}$.

There is a $\log n$ space-bounded recognizer $\tilde{B}$ for B that, by Lemma 11, has error probability $\delta < 1/(32T(n))$, and that, by Theorem 1, halts with probability 1. The probability that the simulation of $N^{\langle B\rangle}$ is correct is at least the probability that $\tilde{B}$ correctly answers $8T(n)$ consecutive queries, and that a correct path of $N^{\langle B\rangle}$ of length at most $8T(n)$ is chosen. Thus, the probability of a correct simulation is

$$\begin{aligned} &\geqslant (1-\delta)^{8T(n)}\, p \\ &\geqslant (1 - 8T(n)\,\delta)\, p \\ &> (1-\tfrac{1}{4})(\tfrac{3}{4}) = \tfrac{9}{16} > \tfrac{1}{2}. \end{aligned}$$ ∎

LEMMA 11. *Let L be accepted in space $S(n)$ by a probabilistic Turing machine M with bounded error. Then for any $\varepsilon(n) \geqslant 2^{-2^{O(S(n))}}$, L is accepted by some $S(n)$ space-bounded machine with error probability at most $\varepsilon(n)$.*

Proof. If M exhibited 1-sided error, it would suffice to run M for $2^{O(S(n))}$ trials (more precisely, $\log_2(1/\varepsilon(n))$ trials), accepting if and only if M accepted in at least one trial. For the more general case of bounded 2-sided error, it still suffices to run M for $2^{O(S(n))}$ trials (more precisely, $O(\log(1/\varepsilon(n)))$ trials), accepting iff M accepts in the *majority* of the trials, as shown below. Let M accept with probability $p < \frac{1}{2}$ and let $q = 1 - p$. The probability that M accepts in the majority of t trials is

$$\begin{aligned} &\sum_{i=t/2}^{t} \binom{t}{i} p^i q^{t-i} \\ &\quad \leqslant (4pq)^{t/2} \\ &\quad \leqslant \varepsilon(n), \qquad \text{when} \quad t \geqslant 2 \log_2(\varepsilon(n))/\log_2(4pq) = \Theta(\log(1/\varepsilon(n))). \end{aligned}$$

The first inequality is a special case of the Chernoff inequality [13, p. 246]. Also note $\log_2(4pq) < 0$. ∎

Notice that Theorem 8 coupled with [5] shows that any language in the $\log n$ space oracle hierarchy is in $NC^{(2)}$, i.e., can be recognized by a uniform circuit of polynomial size and $O(\log^2 n)$ depth, and hence is in $P \cap \mathrm{DSPACE}(\log^2 n)$, and by [15] is also in $A \sum^{L}_{O(\log^2 n)}$.

5. Comments

Figure 1 summarizes known relationships among many of the space-bounded complexity classes discussed in this paper. Figure 2 summarizes the analogous results for polynomial time-bounded classes. Although there are often strong parallels between logarithmic space- and polynomial time-bounded classes, there are two striking differences here.

First, adding probabilism to a space-bounded machine seems to increase its power much more than it does in the time-bounded case. For instance, 1-sided error polynomial time-bounded languages (variously denoted R, RP, or VPP) form a subset of NP. Evidence in [1, 9] strongly suggests that it is a proper subset. Similar evidence suggests that bounded 2-sided error polynomial time-bounded languages (BPP) do not include NP [4], but they are known to be in the polynomial hierarchy [20]. In contrast, the 1-sided error logarithmic space-bounded languages (VPL) are exactly equal to NL [7], and bounded 2-sided error logarithmic space-bounded languages (BPL) contain not only NL but also the entire log-space hierarchy. What causes this difference in relative power? The only intuitive motivation we can suggest is that space-bounded probabilistic machines can run for exponentially longer than deterministic or nondeterministic machines having the same space bound. Both Gill's result that VPL = NL [7] and our results make central use of this ability. No analogous difference between the machine types is evident in the time-bounded case.

The second major difference in the behavior of the time versus space-bounded complexity classes is the (presumed) separation of the alternation and oracle

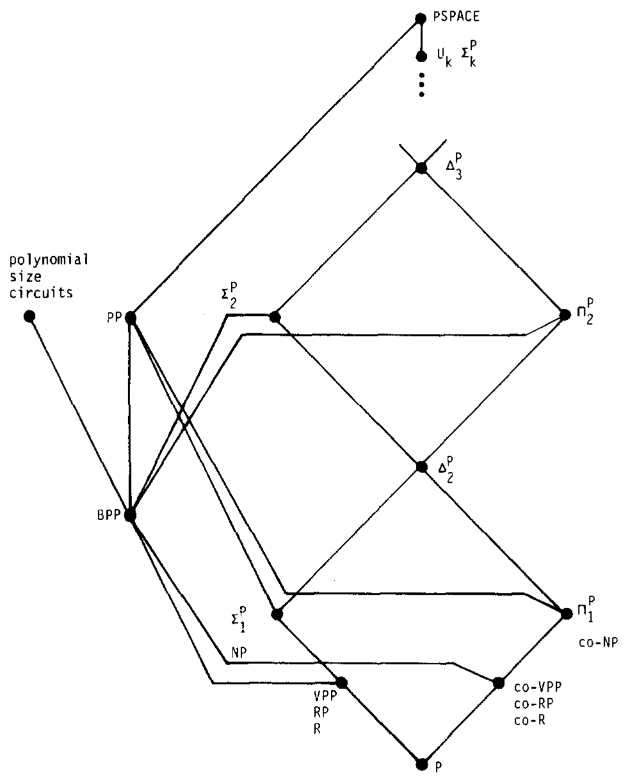

FIG. 2. Some time-bounded complexity classes.

hierarchies in the space-bounded case. As motivation for this result, consider trying to adapt the standard proof [6] that O $\sum_k^P \subseteq$ A $\sum_k^P$ to O $\sum_k^L$ vs. A $\sum_k^L$. The alternating machine simulates the nondeterministic oracle machine directly, except for queries. Queries to which it guesses the answer is "yes" may be checked directly, since by induction they are recognizable by A $\sum_{k-1}^P$ machines. However, queries to which it guesses the answer is "no" must be saved until the end, when all can be checked at once with one alternation to a universal state, since again by induction they are recognizable by co–A $\sum_{k-1}^P$ machines. Saving these queries may take nearly as much space as the machine's time bound. This is no problem for an A $\sum_k^P$ machine, but would destroy the space bound of an A $\sum_k^L$ machine. Hence the proof fails for logarithmic space-bounded machines. Note that any proof must fail, unless the hierarchies collapse (Theorem 5).

Finally, we will give a few remarks about generalizing these results to other space bounds. At the end of Section 3 we noted how Theorem 4 could be generalized to arbitrary space bounds. Generalizing Theorems 8 and 10 is more difficult, and again the difficulty stems from subtleties in the definition of relativized space.

EXAMPLE 2. By our definition or Ladner and Lynch's,

$$\mathrm{DSPACE}(n)^{\mathrm{DSPACE}(n)} = \bigcup_c \mathrm{DSPACE}(c^n)$$

(hence, is not in DSPACE(n) or BPSPACE(n), for example). This is proved by the following construction. The oracle machine on input x writes x followed by c^n padding characters; the query responder then can answer in $O(n')$ space, $n' = n + c^n$. Thus $\bigcup_c \mathrm{DSPACE}(c^n) \subseteq \mathrm{DSPACE}(n)^{\mathrm{DSPACE}(n)}$. The converse is straightforward. ∎

This behavior is quite different from that of other hierarchies. For example, for time-bounded classes, $\mathrm{P}^{\mathrm{P}} = \mathrm{P}$. This seems consistent with the intuitive motivation for relativized computations: A^B shows what A machines can do given a subroutine which solves B problems. Thus for any suitably robust class like P, one would expect $A^A = A$. The example above clearly doesn't fit this pattern. The conclusion seems to be that it is a mistake to exclude a 1-way, write-only tape from the space bound, since the writer in some sense is able to "read" it via the oracle. The use of padding in the example above and the use of nondeterminism in Example 1 seem to be abuses of this space. Our definition from Section 4 eliminates the latter abuse. The use of padding could be eliminated by restricting the length of queries to be, say, polynomial in the length of the input. With this definition, Theorem 10 generalizes so that for $S(n) = O(\log^k n)$ for any k,

$$\mathrm{BPSPACE}(S(n))^{\langle \mathrm{BPSPACE}(S(n)) \rangle} \subseteq \mathrm{BPSPACE}(S(n)).$$

For larger space bounds some padding effects show up again, analogous to the situation for time where, for example, $\mathrm{DTIME}(n^2)^{\mathrm{DTIME}(n^2)} = \mathrm{DTIME}(n^4)$.

REFERENCES

1. L. ADLEMAN, Two theorems on random polynomial time, *in* "19th Annual Symposium on Foundations of Computer Science," October 1978, pp. 75–83.
2. R. ALELIUNAS, R. M. KARP, R. J. LIPTON, L. LOVÁSZ, AND C. RACKOFF, Random walks, universal traversal sequences, and the complexity of maze problems, *in* "20th Annual Symposium on Foundations of Computer Science," October 1979, pp. 218–223.
3. D. ANGLUIN, On relativizing auxiliary pushdown machines, *Math. Systems Theory* **13** (4) (1980), 283–299.
4. C. H. BENNETT AND J. GILL, Relative to a random oracle A, $\mathrm{P}^A \neq \mathrm{NP}^A \neq \text{co–}\mathrm{NP}^A$ with probability 1, *SIAM J. Comput.* **10** (1) (1981), 96–112.
5. A. BORODIN, S. COOK, AND N. PIPPENGER, Parallel computation for well-endowed rings and space-bounded probabilistic machines, Technical Report #162/83, University of Toronto, April 1983.
6. A. K. CHANDRA, D. C. KOZEN, AND L. J. STOCKMEYER, Alternation, *J. Assoc. Comput. Mach.* **28** (1) (January 1981), 114–133.
7. J. GILL, Computational complexity of probabilistic Turing machines, *SIAM J. Comput.* **6** (4) (December 1977), 675–695.
8. J. GILL, J. HUNT, AND J. SIMON, Deterministic simulation of tape-bounded probabilistic Turing machine transducers, *Theoret. Comput. Sci.* **12** (3) (1980), 333–338.

9. R. M. Karp and R. J. Lipton, Some connections between nonuniform and uniform complexity classes, *in* "Proceedings of the Twelfth Annual ACM Symposium on Theory of Computing," April 1980, pp. 302–309.
10. R. E. Ladner and N. A. Lynch, Relativization of questions about log space computability, *Math. Systems Theory* **10** (1) (1976), 19–32.
11. H. R. Lewis and C. H. Papadimitriou, Symmetric space-bounded computation, *Theoret. Comput. Sci.* **19** (1982), 161–187.
12. N. Lynch, Log space recognition and translation of parenthesis languages, *J. Assoc. Comput. Mach.* **24** (4) (October 1977), 583–590.
13. W. W. Peterson, "Error-Correcting Codes," MIT Press, Cambridge, Mass., and Wiley, New York, 1961.
14. J. H. Reif, Symmetric complementation, *in* "Proceedings of the Fourteenth Annual ACM Symposium on Theory of Computing," San Francisco, California, May 1982, pp. 201–214.
15. W. L. Ruzzo, On uniform circuit complexity, *J. Comput. System Sci.* **22** (3) (June 1981), 365–383.
16. W. J. Savitch, "A note on relativized log space," Technical Report CS-055, University of California, San Diego, March 1982.
17. I. Simon, "On some subrecursive reducibilities," Ph.D. thesis, Stanford University, April 1977. Computer Science Department Technical Report STAN-CS-77-608.
18. J. Simon, On the difference between one and many, *in* "Automata, Languages, and Programming," Lecture Notes in Computer Science, Vol. 52, pp. 480–491, Springer-Verlag, New York/Berlin, 1977.
19. J. Simon, Space-bounded probabilistic Turing machine complexity classes are closed under complement, *in* "Proceedings of the Thirteenth Annual ACM Symposium on Theory of Computing," Milwaukee, Wisconsin, May 1981, pp. 158–167.
20. M. Sipser, A complexity theoretic approach to randomness, *in* "Proceedings of the 15th Annual ACM Symposium on Theory of Computing," Boston, Massachusetts, May 1983, pp. 330–335.
21. L. J. Stockmeyer, The polynomial-time hierarchy, *Theoret. Comput. Sci.* **3** (1977), 1–22.