An NL Hierarchy

Jianer Chen[1]
Computer Science Department
Texas A&M University
College Station, TX 77843-3112

Jim Cox[2]
Department of Computer Science
Brooklyn College, CUNY
Brooklyn, N.Y. 11210

Bud Mishra[3]
Computer Science Department
New York University
New York, NY 10012

Abstract

We introduce a new nondeterministic oracle model. Our model is significantly different from all previous space-bounded oracle models, and has the following properties: First, it permits a non-trivial NL-hierarchy, which does not collapse without a similar consequence for the polynomial-time hierarchy. This is immediate from the result that Σ_k^p *is exactly the* $2k^{\mathrm{th}}$ *level in our NL-hierarchy, for all* $k \geq 1$. *On the other hand, it still behaves very well in relativized worlds. In particular, the new model satisfies the Immerman-Szelepcsenyi theorem and Savitch's theorem in the relativized worlds for oracle sets in each level of the polynomial-time hierarchy.*

Key Words: computational complexity, nondeterministic computation, query machines, space-bounded computation

[1] Supported in part by Engineering Excellence Award from Texas A&M University.
[2] Supported in part by PSC-CUNY 669287.
[3] Supported by NSF Grants #CDA-90-18673, #CCR-9002819 and ONR Grant #N00014-89-J3042.

1 Introduction

Immerman and Szelepcsenyi [3, 8] have independently proven that nondeterministic space is closed under complementation. As a consequence, several space bounded hierarchies have become finite. These results suggest that the space bounded computations may be easier to understand. Hartmanis [2] has issued a call to renew an attack on the space bounded computations and relativization of space bounded computations.

There are several options for the access mechanism to the oracle for space bounded computations. For deterministic and nondeterministic $\log n$ space bounded computations, we can allow both types of machines to use an additional one-way-write-only oracle tape on which they can write polynomial-length queries. (Call this model the "unrestricted model".) Denote the corresponding language classes by DL^A and NL^A, respectively. The problem for this model is that many important results in unrelativized worlds cannot be relativized. For example, using a straightforward Baker-Gill-Solovay oracle construction [1, 4], one can construct an oracle A such that

$$NL^A \not\subseteq P^A$$

Similarly, we can construct oracles B and C which for this model of relativization invalidate Savitch's result and the Immerman-Szelepcsenyi result, respectively

$$NL^B \not\subseteq (DL^2)^B \qquad \text{and} \qquad NL^C \neq \text{co-}NL^C$$

All these results suggest that the nondeterministic writing on the oracle tape is "too powerful" for relativized space bounded computations. To avoid these anomalies, another oracle access mechanism was introduced by Ruzzo, Simon and Tompa [6]. In this model, there is again an additional one-way-write-only oracle tape, but it is required that the computation of the oracle query is done deterministically by both types of machines. Now both types of machines can, for any given input, query only polynomially many strings in an oracle set A and the above anomalies are avoided. However, recent work on this model, most are based on "counting" techniques, and finally superseded by the Immerman-Szelepcsenyi result, has shown that this model is "too weak" to build up a non-trivial space bounded hierarchy [5, 9, 3, 8].

In this paper, we investigate the *inductive counting* technique, the main proof technique used in collapsing space bounded hierarchy, and the conditions under which the technique relativized. Specially, we investigate what restrictions need to be placed on a space-bounded oracle machine so that inductive counting holds in a relativized world. We introduce a new nondeterministic oracle model. (We will call this model the "restricted model" indicated by a subscript "r".) Based on our nondeterministic oracle model, we introduce a hierarchy of languages

$$NL_1 = NL, \quad NL_2 = NL_r^{NL}, \quad NL_3 = NL_r^{NL_2}, \quad \ldots$$

Our oracle model and the NL hierarchy have several interesting properties. First, the Immerman-Szelepcsenyi theorem and Savitch's theorem can be generalized to relativized worlds under our model for oracle sets in each level of the polynomial-time hierarchy:

$$\text{co-}NL_r^{\Sigma_k^p} = NL_r^{\Sigma_k^p} \qquad \text{and} \qquad NL_r^{\Sigma_k^p} \subseteq (DL^2)^{\Sigma_k^p}$$

for every integer $k \geq 1$, where $(DL^2)^{\Sigma_k^p}$ is the set of languages accepted by $O(\log^2 n)$ space-bounded deterministic oracle Turing machines relative to oracle sets in Σ_k^p. Secondly, our NL hierarchy is a refinement of the polynomial-time hierarchy:

$$NL_{2k} = \Sigma_k^p$$

for every integer $k \geq 1$, thus our NL hierarchy does not collapse without a similar consequence for the polynomial-time hierarchy. Thirdly, our oracle model differs from nondeterministic positive reducibility [7], as the model uses completely syntactic notions. Thus it can be easily checked if a particular machine satisfies the syntactic properties of the model. Hence it allows one to construct efficient universal machines, which is important for diagonalization proof techniques.

The result $NL_{2k} = \Sigma_k^p$ for each integer $k \geq 1$ is not just a simple observation of the trivial fact that $NL^{NL} = NP$ since we are also able to show that the similar result for the NL hierarchy based on the unrestricted oracle model would imply the collapse of the polynomial-time hierarchy.

2 A nondeterministic oracle model

An oracle Turing machine is a Turing machine that is equipped with a special one-way-write-only oracle tape, and a distinguished state q_Q, called the query state and two distinguished query answer states q_{yes} and q_{no}, and operates as follows: during a computation with respect to an oracle set A, whenever the machine enters the query state, in the next step of the computation the oracle tape is erased and the machine enters either of the two query answer states. Let w be the nonblank contents of the oracle tape when it entered the query state. Then if $w \in A$ the machine enters q_{yes}, else it enters q_{no}.

Definition [Restricted Nondeterministic Oracle Machine] A *restricted nondeterministic oracle machine* is an oracle machine with nondeterministic transitions subject to the following restriction: whenever the machine begins writing on a blank oracle tape it must enter a deterministic phase until the query state q_Q is entered, while retaining nondeterminism in the characters written to the oracle tape. Moreover, we add the following restrictions to our oracle model. If the oracle machine ever uses nondeterminism in making a query string, then at least one computation path writes a query string that is bound to get a "no" answer. (To implement this property, we can let a computation path write an ill-formed character on the oracle tape whenever the oracle machine is nondeterministically writing on the oracle tape.)

Note that because under our oracle model the machine operates deterministically during oracle writing phase except for the choice of characters written to the oracle tape, the machine can never "remember" the fact that it has written an ill-formed character. So its state and work tape contents are the same whether it has nondeterministically chosen to write the ill-formed character or not.

In some sense our oracle machines can be thought of as theorem provers. These machines may "prove" a set of theorems in the following manner. Whenever the machine generates a query deterministically it can make use of a "no" answer. Whenever it uses nondeterminism during the oracle-writing phase, it is only interested in whether there exists a proof of a given theorem. Since it can always guess an incorrect or ill-formed proof, it can extract no real information from a "no" answer.

A *full configuration* of a Turing machine M consists of the state, the content and head position of the input tape, and the contents and head positions of the work tapes of M. In case that M is an oracle machine, a full configuration of M also contains the content and head position of the oracle tape. A *deleted configuration* of a Turing machine (with or without an oracle tape) is a subset of a full configuration consisting of the state, input head position and currently scanned input character, and the contents and head positions of the work tapes.

Lemma 2.1 *A $s(n) = \Omega(\log n)$ space-bounded oracle machine halts on all inputs of length n in time at most $2^{O(s(n))}$.*

PROOF. The result is clearly true for deterministic space-bounded oracle machines, so it suffices to show the result for nondeterministic machines.

First we show that the length of any query is bounded. Since the oracle tape is write-only, during an oracle writing phase, each computation step transition depends only on the deleted configuration. Since the machine uses at most space $s(n)$ it follows that the number of distinct deleted configurations is bounded by $2^{ks(n)}$ for some fixed positive constant k. Thus by our model if the machine ever repeats a deleted configuration during an oracle writing phase, the computation is looping and will continue writing to the oracle tape forever, without entering the query state.

Now the number of deleted configurations following a query (the state is either q_{yes} or q_{no}) is also similarly bounded. Thus by the usual argument there exists a fixed positive constant $c > k^2$ so that if an accepting computation takes more than $2^{cs(n)}$ steps, there must be a repeated configuration and a shorter accepting computation can be found. $\square$

3 Toward an NL Hierarchy

For a complexity class C, let NL_r^C denote the class of languages recognized by $O(\log n)$ space-bounded restricted nondeterministic oracle machines using some oracle set from the class C. An NL hierarchy $\bigcup_{k\geq 1} NL_k$ is defined as follows:

$$NL_1 = NL, \quad NL_2 = NL_r^{NL}, \quad NL_3 = NL_r^{NL_2}, \quad \ldots$$

Lemma 3.1 *For each integer $k \geq 1$, $\Sigma_k^p \subseteq NL_{2k}$.*

PROOF. The proof is by induction on k. The base case $k = 1$ follows from the fact that $NP \subseteq NL_r^{DL} \subseteq NL_r^{NL}$. Here to simulate an NP machine M, we use an NL_r oracle machine to guess a computation of M, and use a DL machine to verify that the guessed computation is an accepting computation.

For the inductive step we assume that $\Sigma_{k-1}^p \subseteq NL_{2k-2}$.

Let M be a polynomial-time bounded nondeterministic oracle machine. For any oracle set A, define a language L_{M^A} as follows (where C_i, i =

$1, \cdots, m$, are full configurations of M^A):

$$L_{M^A} = \left\{ \langle x, C_0\#C_1\#\cdots\#C_m \rangle \;\middle|\; \begin{array}{l} C_0\#C_1\#\cdots\#C_m \text{ encodes an} \\ \text{accepting computation of } M^A \\ \text{on input } x. \end{array} \right\}$$

Since a DL machine equipped with the oracle A can verify that $C_i \vdash C_{i+1}$ is a valid transition of configurations of the machine M with the oracle set A for all $0 \leq i \leq m-1$, we get

$$L_{M^A} \in DL^A \subseteq NL_r^A.$$

Let $L = \mathcal{L}(M^A)$ be a language in Σ_k^p where $A \in \Sigma_{k-1}^p$. By the inductive hypothesis $A \in NL_{2k-2}$. Now we have that L_{M^A} is in $NL_r^A \subseteq NL_r^{NL_{2k-2}} = NL_{2k-1}$. So L can be recognized by an NL_r oracle machine M_1 equipped with an oracle set L_{M^A} as follows: given an input x, the NL_r oracle machine guesses a string on the oracle tape, then checks by querying the oracle set L_{M^A} that the guessed string is a valid accepting computation of M^A on the input x. Therefore, $L \in NL_r^{NL_{2k-1}} = NL_{2k}$. The inductive step follows immediately. $\square$

To show that the converse of the above lemma is also true, we first show a relativized Immerman-Szelepcsenyi theorem. The inductive counting technique plays an important role in our proof.

Theorem 3.2 *For each integer $k \geq 1$, co-$NL_r^{\Sigma_k^p} = NL_r^{\Sigma_k^p}$.*

PROOF. The idea is to use inductive counting to find all the reachable deleted configurations of the NL_r oracle machine M^A on input x of length n, where $A \in \Sigma_k^p$, and accept if and only if none are accepting. We carry out this computation using an NL_r machine M_1 with Σ_k^p oracle. The inductive counting is carried out as in Immerman's proof, with the exception that we regard an entire oracle writing phase as atomic. Specifically we call an *atomic step* of M one computation step in which the oracle tape remains blank, or a sequence of steps, starting with a step in which an initially blank oracle tape is written to, and ending in the first subsequent step in which the query state is entered. Then we compute all the reachable deleted configurations in which the oracle tape is blank. We first show how to determine whether one deleted configuration of M yields another in one atomic step.

Suppose we are given two deleted configurations d_1 and d_2 of M^A, both with a blank oracle tape. If d_1 is not writing to the oracle tape, then M_1 can certainly determine whether $d_1 \vdash d_2$.

Suppose now that d_1 is writing to a currently blank oracle tape and d_2 is in the state q_{yes} or q_{no}. We can first attempt to have M_1 simulate the computation of M. We can always determine if $d_1 \overset{*}{\vdash} d_2$, except for the state of d_2 (which can be either q_{yes} or q_{no}). This is because by assumption the entire atomic step of M is deterministic except for the characters written to the oracle tape.

In case nondeterminism was used in generating the query, we have several subcases. If the state of d_2 is q_{no}, then we can determine if $d_1 \overset{*}{\vdash} d_2$, irrespective of any oracle query answer M_1 gets. This is because by our model if a query is generated nondeterministically, at least one computation path will get a "no" answer from A.

If the state in d_2 is q_{yes}, M_1 could look at what answer it got for its querying on its own oracle tape. We may have gotten lucky and M_1 has already queried a string w that gets a "yes" answer from A. In this case, again we can determine if $d_1 \overset{*}{\vdash} d_2$. However, if we are not so fortunate and M_1 got a "no" answer for its query, we cannot simply say that d_2 is not reachable from d_1, since a "no" answer makes no sense to M_1 as M_1 simulates a nondeterministic oracle writing phase of M. Here we show how to solve this problem by using the closure of Σ_k^p under polynomially bounded existential quantification. We essentially need an answer to the following question:

> Does there exist a partial computation (atomic step) of M on input x, taking a deleted configuration d_1 to a deleted configuration d_2 in the query state, so that a string $w \in A$ is queried?

Let B be the set of triples $\langle x, d_1, d_2 \rangle$ that give a "yes" answer to the above question. To see that $B \in \Sigma_k^p$, note that an $NP^{\Sigma_{k-1}^p}$ machine can nondeterministically guess the partial computation of M, write its query w to a work tape, and then perform an $NP^{\Sigma_{k-1}^p}$ computation that verifies $w \in A$.

Let $C = A \oplus B = \{0x | x \in A\} \bigcup \{1x | x \in B\}$. Then clearly, $C \in \Sigma_k^p$.

Now we can carry out inductive counting with the $NL_r^{\Sigma_k^p}$ machine M_1^C. Let the run time of M be bounded by the polynomial $p(n)$. As in Immerman's proof, suppose inductively that we have computed m_{i-1} the number of deleted configurations reachable in at most $i-1$ of the above atomic steps of M. We show how to compute m_i.

M_1^C can lexicographically iterate through all deleted configurations d_t. For each such target configuration d_t, we also iterate through each deleted configuration d, see if we can guess a computation of M of at most $i-1$ atomic steps to our intermediate target d. If so we decrement a counter initialized to m_{i-1}. Then we determine if d yields d_t in an atomic step. Since the query string $\langle x, d, d_t \rangle$ to the oracle set B can always be written deterministically, both "yes" and "no" answers to the query can be used by our oracle machine, i.e., the question $d \overset{*}{\vdash} d_t$ can be answered definitely by either "yes" or "no". If $d \overset{*}{\vdash} d_t$, we increment our counter for m_i and generate the lexicographical successor to the target d_t. Else we generate the successor to the intermediate target d. If we guess computations to m_{i-1} such intermediate targets d, and none yield d_t, we can continue with the next target. Thus given m_{i-1} we can successfully compute m_i and carry out the inductive counting.

After successfully computing $m_{p(n)}$ we can determine if M^A rejects x. The work tapes use just enough space to store several counters that count up to $p(n)$, and to store several deleted configurations at once. The work space is thus seen to be at most $c \log n$ for some positive constant c. $\square$

With the previous theorem in hand it is now a straightforward matter to use "oracle replacement" to show

Theorem 3.3 *The NL_r-hierarchy is a refinement of the polynomial-time hierarchy, that is, for each integer $k \geq 1$,* $\quad NL_{2k} = \Sigma_k^p$.

PROOF. We show by induction on k that $NP^{\Sigma_{k-1}^p}$ can simulate NL_{2k}. Let $L \in NL_{2k}$. Let $L = \mathcal{L}(M_1{}^A)$, where M_1 is an NL_r oracle machine and we can assume by the inductive hypothesis that $A \in NL_{2k-1}$ is accepted by an NL_r oracle machine M_2 with Σ_{k-1}^p oracle B. Then by the previous theorem there exists an NL_r machine M_3 with Σ_{k-1}^p oracle C which accepts $\overline{A}$.

An NP machine with Σ_{k-1}^p oracle $B \oplus C$ can accept L as follows. It first simulates M_1 on input x, writing each query to a work tape. To get the answer to a query w it nondeterministically simulates either M_2 or M_3, using oracle $B \oplus C$, and continues the simulation just in the case that w has been accepted by either M_2 or M_3. In this case it has a definite answer to the query w and can continue simulating M_1. Thus $L \in \Sigma_k^p$. Thus $NL_{2k} \subseteq \Sigma_k^p$.

Combining with Lemma 3.1, we obtain the theorem. $\square$

The same technique enables us to give the following results which generalizes the well-known results $NL \subseteq P$ and $NL \subseteq DL^2$ in unrelativized worlds.

Theorem 3.4 For each integer $k \geq 1$, we have

$$NL_r^{\Sigma_k^p} \subseteq P^{\Sigma_k^p} \qquad \text{and} \qquad NL_r^{\Sigma_k^p} \subseteq (DL^2)^{\Sigma_k^p}$$

where $(DL^2)^{\Sigma_k^p}$ is the set of languages accepted by $O(log^2 n)$ space-bounded deterministic oracle Turing machines relative to oracle sets in Σ_k^p.

Proof. The well-known proofs for the unrelativized cases can be modified in a straightforward manner. □

We point out that our result that $NL_{2k} = \Sigma_k^p$ must not be perceived as a simple generalization of the trivial fact that $NL^{NL} = NP$. In fact, a similar result for the unrestricted space-bounded model would imply the collapse of the polynomial-time hierarchy, as we will show below:

Theorem 3.5 If we define the NL hierarchy $\left\{NL_k^{(u)}\right\}$, in the same way, under the unrestricted space-bounded oracle machines, then either $NL_{2k}^{(u)} = \Sigma_k^p$ or $NL^{\Sigma_k^p} = \text{co-}NL^{\Sigma_k^p}$ imply the collapse of the polynomial-time hierarchy.

Proof. It is easy to check that $NL^{\Sigma_k^p} = \Sigma_{k+1}^p$. Thus $NL_{2k}^{(u)} = \Sigma_k^p$ for $k \geq 2$ would imply $NL_{2k+1}^{(u)} = NL^{\Sigma_k^p} = \Sigma_{k+1}^p = NL_{2k+2}^{(u)}$ and $NL^{\Sigma_k^p} = \text{co-}NL^{\Sigma_k^p}$ would imply $\Sigma_{k+1}^p = \text{co-}\Sigma_{k+1}^p$. □

References

[1] T. Baker, J. Gill, and R. Solovay, "*Relativizations of the $P = NP$ question*," **SIAM J. Comput. 4**, 1975, pp. 431-442.

[2] J. Hartmanis, "*New developments in structural complexity theory*," **Theoretical Computer Science 71**, 1990, pp. 79-93.

[3] N. Immerman, "*Nondeterministic Space is Closed Under Complement*," **SIAM J. Comput. 17** 1988, pp. 935-938.

[4] R. E. Ladner and N. A. Lynch, "*Relativization of Questions about log Space Computability*," **Math. System Theory 10 (1)**, 1976, pp. 19-32.

[5] K. Lang, B. Jenner, and B. Kirsig, "*The logarithmic alternation hierarchy collapses: $A\Sigma_2^L = A\Pi_2^L$*," **Proc. Automata, Languages, and Programming**, Lecture Notes in Computer Science 267, 1987, pp. 529-541.

[6] W. Ruzzo, J. Simon, and M.Tompa, "*Space-Bounded Hierarchies and Probabilistic Computations*," **Journal of Computer and System Sciences 28**, 1984, pp. 216-230.

[7] A. L. Selman, "*Reductions on NP and P-selective sets*," **Theoretical Computer Science 19**, 1982, pp. 287-304.

[8] R. Szelepcsenyi, "*The method of forcing for nondeterministic automata*," **Bull. EATCS 33**, 1987, pp. 96-100.

[9] S. Toda, "*Σ_2PSPACE(n) is closed under complement*," **Submitted for publication**, 1987.