Languages Simultaneously Complete for One-Way and Two-Way Log-Tape Automata

by

J. Hartmanis

and

S. Mahaney

TR 80-413

Department of Computer Science
Cornell University
Ithaca, New York 14853

Languages Simultaneously Complete for One-Way and Two-Way Log-Tape Automata

J. Hartmanis* and S. Mahaney
Department of Computer Science
Cornell University
Ithaca, New York 14853

Abstract

In this paper we study languages accepted by nondeterministic log n-tape automata which scan their input only once and relate their computational power to two-way, log n-tape automata. We show that for the one-way, log n-tape automata the nondeterministic model (1-NL) is computationally much more powerful than the deterministic model (1-L); that under one-way, log n-tape reductions there exist natural complete languages for these automata and that the complete languages cannot be sparse. Furthermore, we show that any language complete for nondeterministic one-way log n-tape automata (under 1-L reductions) is also complete for the computationally more powerful nondeterministic two-way, log n-tape automata (NL), under two-way, log n-tape reductions. Therefore, for all bounds $T(n)$, $T(n) \geq \log n$, the deterministic and nondeterministic $T(n)$-tape bounded computations collapse iff the nondeterministic one-way log n-tape computations can be carried out by two-way deterministic log n-tape automata.

*This research has been supported in part by National Science Foundation Grant MCS 78-00148.

1. Introduction

The work in computational complexity theory has been strongly influenced by the study of the families of languages accepted by Turing machines in polynomial tape, nondeterministic polynomial time, deterministic polynomial time, nondeterministic logarithmic tape and deterministic logarithmic tape, denoted by

PTAPE, NP, P, NL and L,

respectively [1,5].

Though the proper containment relations between these families of languages are not yet known, they form a natural hierarchy to classify the complexity of many practical computational problems by their membership in these families. Furthermore, each of these families contain many natural complete problems which, in a sense, characterize and represent their computational complexity. Currently, among the most challenging problems in the theory of computation is to establish the proper containment relations between these families of languages and to gain a better understanding of their structure.

In this paper we show that the above described "hierarchy" of families of languages, PTAPE, NP, P, NL and L, can be naturally extended downward to one-way, log n-tape deterministic and nondeterministic Turing machines. These are (nondeterministic) two-tape Turing machines with a one-way input tape (with end marker) and for an input of length n a two-way, read-write work tape of length $\lceil \log_2 n \rceil$ (with end markers). We denote the families of languages accepted by the nondeterministic and deterministic models by 1-NL and 1-L, respectively.

The results in this paper show that these automata exhibit some very interesting properties and that they capture our intuitive idea of what can be achieved "by guessing and polynomial bounded counting in one scan of the input."

Their importance is further enhanced by their relations to nondeterministic two-way, log n-tape automata. We prove that every complete 1-NL language, under one-way, log n-tape reductions, is also complete for NL languages, under two-way, log n-tape reductions. Thus, these two different families of languages share many complete sets. From this follows that for all tape bounds T(n), $T(n) \geq \log n$, the deterministic and nondeterministic T(n)-tape bounded computations collapse iff the nondeterministic one-way, log n-tape computations can be performed by two-way, deterministic log n-tape machines. Thus, we see that the basic question about eliminating nondeterminism in tape bounded computations is equivalent to the question whether we can replace nondeterminism in one-way, log n-tape computations by deterministic two-way computations.

Finally, we show that if NL$\neq$L then there exist natural incomplete langauges in 1-NL - 1-L. Recall that the incomplete langauges in NP-P, under the assumption that P$\neq$NP, are obtained by delayed diagonalization and that no natural incomplete languages are known for NP [9].

2. 1-NL Languages

First, we observe that, though we do not yet know whether

$$P \neq NP \text{ and } L \neq NL$$

we can easily show that for one-way, log n-tape computations the nondeterministic computations are more powerful than the deterministic ones, i.e.,

$$\text{1-L} \overset{c}{\neq} \text{1-NL}$$

Actually, we prove that the gap between deterministic and nondeterministic one-way, log n-tape computations is exponential. This shows that the one-way computations with small amounts of work tape behave differently than two-way, tape-bounded computations (with log n or more tape) which going from nondeterministic to deterministic computations needs no more than to square the amount of tape used [11].

Let 1-TAPE[T(n)] denote the family of languages accepted by deterministic, one-way Turing machines with T(n) work tape.

Theorem 1:

If T(n) is such that

$$\lim_{n\to\infty} \frac{T(n)}{n} = 0$$

then

$$\text{1-NL} \nsubseteq \text{1-TAPE}[T(n)].$$

Proof: The language

$$A = \{u \# v \mid u, v \in \Sigma^*, \# \notin \Sigma \text{ and } u \neq v\}$$

is in 1-NL but not in 1-TAPE[T(n)], if the above limit condition holds for T(n). To see this, note that A can be recognized by a nondeterministic, one-way, log n-tape machine which guesses whether $|u| \neq |v|$ or $|u| = |v|$ and $u \neq v$; in the first case the guess is easily verified on log n-tape, in the second case the machine guesses in which digit the two strings differ and then counts up to the position in u and v, respectively, using its work tape to verify its guess.

On the other hand, if

$$\lim_{n\to\infty} \frac{T(n)}{n} = 0$$

then for any q, $q > 0$, and sufficiently large n

$$q^{T(n)} < 2^n .$$

Therefore for sufficiently long sequences u_1 and u_2, $u_1 \neq u_2$, the machine must be in the same total configuration after scanning

$$u_1\# \text{ and } u_2\# .$$

But then

$$u_1 \# u_1 \text{ and } u_2 \# u_1$$

are either both accepted or both rejected, showing that A is not in 1-TAPE[T(n)]. ∎

Next we show that there exist natural complete problems for the 1-NL family of languages. To do this we use deterministic one-way, log n-tape reductions that were introduced and studied in [6]. A 1-L transducer is a deterministic, one-way, log n-tape machine with a one-way output tape. From Theorem 1 we know that the 1-L transducers are not computationally strong enough to recognize the languages in 1-NL.

Recall that it is not yet known whether P $\neq$ NP and/or L $\neq$ NL, and therefore we do not know whether the polynomial time reductions used to study NP problems are not sufficiently powerful computationally to recognize directly all the languages in NP; the related question about two-way, log n-tape reductions also remains open. On the other hand, from [6] we know that the well known "classic" complete problems for PTAPE, NP, P, NL and L, respectively, all remain complete for these families under one-way, log n-tape reductions and that these transducers are not capable of recognizing the languages in L. At the same time, it was also shown that there are complete problems, say, in NP under polynomial time reductions which are not complete under one-way, log-n tape reductions [6].

We recall that there is a nice hierarchy of "graph connectivity problems" which form complete sets for PTAPE, P, NL, and L. The complete problem that we will define below for 1-NL fits in naturally in this hierarchy, extending it downward. For the sake of comparison we describe the other complete "graph connectivity" problems.

A complete language for PTAPE is formed by all directed graphs with an IN and OUT node for which there exists a winning strategy for the first player of the game of hex on this graph [1].

A complete language for P is formed by all directed graphs whose nodes are labelled with either AND or OR and are such that the IN and OUT nodes are connected by a path system which must take all possible edges out of an AND node and some edge out of an OR node.

A complete language for NL is formed by the set of all directed graphs with an IN and OUT node, which have a directed path from the IN to the OUT node [7].

A complete language for L is formed by all directed graphs of outdegree one for which there is a path from the IN to the OUT node [3,7].

We will say that the representation of a directed, acyclic graph, G, is <u>topologically sorted</u> if for any pair of edges (a,b) and (b,c) in G, edge (a,b) is listed before (b,c).

Next we show that the problem of determining whether the topologically sorted representations of acyclic graphs have a directed path from the IN to an OUT node is a complete 1-NL problem. Denote the set of these graph representations by TAGAP It is seen that TAGAP fits in naturally in the hierarchy of the other complete "graph connectivity problems," for PTAPE, P, NL and L, where the acyclic nature of the graphs directly mirrors the limited ability of reading the input only once. Similarly,

the difference between the complete graph problems for NL and L is captured in the difference between the outdegree of the graphs: for NL we can use graphs with outdegree $k \geq 2$, whereas for L we are restricted to outdegree one.

Theorem 2: The set of topologically sorted representations of directed, acyclic graphs with a directed path from the IN to an OUT node is a complete language for l-NL.

Proof: TAGAP is contained in l-NL since a one-way, nondeterministic device can successively guess on its log n tape the sequence of nodes which form a path from IN to OUT. Since the representation of the graph is topologically sorted the correctness of these guesses can be verified in one scan of the input.

To see that TAGAP is complete for l-NL, we show that for every one-way, log n-tape, nondeterministic machine M, we can reduce every input string w (by a l-L transducer) to an acyclic graph in which IN is connected to OUT iff w is accepted by M. The nodes of the graph G_w, corresponding to the input w, consist of triplets

$$(n_1, n_2, x),$$

where n_1, $1 \leq n_1 \leq |w| = n$, indicates the head position of M on the input tape, n_2, $0 \leq n_2 \leq q^n$, indicates the number of operations performed since the last head move on the input, and x describes the configuration of M (input symbol scanned, state, worktape content and head position on work tape).

For any fixed M we can construct a one-way, deterministic, log n-tape transducer which for input w enumerates in topological order on its output tape all pairs of nodes

$$((n_1, n_2, x),(n_1', n_2', x'))$$

such that in one move the machine M goes from the total configuration described by (n_1, n_2, x) to the one described by (n_1', n_2', x'). If the input head is moved, then

$$n_1' = n_1 + 1 \quad \text{and} \quad n_2' = 0$$

otherwise, if the machine performs an operation and $n_2 < q^n$,

$$n_1' = n_1 \quad \text{and} \quad n_2' = n_2 + 1 .$$

We denote the pair

$((n_1, n_2,x),(n_1',n_2', x'))$ by $((n_1, n_2, x), \text{Succ}(n_1, n_2, x))$.

The enumeration is described by the following program:

```
begin
  For each n1 = 1,2,...,|w| do
    begin
      For each n2 = 0,1,2,...,|w| do
        begin
          For each x, |x| ≤ [log|w|] do
            begin
              print all pairs ((n1,n2,x),Succ(n1,n2,x))
            end
        end
    end
end
```

The resulting graph is topologically sorted and the IN node, the initial configuration, is connected by a directed path to an OUT node, a node whose configuration contains the halt state, iff w is accepted by M.

Thus we see that TAGAP is complete for 1-NL. ∎

3. Relations Between One-Way and Two-Way Computations

We know that the nondeterministic two-way log n-tape computations are computationally much more powerful than the nondeterministic one-way log n-tape computations, i.e., 1-NL ≠ NL. Nevertheless, we will show in this section that there are sets which are simultaneously complete for both families of languages. More precisely, we know that TAGAP is complete for 1-NL under one-way, log n-tape reductions and we prove below that TAGAP is also complete for NL under two-way, log n-tape reductions. Later we will see that this result has some very interesting implications.

Theorem 3: TAGAP is a complete language in NL under two-way, log n-tape reductions.

Proof: Since TAGAP is in 1-NL it is also contained in NL.

To show that TAGAP is complete for NL, let A be a language in NL. We know that A is accepted by a nondeterministic two-way, log n-tape automaton M. This automaton can be easily converted to an oblivious automaton, M', that moves its input head in complete sweeps from (left) end marker to (right) end marker (and back). In other words, the head movements on the input tape are oblivious of the input and machine configuration. Furthermore, we know that for M' there exists a k such that if any input w is accepted then w is accepted by M' within $k+n^k$ operations, $n=|w|$.

We will now show that a deterministic two-way, log n-tape transducer can reduce any set A in NL, using an oblivious acceptor M' of A, to topologically sorted representations of acyclic graphs, G_w, such that an OUT node of G_w can be reached from the IN node iff w is in A.

For input w, $|w|=n$, the graph G_w has nodes of the form

$$(n_1,n_2,n_3,x)$$

where: n_1, $1\le n_1\le k+n^k$, counts the sweeps on input w,

n_2, $1\le n_2\le n$, indicates the head position on the input tape,

n_3, $0\le n_3\le k+n^k$, counts the operations performed since the last head motion on the input tape,

x, $x\in\Gamma^*$, $|x|\le\log n$, represents a configuration of M'.

The graph G_w has an edge of the form

$$((n_1,n_2,n_3,x),(n_1',n_2',n_3',x'))$$

iff in one possible operation M' transforms the state described by (n_1,n_2,n_3,x) to (n_1',n_2',n_3',x'). The IN node is again the initial configuration of M' on w and the OUT nodes are the nodes with an accepting state in x.

For input w the two-way, log n-tape transducer enumerates all the nodes of G_w in sequence for sweeps, head position, how long it has computed without moving the head and the machine configuration and lists the possible edges of G_w, if there is a one step transition

$$((n_1,n_2,n_3,x),(n_1',n_2',n_3',x'))$$

This can be done by an algorithm similar to the one used in the proof of Theorem 2.

It is easily seen that G_w has the desired property that IN is connected to an OUT iff $w \in A$ and that G_w is an acyclic graph, printed in a topologically sorted form. Thus TAGAP is complete for NL, as was to be shown. ∎

From the previous result we can see that for all tape bounds T(n), $T(n) \geq \log n$, the deterministic and nondeterministic T(n)-tape bounded computations collapse, i.e.,

$$TAPE[T(n)] = NTAPE[T(n)],$$

if and only if nondeterminism in one-way, log n-tape computations can be eliminated by using deterministic two-way computations.

Corollary 4: 1-NL$\subseteq$L iff NL=L.

Proof: NL=L implies that 1-NL$\subseteq$L.

Conversely, 1-NL$\subseteq$L implies that TAGAP is in L. Since TAGAP is complete for NL we see that L=NL. ∎

The above corollary puts a premium on understanding the power of two-way computations and their relation to one-way, nondeterminist computations. It is interesting to recall that the related quest about one-way, nondeterministic pushdown automata and two-way deterministic pushdown automata also remains unsolved: we know that deterministic two-way pushdown automata can accept languages which are not context-free, but we do not know whether they can accept all context-free languages.

From the study of NP complete problems we know that if P$\neq$NP, then there exist incomplete languages in NP-P[9]. On the other hand, the known incomplete languages are constructed by delayed diagonalization and so far no natural languages are known to be incomplete.

For 1-NL languages the situation is quite different. Let A be given by

$$A = \{u\#v \mid u, v \in \Sigma^*, \# \notin \Sigma \text{ and } u \neq v\}.$$

Theorem 5:

If L$\neq$NL then the language A is not complete for 1-NL and is in 1-NL but not in 1-L.

Proof: The language A is seen to be in 1-NL and also in L, but, by Theorem 1, A is not in 1-L. If A is complete for 1-NL then, by Theorem 2, it is also complete for NL, but then NL=L. ∎

Non Existence of Sparse Complete Sets

In this section we show that there cannot exist sparse complete sets for 1-NL and compare this result with the not yet completely resolved question for NP complete languages. In the last section we discuss the corresponding problem for complete NL problems under two-way, log n-tape reductions.

A set B, $B \subseteq \Sigma^*$, is said to be sparse iff there exists a k such that for all n

$$|B \cap \Sigma^n| \leq k + n^k$$

It has been shown [2,5] that the known complete sets in NP (and similarly in PTAPE) are isomorphic under polynomial time mappings and therefore the known complete languages are similar in a very strong technical sense. The existence of sparse complete languages for NP would prove that not all complete languages for NP are polynomial time isomorphic and therefore there would exist (as yet undiscovered) radically different types of complete languages. Furthermore, it would prove that the necessary information to solve NP complete problems can be condensed into a sparse oracle tape (which could be computed once up to sufficiently long strings and then used to solve NP problems in deterministic polynomial time) [2,5].

At the present we do not know if there exist sparse complete sets in NP under polynomial time reductions. Recently it has been shown that if a language A over a single letter alphabet, $A \subseteq a^*$, is complete for NP then P = NP [3]. Similarly, if there exists a sparse complete language for CO-NP then P = NP [4]. Furthermore, the existence of a sparse complete language in PTAPE implies that P = PTAPE and therefore P = NP = PTAPE [10]. Unfortunately, the main problem, whether there can exist sparse NP complete sets under polynomial time reductions without forcing P = NP, remains unsolved.

The situation is different for one-way, log n-tape reductions, for which we show next that there cannot exist sparse complete sets in

1-NL, L, NL, P, NP and PTAPE.

Theorem 6: There are no sparse complete sets in 1-NL, L, NL, P, NP and PTAPE under 1-L reductions.

Proof: We will prove that there exist sets in 1-NL which cannot be reduced to any sparse sets by 1-L reductions. Suppose, to the contrary, that TAGAP is reduced to a sparse set A.

For a 1-L transducer let CONFIG be the total configuration consisting of: the input tape square scanned, the machine state, work-tape content and head position, and output tape content.

Consider as inputs topologically sorted representations of dags which consist of nodes labelled IN, 1, 2, ..., n, and OUT. The edges will be (IN, i) for all i in some set and (j, OUT) for some j.

After reading the prefix $T = \{(IN,i) : i \in REACH\}$ the set REACH is exactly the set of nodes that can connect IN to OUT. If $REACH \neq \phi$, then a suitable choice of (j,OUT) can put the graph into TAGAP. Thus, after reading a string in prefix T, the 1-L reducer for TAGAP to A must print a string from prefixes of A on its output tape.

Since A and PREFIX(A) are sparse sets we see that any time during the reduction of a directed tree of size n the reducer is in one of polynomially many different CONFIG (including the content of the output tape). On the other hand, there are exponentially many graphs with descriptions of length n and different REACH sets. Therefore, there exist two graphs T_1 and T_2, with $REACH[T_1] \neq REACH[T_2]$, which are mapped by the 1-L transducer onto the same

total configuration. But then, by adjoining the same appropriate edge to both of them, connecting some j to the OUT node in T_1 and T_2, we place T_1 in TAGAP and T_2 not in TAGAP. Since the l-L transducer maps T_1 and T_2 onto the same element we see that TAGAP cannot be reduced to a sparse set. ∎

Two-Way Reductions and An Open Problem

It is interesting to note that for l-L reductions we cannot have sparse complete sets in l-NL, L, NL, etc., nor can we have a complete set A over a single letter, $A \subseteq a^*$ for NP under polynomial reductions, if NP $\neq$ P. On the other hand, so far we have not been able to show that the existence of a complete set over a single letter alphabet for NL would imply that

$$L = NL.$$

The assumption that there exists a complete set A, $A \subseteq a^*$ for NL leads to some strange implications.

We know that if L = NL then the deterministic and nondeterministic computations for larger amounts of tape are also equal. In particular, NL = L implies that the deterministic and nondeterministic context-sensitive languages are the same, DCSL = NDCSL. On the other hand, we get the following result.

Fact 6: If A, $A \subseteq a^*$, is complete for NL, then
DCSL = NDCSL implies that L = NL.

Furthermore, if A, $A \subseteq a^*$, is complete for NL then every language in NL can be recognized by a log n-tape automaton which uses its nondeterminism only after it has scanned the input tape.

We refer to a two-way, log n-tape automaton which operates deterministically while it scans its input tape and only uses nondeterministic operations (on its log n work tape) after the input has been completely scanned, as a restricted nondeterministic, log n-tape automaton and designate the family of languages accepted by these automata by RNL.

Theorem 7: There exists a complete NL language A, $A \subseteq a^*$ iff

$$RNL = NL.$$

Proof: If RNL = NL then we can construct from any complete language A for NL a complete language A' for NL such that $A' \subseteq a^*$. To do this, let M be the restricted recognizer for A. For each input w let w' be the content of the work tape after M has finished scanning the input w in a deterministic mode. Let the complete set A', $A' \subseteq a^*$, be the set obtained from A by expressing in unary form all the deterministically computed work tape contents, w', which we can view as binary representation of integers.

Clearly, there is an L reducer of A to A' and since A is a complete set for NL, it is easily seen that A', $A' \subseteq a^*$, is a complete set for NL.

Conversely, let A, $A \subseteq a^*$, be a complete set for NL. Then any other set B in NL can be reduced to A and the resulting element a^k can be represented on the transducer's log n tape as a binary number. After that we simulate on a separate track of the log n tape the acceptor of A (the input a^k is simulated by counting up to k) and accept

iff the simulated acceptor accepts. Since the above device uses nondeterminism only after it has finished scanning the input (computing a^k), we see that any set in NL is accepted by a restricted machine. Thus

$$RNL = LN,$$

as was to be shown. ∎

References

[1] A. V. Aho, J. E. Hopcroft and J. D. Ullman, "The Design and Analysis of Computer Algorithms," Addison-Wesley, Reading, MA, 1974.

[2] L. Berman and J. Hartmanis, "On Isomorphisms and Density of NP and Other Complete Sets," SIAM J. Comput., 6 (1977), pp. 305-323.

[3] P. Berman, "Relationships Between Density and Deterministic Complexity of NP Complete Languages," Fifth Int. Symp. on Automata, Languages and Programming, Udine, Italy, Springer-Verlag Lecture Notes in Computer Science, Vol. 62, 1978, pp. 63-71.

[4] S. Fortune, "A Note on Sparse Complete Sets," SIAM J. Comput., 8(1979), pp. 431-433.

[5] J. Hartmanis, "Feasible Computations and Provable Complexity Properties," CBMS-NSF Regional Conference Series in Applied Math., Vol. 30, SIAM, Philadlephia, PA, 1978.

[6] J. Hartmanis, N. Immerman and S. Mahaney, "One-Way Log-Tape Reductions," IEEE 19th Annual Symposium in Foundation of Computer Science," October 16-18, 1978, pp. 65-72.

[7] N. D. Jones, "Space-Bounded Reducibility Among Combinatorial Problems," J. Comput. System Sci., Vol. 11 (1975), pp. 68-85.

[8] N. D. Jones and W. T. Laaser, "Problems Complete for Deterministic Polynomial Time," Theoret. Comput. Sci., Vol. 3 (1977), pp. 105-117.

[9] R.E. Ladner, "On the Structure of Polynomial Time Reducibility," J. Assoc. Comput. Mach. , Vol. 22 (1975), pp. 155-171.

[10] A. R. Meyer and M. S. Paterson, "With What Frequency are Apparently Intractable Problems Difficult?", to be published.

[11] W. J. Savitch, "Relations Between Nondeterministic and Deterministic Tape Complexities," JCSS, Vol. 4 (1970), pp. 177-192.