ONE-WAY LOG-TAPE REDUCTIONS

by

J. Hartmanis*
N. Immerman+
S. Mahaney*

TR78-347

Department of Computer Science
Cornell University
Ithaca, New York 14853

*This research has been supported in part by National Science Foundation grants MCS 75-09433 and MCS 78-00418.
+This research has been supported by a National Science Foundation Fellowship.

ONE-WAY LOG-TAPE REDUCTIONS

by

J. Hartmanis*
N. Immerman+
S. Mahaney*

Cornell University
Ithaca, New York

Abstract

One-way log-tape (1-L) reductions are mappings defined by log-tape Turing machines whose read head on the input can only move to the right. The 1-L reductions provide a more refined tool for studying the feasible complexity classes than the P-time [2,7] or log-tape [4] reductions. Although the 1-L computations are provably weaker than the feasible classes L, NL, P and NP, the known complete sets for those classes are complete under 1-L reductions. However, using known techniques of counting arguments and recursion theory we show that certain log-tape reductions cannot be 1-L and we construct sets that are complete under log-tape reductions but not under 1-L reductions.

1. Introduction.

In this paper we study reductions that are provably weaker than the feasible complexity classes; i.e. these reductions cannot recognize all the sets in the families L, NL, P or NP. The P-time reductions [2,7] that have been used to study the classes P and NP can recognize sets in P. Thus the reductions do not aid in separating the two classes. The log-tape reductions [4,5,6], while useful to study smaller feasible classes such as P and NL, do not help in separating classes since it is not known if $P \neq L$.

We define here one-way log-tape (1-L) reductions by machines that are weaker than log-tape in their power to recognize sets. These reductions are restricted to reading their input once from left to right. They retain very little information about the input on their log work tape. Nevertheless the 1-L reductions are powerful enough to give complete sets for L, NL, P, NP etc. similar to those found under P-time or log-tape reductions. We show here that an understanding of the 1-L reductions would determine the relations among L, NL, P, NP etc. For example, we show that $L \neq NL$ if and only if there do not exist certain easily definable 1-L reductions between two classes of graphs.

This approach reduces the study of relations among the feasible classes to the study of mappings by computationally very weak one-way automata where known techniques such as counting arguments or crossing sequence arguments could be used. In this paper we prove a variety of results about 1-L reductions and indicate how to exploit the weakness of the reductions to study the feasible complexity classes.

We note first that log-tape is necessary for reductions.

Proposition 1.1 If

$$\lim_{n \to \infty} \frac{L(n)}{\log(n)} = 0,$$

then there are no complete sets for P, NP, PTAPE etc. under $L(n)$-tape bounded reductions.

Proof. The output of $L(n)$-tape bounded reductions cannot be polynomially greater than their input size. Thus if there were a complete set in P under $L(n)$ tape bounded reductions, then we could show all sets in P to be in

$$\text{Time } (n^k)$$

for some fixed k. The proof is similar for other classes.#

The usual definition of log-tape reduction has a two-way read-only input of length n, a log(n) bounded work tape, and a one-way write-only output tape. We define 1-L reductions similarly but restrict the input tape to be read once.

Definition 1.2. A) A one-way log-tape reduction (or 1-L reduction) is a mapping defined by a Turing Machine that on input of length n

1) starts with a log(n) work tape laid off, on which it may read, write, and move both ways.

2) reads the input tape once from left to right, and

3) writes the output from left-to-right on a write-only tape.

In the usual way [6] we can define for 1-L reductions:

B) X is reducible to Y (write $X <_{(1\text{-}L)} Y$), and

C) hard and complete sets for a class of sets.

Proposition 1.3. Let R, S, and T be languages over some alphabet. Then

1) If R $<$(1-L) S and S $<$ (1-L) T, then R $<$(1-L) T; and
2) if R is in 1-L reducible to S and S is in L, NL, P, NP or PTAPE, then R is in the same class.

The definition of 1-L reductions deserves some comments. First the 1-L reductions can depend on input formats in ways that log-tape reductions will not (see the discussion in [6]). This increases the burden of specifying the formats for the inputs and outputs of 1-L reductions. We have shown, however, that this poses no difficulty. Specifically, we have checked the complete sets and reductions of

1) Karp [7] and Aho, Hopcroft and Ullman [1] for NP,
2) Jones and Laaser [5] for P, and
3) Jones, Lien and Laaser [6] for NL

and have found that the complete sets in those sources are all complete under 1-L reductions. Many of the reductions require modification as Theorem 4.1 shows.

As a second point, 1-L computations are provably weaker than log-tape. As recognizers of sets, 1-L computations cannot recognize

$\{w\#w : w$ is a word over $\{0,1\}\}$.

Thus the 1-L computations are weaker than L and any larger classes.

Our goal in studying 1-L reductions is to find a sharp tool that might be used to separate feasible complexity classes. It is not our intent to promote 1-L reductions for establishing the equivalence of problems, as this is done easily with P-time and log-tape reductions.

The remainder of this paper treats three subjects. In Section 2 we characterize the class L by complete sets and discuss the relation of 1-L reductions to showing $L \neq NL$. Section 3 shows a possible paradigm for separating complexity classes. In Section 4 we give several theorems that prove the non-existence of 1-L reductions where stronger reductions are known to exist.

2. The Class L.

In this section we consider the class L consisting of sets accepted in deterministic log-tape. Clearly log-tape reductions are too powerful to give a useful characterization of L by complete sets. Jones [4] studied the class L and gave complete sets using log-rudimentary reductions. However, these reductions do not correspond to any natural machine model and are difficult to use. We can easily characterize the class L using 1-L reductions.

<u>Theorem 2.1</u>. The following two sets are complete for L under 1-L reductions.

1) $\{Mi \# x \# |x|^k \mid M_i$ accepts x with a $\log(|x|)$ work tape that uses at most 2^k symbols.$\}$

2) GAP1 = $\{ \langle s, s. G\rangle \mid G$ is a directed graph in which every node has outdegree at most 1 and node g is reachable from node s by a directed path.$\}$

<u>Proof</u>. 1) is straightforward. We prove 2) in detail to illustrate a 1-L reduction and for use in the proof of Theorem 2.2. Assume G is formatted as a list of pairs (v1, v2) denoting an edge from v1 to v2. To recognize members of GAP1 with deterministic log-tape,

we define an automaton that follows the path reachable from s. This can be done with two pointers and a counter each requiring log(n) bits. The path from s terminates either in a final node or a loop (see Fig. 1). Clearly the two pointers can follow the path and the counter can shut off the machine in case of looping.

To reduce an arbitrary set in L to GAP1 we proceed similar to [4,9] in which it was shown that the GAP problem for graphs of outdegree greater than one was complete for NL. Let Mi be a TM that operates in deterministic log-tape. The instantaneous descriptions (ID's) of Mi can be given by

a) the state of Mi,

b) the position on the work tape,

c) the symbols on the work tape, and

d) the position on the input tape.

On an input x of length n we can encode these in log(n) tape. The reduction generates the graph of all pairs (ID1, ID2) where ID1 $\vdash$ ID2, with the exception that all ID's with accepting states are smashed into a single instantaneous description, IDs, from which no other ID follows. The 1-L reduction keeps the input position recorded on a track of the work tape. At each input position the work tape is used to generate all possible values of a, b and c. For each IDj so generated, the reducer determines the next ID, IDk, that Mi would have and prints (IDj, IDk) on the output tape. The start node, IDs, corresponds to the starting configuration of Mi. Then there is a path from IDs to IDg if and only if there is a sequence IDs $\vdash \ldots \vdash$ IDg if and only if Mi accepts x. Finally the outdegree of the graph is at most one since Mi is deterministic.#

The 1-L reductions easily give a partial solution to a problem posed by Jones, Lien and Laaser [6]. They showed that

UGAP = { <s, s, G> | G is an undirected graph with a path from node s to node g }

and several equivalent problems are in NL but they could neither show them to be in L or hard for NL. By Theorem 4.4 it is possible that a set in NL not be hard for L. This is not the case for UGAP since:

Theorem 2.2. UGAP is hard for L.

Proof. We use the above reduction of a set in L to GAP1 but regard the edges as undirected to obtain a reduction to UGAP. If there is an undirected path from s to g, then the path will have been "directed" in the original reduction. If not, let vj be the first node along the path whose next edge is backwards (see Fig. 2). Let vk be the next node along the path whose next edge is forwards. Such a node exists since g (in the reduction) has no outedges. Then vk has outdegree 2, contradicting the construction of the 1-L reduction.#

Finally, we recall that

GAP = { <s, s, G> | G is a directed graph and node g is reachable from node s }

is complete for NL [6,9]. It is interesting to note the structural sijilarities of the problems GAP1, UGAP and GAP. We have

Theorem 2.3. L = NL if and only if GAP <(1-L) GAP1.

Thus, showing that there are no 1-L reductions of GAP to GAP1 would show that L $\neq$ NL. It should be observed that if a 1-L reduction of GAP to GAP1 exists, then there is a 1-L reduction which, after processing each input symbol, erases the work-tape except for a trach which keeps the count of how many input symbols have been read. These transducers cannot even determine the maximal outdegree of input graphs. Intuitively, it seems impossible that such an emasculated transducer could reduce GAP to GAP1.

We believe that the above result and comments show the importance of 1-L reductions and that they give a new approach to the attempted separation of the classes L, NL, P and NP. We discuss these classes further in Section 3.

3. A Paradigm.

One of the primary problems of computational complexity is the relation of P and NP. Although it is known that

$$L \subseteq NL \subseteq P \subseteq NP,$$

none of the containments are known to be proper. We believe that manipulating the power of reductions might provide a key to answering these questions.

Using P-time reductions to compare the classes P and NP suffered from the defect that the class P was not characterized by complete sets. The consideration of log-tape reductions allowed characterizing P and NL by complete sets [4,5,6], as well as preserving the known complete sets for NP, PTAPE etc.

We are considering the possibility of another defect: that the reductions cannot be distinguished from the recognizing machines. Since it is possible that L=NP, the log-tape reductions do not provide a "lever" to show separation of the feasible classes. The 1-L reductions are an attempt to provide such a lever.

In Section 4 we will give examples of sets in feasible classes that are complete for log-tape (P-time) reductions, but not for 1-L (log-tape) reductions.

The next result uses two different types of reductions to formulate a paradigm for separation of families of languages.

Theorem 3.1. Let A be complete for NP under P-time reductions and let B be complete for P under 1-L reductions. Then P = NP if and only if A is 1-L reducible to B.

The theorem has obvious generalizations to classes such as L and NL.

Proof. If A is 1-L reducible to B and X is an arbitrary set in NP, then X is reducible to P by composing the reductions:

$$X \xmapsto[\text{P-time}]{} A \xmapsto[\text{1-L}]{} B,$$

and running the B recognizer. The other way is obvious.∎

Thus, a strategy to show P ≠ NP could use the power of P-time reductions to find a clever NP complete set that defeats any 1-L reduction to a set that is 1-L complete for P.

4. Non-Existence Results for 1-L reductions.

In this section we use combinatorial arguments and recursive function theory to prove that certain 1-L reductions do not exist. Theorem 4.1. shows hy a combinatorial argument that a classical reduction of CNF-SAT to CLIQUE cannot be 1-L; but that other reductions exist that are 1-L. Theorems 4.2. and 4.3. show

1-L (log-tape) reductions are strictly weaker than log-tape (P-time) reductions. Finally, Theorem 4.4. strengthens a result of Ladner's [8].

We recall here the canonical reduction of CNF-SAT to CLIQUE [1,2,7]. Let a boolean formula F be given in conjunctive normal form:

$$F = F1 \ \& \ F2 \ \& \ \ldots \ \& \ Fq,$$

where each Fi is a disjunction of boolean letters and their negations:

$$Fi = Ti1 + Ti2 + \ldots + Tij.$$

Then F is reduced to the graph $G = \langle V, E\rangle$ where

$V = \{ (Fi, Tij) \mid Tij \text{ occurs in } Fi \}$ and

$\{ \langle (Fi, Tij), (Fk, Tkm)\rangle \mid i = k \text{ and } Tij \neq Tkm \}$

Then F is satisfiable if and only if G has a q-clique.

Theorem 4.1. 1) No 1-L reduction can map all CNF formulas to a graph isomorphic to the graph defined above.

2) There is a 1-L reduction of CNF-SAT to CLIQUE.

Proof. 1) We show for an arbitrary 1-L reducer R that we can construct formulas that will be reduced incorrectly. We will consider formulas with an even number, q = 2n, of factors, distinct variables within each factor, and two or more terms of each factor generate a maximal totally disconnected subgraph which we will refer to as an S-node. We say that an S-edge exists between two S-nodes if some potential edge is misssing; i.e. if a term appearing in one of the node's factor appears negated in the other node's factor. See Figure 4.

Consider halting the reducer after it has read n factors. At this point the configuration of R must "remember" enough about F1, ..., Fn to construct the canonical reduction.

Figure 5. shows a scheme for constructing non-isomorphic bipartite graphs. The 2n dark edges will be in every graph. Each subset of {E1,...,En-3} gives a graph non-isomorphic to any other such graph. So with 2n nodes we get at least

2^{n-3}

non-isomorphic bipartite graphs.

To realize a particular graph by a canonically reduced formula, we first let Fn+j = (Vn+j). If an S-edge is to exist between Fn+j and Fi, we let Vn+j occur in Fi. Finally we pad each factor to at least two terms with new distinct variables. The formula so constructed will be reduced canonically to the desired S-graph.

For sufficiently large n, this construction gives at least two subformulas F1 & ... & Fn and G1 &...& Gn which yield the same configuration of the reducer R. Then

F1 &...& Fn & Fn+1 &...& F2n

and

G1 &...& Gn & Gn+1 &...& G2n

will produce the same S-graph by 2 but different S graphs under the canonical reduction.

2) We note that Galil's reduction [3] is a 1-L reduction of NP to CLIQUE, however we present here a modification of the above reduction which is 1-L. Let T1,...,Tk be a list of all the atomic boolean letters and their negations that occur in F and assume they are numbered consecutively from

1. Define $G' = \langle V', E' \rangle$ by putting

$$V' = \{ (Fi, Tj) \mid 1 <= i <= q \text{ and } 1 <= j <= k \} \cup \{V\}$$
$$1 <= j <= k \} \quad \cup \{V\}$$

where V is a new point, and putting

$$E' = \{ \langle (Fi, Tj), (Fk, Tm) \rangle \mid i \neq k \text{ and } Tj \neq TM\}$$
$$\cup \{ \langle (Fi, Tj), V \rangle \mid Tj \text{ occurs in } Fi \}.$$

See Fig. 3. We claim that G' has a clique of size q+1 if and only if F is satisfiable. To see that this reduction is 1-L, simply note that the edges connected to V are generated in a single reading of the input. The other edges are then enumerated using a count of the terms.#

It is startling that all of the natural complete sets we have encountered are still complete under 1-L reductions. This need not have been the case.

Theorem 4.2. There exist sets in P, NP etc. that are complete under log-tape reductions but are not complete under 1-L reductions.

Proof. This is proved in much the same way that one shows

$$\{w\#w \mid w \text{ is a word over } \{0,1\}\}$$

is not recognizable by a 1-L machine. Let U be any NP complete set under log-tape reductions. But

$$T = \{u\#u \mid u \text{ is in } U\}.$$

Then T is not complete for NP under 1-L reductions. Indeed we show that if

$$S = \{(x,y,z,a) \mid x,y,z \text{ are in } \{0,1\}^* \text{ and } (a = 0 \text{ and } x = y) \text{ or } (a = 1 \text{ and } y = z)\},$$

then S is not 1-L reducible to T. Assume for the sake of a contradiction that R is a 1-L machine which reduces S to T. Let R have k symbols and r states. Choose n so that

$$2^n > rn^{k+1}.$$

Then there are distinct words x1 and x2 of length n such that just after reading x1 or x2, R is in the same configuration. Suppose that in the two cases it has already printed w1 and w2. Then w1 $\neq$ w2; otherwise R would give the same result for (x1, x1, 0, 0) and (x2, x1, 0, 0). Similarly, it is not the case that w1 = bw#b and w2 = cw#c.

Now consider what R does on (x1, 0, 0, 1) and (x2, 0, 0, 1). Since after reading the first ',' it will be in identical configurations, R(x1, 0, 0, 1) = w1z and R(x2, 0, 0, 1) = w2z for some z. Also both of these are in T. So either the '#' appears in w and w1 = w2 or w1 = bw#b and w2 = cw#c. Both of these were ruled out above. So S is not 1-L reducible to T.#

Theorem 4.3. If P $\neq$ PTAPE (or equivalently there is a set S $\subseteq$ a* with S in P but not in L), then there are sets in PTAPE which are complete under P-time reductions but not under log-tape reductions.

Proof. Let

$$U = \{M\#w\#0^m \mid M \text{ accepts } w \text{ in tape } m\}$$

be the universal PTAPE complete set. Given S $\subseteq$ a* as above, define

$$f(x) = i_1,\ldots,i_{|x|}$$

where

$$i_j = \begin{cases} 0 \text{ is } j \text{ is in } S \\ 1 \text{ otherwise.} \end{cases}$$

Note that f may be computed in P-time but not in log-tape. Put

$T = \{u\#f(u) \mid u \text{ is in } U\}$.

Then T is complete for PTAPE under P-time reductions but not under log-tape reductions. The proof of this uses the recursion theorem fo find a way to compute f given any reduction of U to T. Assume for the sake of a contradiction that R is a log-tape machine which reduces U to T.

Define the recursive function g from Turing Machines to Turing Machines as follows. Choose r such that the function f is in

$$\text{Time }(n^r).$$

For any machine M, g(M) is the machine in

$$\text{Space }(2n^r)$$

which on input w does the following:

Step 1. Simulate R on input

$$M\#w\#0^{2|w|^r}$$

This can be done in the available space by keeping a counter for R's head position in the field of 0's. When R would begin to output f(u), g(M) checks that the first |w| digits printed by R are correct. (This involves computing f(w) which may be done on the available tape.)

If any of the following conditions hold, then g(M) accepts w:

a) R's output is not of the form

$$A\#B\#0^k\#C$$

with

$$|C| = |A\#B\#0^k|,$$

i.e. in the form of T.

b) R's output is in correct form, but $|C| \geq |w|$, or

c) R's output is in correct form, but

$$C \neq f(A\#B\#0^k).$$

Otherwise, go to step 2.

Step 2. In this case R's output is $u\#f(u)$ where

$$u = A\#B\#0^k.$$

Also, $|u| < |w|$. So in $|w|$ space $g(M)$ can simulate A and check whether u is in U. If so then $g(M)$ rejects w; otherwise it accepts.

By the recursion theorem there is an M with $g(M) = M$. Let

$$s(w) = g(M)\#w\#0^{2|w|^r}.$$

Then by assumption $g(M)$ accepts w if and only if $s(w)$ is in U if and only if $H(s(w))$ is in T. Let $H(s(w)) = u\#C$. By the definition of g and the above equations, $|C| >= |w|$ and the first $|w|$ digits of C are $f(w)$. We constructed g so that if that were not the case, then $g(M)$ accepts w if and only if M does not. Thus to compute $f(w)$ in log-space we can simulate it on input $s(w)$ and output the first $|w|$ digits after the third '#'.

This contradiction shows that there is no log-tape reduction of U to T.#

Ladner [8] has shown that if $P \neq NP$, then there are sets in NP - P that are not complete for NP. We extend Ladner's result under 1-L reductions. (Or assuming $P \neq L$, we could use log-tape reductions for part a.)

<u>Theorem 4.4</u>. a) Assuming $P \neq NP$, then there are sets in NP-P that are not hard for P (and thus not NP-complete). b) Assuming $L \neq NL$, then there are sets in NL - L that are not hard for L.

Proof. We show a); b) is similar. Let U be complete for P under 1-L reductions, let B be in NP - P and let Ri be an enumeration of 1-L reducers. We construct a set A so that

1) A is in NP,

2) U is not 1-L reducible to A, and

3) A is not 1-L reducible to U,

thus proving the theorem. A is constructed in stages so that at stage i we acquire witnesses to prove that Ri neither reduces A to U nor U to A. The machine M of Algorithm 4.5. accepts the set A.

By application of the recursion theorem M is well defined. Note that non-determinism is used at lines 1, 3 and 5. The simulations at lines 2, 4 and 6 are deterministic. The simulation at line 2 sets the variables i and s to be used in later steps.

```
     IF x = 0
     THEN DO i :=0; s :=0;
1      accept x iff x in B;
       RETURN END;
2    with |x| moves simulate M on w = 0, 1, ... ;
     {i and s are set by last completed simulation}
     IF no simulation completes
     THEN DO i :=0; s :=0;
3      accept x iff x in B;
       RETURN END;
     IF s = 0
     THEN DO
       {set witness to show A ⋠ (1-L) U by Ri}
```

```
4       with |x| moves search for w = 0,1,... so that w is in
        A iff Ri(w) is not in U;
        IF found
5       THEN s :=1;
        accept x iff x in B;
        END
     ELSE DO
        {set witness so that U ≰ (1-L) A by Ri}
6       with |x| moves search for w=0,1,... so that w is in
        U iff Ri(w) is not in A;
        IF found
        THEN DO s :=0; i :=1+1; END;
        reject x
        END
```

Algorithm 4.5. The Machine M on input x.

At the i-th stage with s equal 0, a witness to "A is not 1-L reducible to U by Ri" is sought. To see that the search eventually succeeds, assume, to the contrary that it does not. Then the sets A and B are equal except for an initial fragment. Now A and hence B would be reducible to U. This is absurd, so the search succeeds and s is then set to 1.

With s equal 1, a witness to "U is not 1-L reducible to A by Ri" is needed. If no witness were to be found, then A would be finite. This implies that all sets in P are recognizalbe by 1-L automata which is absurd. So the witness will be found; then i is incremented and s is reset to 0.

Finally, A as constructed is in NP since its nondeterministic recognition time is bounded by $c|x|$ + time to decide if x is in B.∎

Note that theorem 2.2. shows that UGAP is not such a set. It is still possible that UGAP is in NP - L thus satisfying Ladner's theorem.

5. Conclusions and Further Research.

We have demonstrated here that the 1-L reductions are provably weaker than the feasible complexity classes, give essentially the same complete sets as were known under other reductions, and permit various techniques not applicable to more powerful reductions to prove the non-existence of reductions.

Three general areas of research are suggested by this approach

1) A deeper investigation of the power of 1-L reductions in an attempt to prove inequalities among L, NL, P, NP, PTAPE etc. This approach is supported by the view that one-way devices are easier to analyze than are computations in the feasible classes and thus known techniques may find application in their study. We conjecture that there does not exist a 1-L reduction of GAP to GAP1.

2) Use 1-L reductions to investigate the relations between the classes PTAPE and EXPTIME.

3) Develop other weak reductions with which to separate complexity classes.

REFERENCES

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

2. Cook, S.A., "The Complexity of Theorem Proving Procedures," Proceedings 3rd Annual ACM Symposium on Theory of Computing, May 1971, 151-158.

3. Galil, Z., "Some Direct Encodings of NOndeterministic Turing Machines Operating in Polynomial Time into NP-Complete Problems," SIGACT NEWS No. 1, January 1974, 19-24.

4. Jones, N.D., "Space-Bounded Reducibility Among Combinatorial Problems," J. of Computer and System Sciences, Vol. 11, (1975), 68-85.

5. Jones, N.D., and W.T. Laaser, "Problems Complete for Deterministic Polynomial Time," Theoretical Computer Science, Vol. 3, (1077) 105-117.

6. Jones, N.D. Y.E. Lien and W.T. Laaser, "New Problems Complete for Log Space," Mathematical Systems Theory, Vol. 10, (1976) 1-17.

7. Karp, R.M., "Reducibility Among Combinatorial Problems," In Complexity of Computer Computations, edited by R.E. Miller and J. Thatcher, Plenum Press, New York, 1972, 85-104.

8. Ladner, R.E., "Polynomial Time Reducibility," Proceedings 5th Annual ACM Symposium on Theory of Computing, April 1973, 122-129.

9. Savitch, W.J., "Relations Between Nondeterministic and Deterministic Tape Complexities," J. of Computer and System Sciences, Vol. 4, (1970), 122-129.