ON ISOMORPHISMS AND DENSITY OF NP AND OTHER COMPLETE SETS*

J. Hartmanis and L. Berman

TR 75-260

October 1975

Department of Computer Science
Cornell University
Ithaca, New York 14853

*This research has been supported in part by National Science Foundation Research Grants GJ-33171X and DCR 75-09433.

ON ISOMORPHISMS AND DENSITY OF NP AND OTHER COMPLETE SETS*

J. Hartmanis and L. Berman
Department of Computer Science
Cornell University
Ithaca, N.Y. 14853

Abstract

If all NP complete sets are isomorphic under deterministic polynomial time mappings (p-isomorphic) then P $\neq$ NP and if all PTAPE complete sets are p-isomorphic then P $\neq$ PTAPE. We show that all NP complete sets known (in the literature) are indeed p-isomorphic and so are the known PTAPE complete sets. Thus showing that, inspite of the radically different origins and attempted simplification of these sets, all the known NP complete sets are identical but for polynomially time bounded permutations.

Furthermore, if all NP complete sets are p-isomorphic then they all must have similar densities and, for example, no language over a single letter alphabet can be NP complete, nor can any sparse language over an arbitrary alphabet be NP complete. We show that complete sets in EXPTIME and EXPTAPE cannot be sparse and therefore they cannot be over a single letter alphabet. Similarly, we show that the hardest context-sensitive languages connot be sparse. We

---

* This research has been supported in part by National Science Foundation Research Grants GJ-33171X and DCR 75-09433.

also relate the existance of sparse complete sets to the existance of simple combinatorial circuits for the corresponding truncated recognition problem of these languages.

I. Introduction

During the past years the importance of the P = NP? problem has been fully realized and today it is one of the most important problems in theoretical computer science [C,AHU,K,HS,SI]. The importance of the P = NP? problem derives from the fact that NP, the family of languages accepted by nondeterministic Turing machines in polynomial time, contains complete problems to which all other problems in NP can be easily reduced and from the fact that very many problems of practical interest in computing are in NP and many of them are NP complete [AHU,C,K,GJS,SA,U]. Thus the search for fast algorithms for a bewildering variety of problems can be reduced to the search for a fast algorithm of a single problem. As a matter of fact, during the last years considerable effort has been expanded in discovering new NP complete problems and it is quite impressive how many diverse problems from many different problem areas have turned out to be NP complete [AHU,C,K,SA,U]. Furthermore, among the known NP complete problems some have been simplified and found still to be NP complete [GJS].

In this paper we show that regardless of their origins and attempted simplifications, all the "known" NP complete sets are essentially the same set. More specifically, we prove that all the known NP complete sets are isomorphic under a deterministic polynomial time mappings. Thus these NP complete sets, except for a deterministic polynomial time recoding, are identical. The proof of this result follows from two technical lemmas which give necessary and sufficient conditions that a set is isomorphic under polynomial time mappings to a given NP complete set, say the set of all satisfiable Boolean functions in conjunctive normal form. To establish polynomial time isomorphisms (p-isomorphism) between NP complete sets we just have to check that these sets satisfy the sufficient conditions of our lemmas, which turn out to be easy to verify for all the NP complete sets found in the literature. We exhibit the proof of the existence of p-isomorphism for the best known NP complete problems and they can be easily supplied for the other NP complete problems which have been described up to date in the literature. Since so far no NP complete problems have been found which are not p-isomorphic and since all attempts to construct such sets have failed, we are forced to conjecture that all NP complete sets are isomorphic under deterministic polynomial time mappings.

It should be observed that a proof of this conjecture implies that $P \neq NP$. To see this, we just have to note that $P = NP$ iff every non-empty finite set is NP complete. Since finite sets cannot be isomorphic to infinite sets, the isomorphism of all NP complete sets implies that $P \neq NP$. As a matter of fact, $P \neq NP$ iff all NP complete sets are isomorphic under recursive mappings.

It still could happen that $P \neq NP$ but that there exist NP complete sets which are not p-isomorphic. We conjecture that this is not the case.

By the same methods we also show that all the known PTAPE complete sets are isomorphic under deterministic polynomial time mappings. Furthermore, if all PTAPE complete sets are p-isomorphic then P $\neq$ PTAPE, since P = PTAPE iff every nonempty finite set is PTAPE complete.

Next we look at the density of NP and PTAPE complete sets. We say that a set A, A $\subseteq \Sigma^*$, is p-sparse iff the number of elements in A up to length n is bounded by a polynomial in n. It is easily seen that the known NP and PTAPE complete sets are not p-sparse and that they cannot be p-isomorphic to p-sparse sets. We suspect that neither NP nor PTAPE complete sets can be p-sparse. Note that a proof that p-sparse sets cannot be NP nor PTAPE complete would prove that

$$P \neq NP \neq PTAPE.$$

On the other hand, we show that p-sparse sets cannot be complete in EXPTIME and EXPTAPE, as first observed by A. Meyer [M]. Our proof actually shows that in EXPTIME there exist sets which are not p-sparse and whose reduction to another set must be one-one almost everywhere, thus, no p-sparse set can be EXPTIME complete. The corresponding result also holds for EXPTAPE and more complex time and tape bounded families of languages. It is still an open problem whether the EXPTIME and EXPTAPE complete sets are all p-isomorphic, respectively.

It should be observed that the existence of p-sparse complete sets for NP or PTAPE would imply the existence of combinatorial circuits of polynomial complexity for the solution of the corresponding truncated recognition problems. Equivalently, the existence of p-sparse complete sets for NP (or PTAPE) implies that we could prepare a tape (table) growing only polynomially in n such that all NP (or PTAPE) problems could be solved in deterministic polynomial time using a fixed tape (for table-look up). Thus the existence of sparse NP complete sets would permit, for all practical purposes, the recognition of NP sets in deterministic polynomial time (using a precomputed, polynomially long tape segment). This seems to be quite unlikely, and the above mentioned results show that this is not the case for EXPTIME and EXPTAPE: there does not exist any sparse set to which complete problems in EXPTIME and EXPTAPE can be reduced in polynomial time.

Finally, we turn to context-sensitive languages. We say (following R. Book) that a context-sensitive language L is <u>hardest</u> if every other context-sensitive language can be reduced to L by a linear-time mapping. It is well known that hardest context-sensitive languages exist [HH] and that hardest context-free languages also exist [GR]. Clearly, the context-sensitive languages are contained in P or NP iff a hardest csl is in P or NP, respectively. Similarly, the deterministic context-sensitive languages are equal to the non-deterministic context-sensitive languages iff a hardest csl is a deterministic csl. We prove that no p-sparse language can be a hardest csl and show that all known hardest csl's are p-isomorphic. These results easily generalize to hardest languages of other families of tape bounded languages.

II. Preliminaries

In this section, we make precise some of the objects which we will treat. Our terminology is reasonably standard and so, this section may be skipped by those familiar with the terminology of complexity theory.

Definitions:

A transducer is a deterministic three tape Turing machine with one two-way read-only input tape, one two-way read-write work tape, and one one-way write-only output tape.

Our acceptor will be a k-tape Turing machine. The input will be written on one of the tapes and all tapes are two-way read-write. Acceptance will be indicated by entering a final state and halting. If the machine has just one tape we call it a single tape Turing machine, otherwise, it is called a multi-tape Turing machine. If the next move function associated with the Turing machine is single-valued, we call it deterministic, otherwise, it is called non-deterministic. We note that a deterministic TM may be considered to be non-deterministic in a trivial fashion.

The amount of time used by a TM on input x is the number of steps in the shortest accepting computation if x is accepted; the number of steps in the longest computation if x is not accepted (if some computation does not halt it is undefined).

The amount of tape used by a TM is the smallest amount of tape used by an accepting computation if x is accepted, or the largest amount used by any computation if x is not accepted (again, if some computation uses unbounded tape, it is undefined.)

A TM, M, runs in time (tape) t(n) for some function t(n) if for all $n \geq 0$ for every x of length n M uses less than t(n) time (tape) on input x.

$$(N)DTIME[t(n)] = \{A \mid A \text{ is accepted by a (non-)deterministic TM which runs in time } t(n)\}.$$

$$(N)DTAPE[t(n)] = \{A \mid A \text{ is accepted by a (non-)deterministic TM which runs on tape } t(n)\}.$$

$$P = \bigcup_{i \geq 0} DTIME(n^i)$$

$$NP = \bigcup_{i \geq 0} NDTIME(n^i)$$

$$PTAPE = \bigcup_{i \geq 0} DTAPE(n^i) = \bigcup_{i \geq 0} NDTAPE(n^i) = NPTAPE$$

$$(N)DEXP\text{-}TIME = \bigcup_{i \geq 0} (N)DTIME(2^{in})$$

$$DEXP\text{-}TAPE = \bigcup_{i \geq 0} DTAPE(2^{in}).$$

A transducer, T, is said to be polynomial time bounded if there is some polynomial p(n) so that T, when considered as a multi-tape TM runs in time p(n).

A transducer, T, is said to be a linear time transducer if there is some constant, $c > 0$, so that T, when considered as a multi-tape TM, runs in time cn.

A set $A \subseteq \Sigma^*$ is said to be reducible to a set $B \subseteq \Gamma^*$ if there is some transducer T such that $T: \Sigma^* \to \Gamma^*$ and $T(x) \in B$ iff $x \in A$. A is said to be reducible to B in polynomial time (p-reducible) if the transducer T runs in polynomial time. Similarly if T runs in linear time A is said to be linearly reducible to B.

A set, B, is C-hard for some class of sets C (e.g. NP or NTAPE(n)) if for every $A \in C$, A is p-reducible to B.

A set, B, is complete for C if it is C-hard and $B \in C$.

A set, B, is C-hardest if B is in C and every A in C is linearly reducible to B. For example, hardest languages exist for the families of context-free languages, context-sensitive languages, deterministic context-sensitive languages, etc. but not for NP or PTAPE.

We say that a set A, $A \subseteq \Sigma^*$, is p-sparse iff there exists a polynomial p(n) such that

$$|\{w \mid w \in A, |w| \leq n\}| \leq p(n).$$

III. Polynomial Time Isomorphisms

In this section we investigate polynomial time isomorphisms between languages.

We say that A and B are p-isomorphic iff there exist surjection $f: \Sigma^* \to \Gamma^*$ such that f is a p-reduction of A to B and f^{-1} is a p-reduction of B to A.

We now prove a polynomial time bounded equivalent of the Cantor-Bernstein-Myhill Theorem.

Theorem 1: Let p and q be length increasing invertible p-reduction of A to B and B to A, respectively. Then A and B are p-isomorphic.

Proof: From p and q we will construct a surjection ϕ such that ϕ and ϕ^{-1} are p-time computable and

$$w \in A \text{ iff } \phi(w) \in B.$$

We note that

$$\Sigma^* = R_1 \cup R_2 \text{ and } \Gamma^* = S_1 \cup S_2$$

with

$$R_1 = \{(q \circ p)^k x \mid k \geq 0 \text{ and } x \neq q(y)\},$$

$$R_2 = \{q \circ (p \circ q)^k x \mid k \geq 0 \text{ and } x \neq p(y)\},$$

$$S_1 = \{(p \circ q)^k x \mid k \geq 0 \text{ and } x \neq p(y)\},$$

$$S_2 = \{p \circ (q \circ p)^k x \mid k \geq 0 \text{ and } x \neq q(y)\}.$$

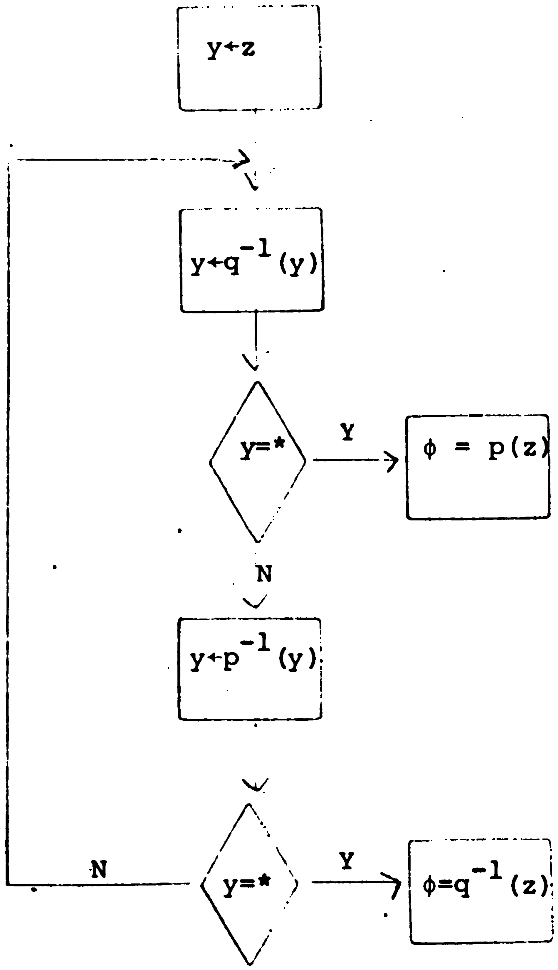
Let $s(n)$ be a polynomial such that p, q, p^{-1} and q^{-1} can all be computed by deterministic TM's within $s(n)$ steps for inputs of length n. We assume that p^{-1} and q^{-1} both output a special symbol, $*$, if they are undefined. This is permissible since they are polynomial time bounded. ϕ and ϕ^{-1} will be computed by the following

$$\phi(z) = \begin{cases} p(z) \text{ if } z \in R_1 \\ q^{-1}(z) \text{ if } z \in R_2, \end{cases}$$

$$\phi^{-1}(z) = \begin{cases} p^{-1}(z) \text{ if } z \in S_2 \\ q(z) \text{ if } z \in S_1. \end{cases}$$

First notice that ϕ maps R_1 onto S_2 and R_2 onto S_1 and in fact ϕ and ϕ^{-1} are inverses.

We will now describe a transducer, T, which computes ϕ and is polynomial time bounded. We describe T by means of the following flowchart:



Flowchart Computing $T(z) = \phi(z)$.

As p and q are both length increasing p^{-1} and q^{-1} are length decreasing and so T need cycle thru the loop at most $\frac{|z|}{2}$ times. At most $(|z| + 1)$ evaluations of p^{-1}, q^{-1}, or p are therefore required and so T runs in time at most $(n+2)\, s(n)$ which is a polynomial.

We note that identical considerations show that ϕ^{-1} is also p-time bounded.

Corollary 2: If p, q, p^{-1}, q^{-1} of Theorem 1 are computable in linear time then ϕ and ϕ^{-1} are computable in n^2-time.

Proof: Previous proof carries through.

In order to simplify the application of Theorem 1 we now establish two technical results which can easily be applied to show that many complete sets are p-isomorphic. We first define padding functions and show that if either set A or B of Theorem 1 (or Corollary 2) have padding functions satisfying some simple hypotheses, then we can remove the length increasing restrictions from the hypothesis of these results.

Definition: Let $A \subseteq \Sigma^*$. Then $S_A: \Sigma^* \rightarrow \Sigma^*$ is a padding function for a set A if it satisfies the following two properties:

1. $S_A(x) \in$ iff $x \in A$
2. S_A is invertible (i.e. one-one).

We say that a padding function, S_A, has time complexity $t(n)$ if both S_A and S_A^{-1} may be computed by deterministic Tm's in time $t(n)$.

Lemma 3: Let f be a one-one, p-time reduction of A to B and let f^{-1} also be computable in p-time. Assume also that either A or B has a padding function S_X (X=A,B) which satisfies conditions:

1. S_X has polynomial time complexity $s(n)$.
2. $\forall y \; |S_X(y)| > |y|^2 + 1$.

Then there exists a reduction f' of A to B which is one-one, p-time, <u>length</u> <u>increasing</u> and has $(f')^{-1}$ computable in p-time.

<u>Lemma 4</u>: If f, f^{-1} have linear time complexity; S_X has linear time complexity and condition 2 of Lemma 3 is replaced by

2'. $\forall y \; |S_X(y)| > 2|y| + 1$

then f' of Lemma 3 exists and has linear time complexity.

<u>Proof</u>: Let X=A and let f, f^{-1} be computable in polynomial time. Let $q(n)$ be a polynomial time bound in which f and f^{-1} can be computed. Then, by condition 2 on the padding function we know there exists an integer r such that for all x $|S_A^r(x)| > q(|x|)$ therefore it follows that $|f \circ S_A^r(x)| > |x|$, since if $|f \circ S_A^r(x)| \leq |x|$ then (as f^{-1} can output at most one digit per move) $|f^{-1} \circ f \circ S_A^r(x)| \leq q(|x|)$, which is a contradiction. So define $f' = f \circ S_A^r$, by the reasoning given above, f' is length increasing and clearly satisfies the other requirements of Lemma 3.

For X=B and f, f^{-1} polynomial computable we again know that there exists an integer r such that for all x $|S_B^r(x)| > q(|x|)$ and also as above $q(|f(x)|) > |x|$. Let $f' = S_B^r \circ f$ then $|f'(x)| = |S_B^r(f(x))| > q(|f(x)|) > |x|$ as needed.

For the linear time bounds a more careful time analysis yields the desired proof.

The primary difficulty in applying Theorem 1 is now seen to be verifying that a given reduction can be inverted in polynomial time.

It is seen that the existence of an invertible reduction depends solely on the richness of the structure of the target set.

Lemma 5: Let A be a set for which two p-time computable functions $S_A(-,-)$ and $D_A(-)$ exist with the following properties:

1. $\forall x,y \quad S_A(x,y) \in A$ iff $x \in A$
2. $\forall x,y \quad D_A(S_A(x,y)) = y$.

Then if f is any p-time reduction of C to A, the map $f'(x) = S_A(f(x),x)$ is one-one and invertible in p-time.

Proof: Assume $f'(x) = f'(y)$. Then

$$x = D(f'(x)) = D(S_A(f(x),x)) = D(f'(y)) = D(S_A(f(y),y)) = y$$

so f' is one-one. If we define

$$q(x) = \underline{\text{if}} \;\; x = S_A(f(D(x)),D(x)) \;\; \underline{\text{then}} \;\; D(x) \;\; \underline{\text{else}} \;\; *$$

we see that $q(f'(x)) = x$ so $q = (f')^{-1}$ and a straightforward time analysis shows f' and q are both p-time computable.

Lemma 6: If S_A, D, and f of Lemma 5 are all linear time computable the so is f'.

Proof: Straightforward.

We now define a number of known NP-complete problems:

1. UNIV - $\{CODE(x_1x_2...x_n) \# M_i \# ^{3|M_i|t} \mid M_i$ accepts $x_1...x_n$ in t steps$\}$, where $CODE(x_1x_2...x_n)$ is a simple digit by digit encoding of $x_1x_2...x_n$ so that $|CODE(x)|=|M_i|$ [HH].

2. CNF_SAT - given an encoding of a boolean expression in conjunctive normal form, is there some assignment of truth values to the variables which gives the expression the value true [C].

3. INEQ[0,1,),(,+,•] - given an encoding of two regular expressions over 0,1,),(,+,• ; do they represent different sets [MS].

4. CLIQUE - given an encoding of an undirected graph and an integer k, is there a subset of k-mutually adjacent nodes [K].

5. HAMILTON CIRCUIT - given an encoding of a directed graph is there a cycle including all nodes which does not intersect it self [K].

We now apply our results to prove that large numbers of NP-complete and PSPACE complete problems are p-time isomorphic. We stress that we know of no complete problems for either class which could not easily have been added to the appropriate lists. Furthermore, all our attempts to construct such problems have failed.

Theorem 7: The following NP-complete problems are p-time isomorphic:

1. UNIV,
2. CNF_SAT,
3. INEQ[0,1,),(,+,•],
4. CLIQUE,
5. HAMILTON CIRCUIT.

Proof: We first show that CNF_SAT has a padding function satisfying Lemma 3 and functions $S_A(-,-)$ and $D_A(-)$ satisfying Lemma 5. Then any set satisfying Lemma 5 will automatically be p-isomorphic to CNF_SAT since Lemma 3 will guarantee all reductions can be taken length increasing and Lemma 5 will show they are all one-one and invertible.

Consider the function $S_A(w,y)$, which is computed as follows:

It examines w to determine if w is a Boolean formula, B, in CNF. If not r=0. If yes, let $x_1, \ldots, x_r$ be variables appearing in B (or at least including every variable in B). (The value of r can be determined in p-time.)

$S_A(w,y) = w \wedge (x_{r+1} \vee \neg x_{r+1}) \wedge z_1 \wedge z_2 \wedge \cdots \wedge z_n$ where z_j is the literal $z_j = \begin{cases} x_{r+1+j} & \text{if } y(j)=1 \\ \neg x_{r+1+j} & \text{if } y(j)=0 \end{cases}$. Thus, w is satisfiable iff $S_A(w,y)$ is satisfiable. $S_A(-,-)$ is clearly p-time computable and a function D_A which examines a string to determine if it has a suffix of the proper form and if so translates it appropriately will also be in p-time. $S_A(-,-)$ and $D_A(-)$ together satisfy Lemma 5. The padding function needed for Lemma 3 is defined by $S(w) = S_A(w, 0^{|w|^2+1})$, which clearly satisfies Lemma 3.

We now show INEQ satisfies Lemma 5 and so is p-time isomorphic to CNF_SAT : $S_A(w,y)$ first checks that $w = R_1 \# R_2$. If so it outputs $(R_1 + 0^n 1y)\#(R_2 + 0^n 1y)$ [where $n = |R_1 \# R_2|$] if not it outputs w##y. The obvious D_A works.

UNIV - $S_A(w,y)$ encodes y in inaccessible new states of M_i and adjusts #'s at end to accomodate new states.

CLIQUE : $S_A(w,y)$ checks that w has format k# G, where G is the encoding of some graph, determins highest labelled vertex, r, used in G. G has vertices $v_1, v_2, \ldots, v_r$. S_A outputs (k+1)#G' where G' has vertices $v_1, v_2, \ldots, v_r, v_{r+1}, v_{r+2}, \ldots, v_{r+2|y|}$ where G' has the edges of G plus $\forall j \leq r$, $\forall i \leq |y|$ G' contains edges (v_j, v_{r+2i}) if $y(i) = 1$ and (v_j, v_{r+2i-1}) if $y(i)=0$. This S_A and the obvious D_A work.

Hamilton circuit: $S_A(w,y)$ checks that format is correct and inserts vertices where r is the highest numbered vertex in G.

$v_{r+1}, v_{r+2}, v_{r+3}, \ldots, v_{r+3|y|+1}$, an edge from $v_r \to v_{r+1}$ edges and $\forall j \leq |y|$

$(v_{r+3(j-1)+k}, v_{r+3j+1})$ for $k=2,3$

$(v_{r+3(j-1)+2}, v_{r+3(j-1)+3})$

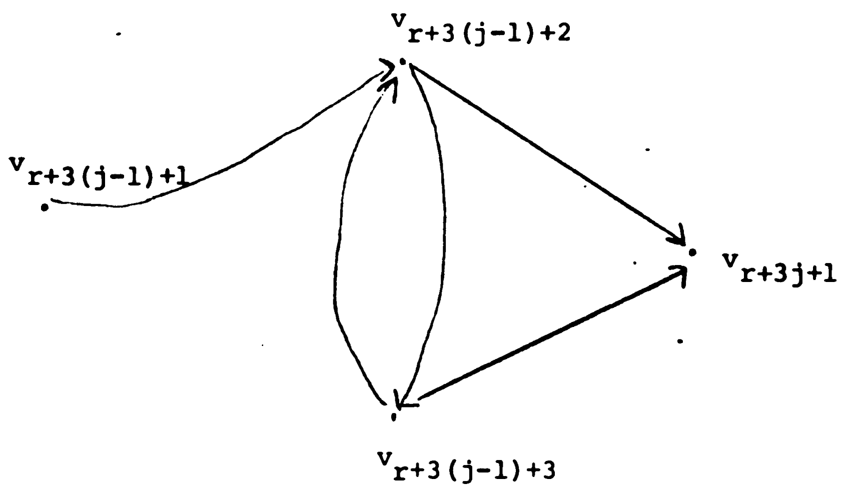
$(v_{r+3(j-1)+3}, v_{r+3(j-1)+2})$

and if $y(j)=1$ $(v_{r+3(j-1)+1}, v_{r+3(j-1)+2})$

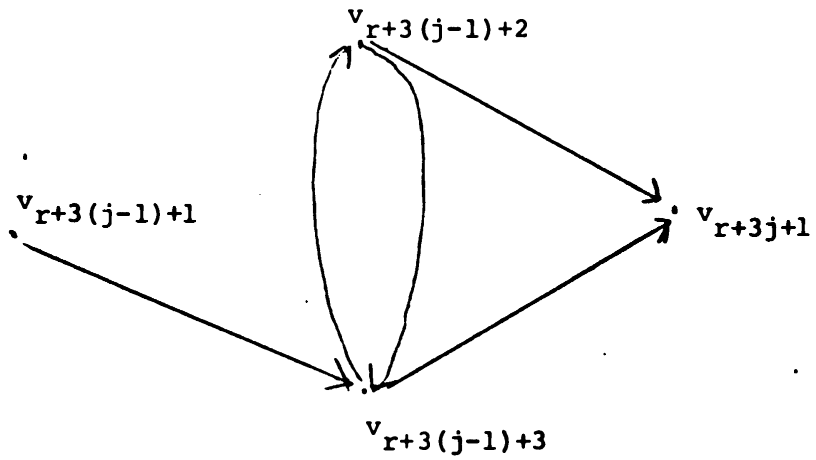
if $y(j)=0$ $(v_{r+3(j-1)+1}, v_{r+3(j-1)+3})$

and if $(v_r, v_i) \in G$ put $(v_{r+3|y|+1}, v_i) \in G'$.

Again the obvious D_A map also works.



SECTION OF GRAPH INDICATING $y(j)=1$



SECTION OF GRAPH INDICATING $y(j)=0$

ENCODINGS IN HAMILTON CIRCUIT

We note that in other known NP problems it is possible to encode the necessary information in a manner not affecting whether a given string is in the language. We note specifically that this technique shows the "simplified" NP-complete problems of Johnson, Stockmeyer, and Garey [GJS] are all p-time isomorphic.

One may argue that our isomorphisms are unnatural, that they were constructed through recursion theoretic techniques which are out of place in discussions of combinatorial problems. We will show, however, that with a little care our results yield not only isomorphisms between the various problems, but in fact, isomorphisms that preserve the underlying combinatorics.

Given a boolean formula in CNF, we may ask how many distinct variable assignments there are which produce a true value for the formula. Similarly, if we were given the encodings of a pair of regular expressions, $R_1 \# R_2$, we might ask how many strings are accepted by one and not the other. For a language L and a fixed $w \in L$, we will call each "piece of information" which evidences $w \in L$ a solution to the (w,L) problem. We use the following notation:

$$Sol(w,L) = \{x | x \text{ encodes a solution to the } (w,L) \text{ problem}\}.$$

$$Sol(w,L) = \phi \quad \text{if } w \notin L.$$

It is, in fact, solutions of the various problems which are of practical importance in computing. We are interested in the elements of Sol(w,L) and not merely whether $|Sol(w,L)| > 0$. It is of little use to a multi-process scheduling algorithm to know that there is a schedule of a given cost; the schedulor must determine the optimal schedule.

Definition: If A and B are NP-complete problems and $f:A \rightarrow B$ is a polynomial time reduction, we say that f is parsimonious if w $|Sol(w,A)| = |Sol(f(w),B)|$.

Parsimonious reductions have been studied before [SI] and it turns out that many of the well known NP-complete problems are related by parsimonious reductions. We feel that a parsimonious reduction should be considered natural since they do not introduce "new" solutions but yield translated problems whose solutions are in one-one correspondence with the solutions of the original problem.

We now state and prove our main result concerning parsimonious reductions:

Theorem 8: Let A be any NP-complete set for which there exist parsimonious p-time reductions $f:A \rightarrow CNF\_SAT$ and $g:CNF\_SAT \rightarrow A$. Let A have functions $S_A(-,-)$ and D_A as in Lemma 5, and furthermore assume that $\forall x \in (0+1)^*$ $S_A(-,x):A \rightarrow A$ is parsimonious; then the isomorphism $\phi:A \rightarrow CNF\_SAT$ guaranteed by Theorem 1 is parsimonious.

Proof: We first note that the composition of parsimonious reductions is parsimonious. Unfortunately, the $S_{CNF\_SAT}(-,-)$ function defined earlier is not parsimonious; however the function $S_{CNF}(w,y) = w \wedge (x_{r+1} \vee x_{r+1}) \wedge z_1 \wedge \dots \wedge z_n$ with z_j as before is parsimonious.

Since $S_{CNF}(-,-)$ and $S_A(-,-)$ are both parsimonious we know, via Lemma 5 and the observation above that the f_* and g_* of Lemma 5 will in fact be parsimonious.

Again S_{CNF} gives us a padding function for CNF_SAT (the function is now parsimonious) which by Lemma 3 tells us the conditions of Theorem 1 are now satisfied. This time, however, all constituents of our isomorphism are parsimonious and since the isomorphism is constructed by application of these reductions, we have that the isomorphism is parsimonious.

We note that the encoding functions of the NP-complete problems INEQ, UNIV, and CLIQUE are all parsimonious, and also that they are each related to CNF_SAT via parsimonious reductions [SI] therefore we have

Theorem 9: The following NP-complete problems are p-time isomorphic via parsimonious mappings:

1) CNF_SAT,
2) INEQ,
3) UNIV,
4) CLIQUE.

Proof: Theorem 8.

We now turn our attention to languages complete for PSPACE.

We again first define a number of PSPACE complete problems:

1. UNIV - $\{M_i \# CODE(x_1 \ldots x_n) \#^{t|M_i|} \mid M_i$ accepts $x_1 \ldots x_n$ on t+n tape squares$\}$ [HH]

2. QBF - Given a quantified boolean formula, e.g. - $\forall x_1 \exists x_2 \forall x_3 (x_1 \vee \neg x_2) \wedge (x_1 \vee x_2 \vee x_3)$, is it true. [MS]

3. HEX - Given a graph and two distinguished vertices a game is defined in which the two players alternately choose vertices. Player 1 wins if he is able to choose vertices which define a path in the graph between the

4. $L_{\Sigma*}$ - $\{R|R$ is a regular expression and $L(R) \neq \Sigma^*\}$ [MS]

Theorem 10: The following PSPACE complete problems are p-time isomorphic:

1. UNIV,
2. QBF,
3. HEX,
4. $L_{\Sigma*}$.

Proof: By the same method used to show CNF could be padded in Corollary 7 we see QBF has padding and $S_A(-,-)$; $D_A(-)$ functions.

S_{UNIV} encodes the second argument in inaccessible states as before

S_{HEX} encodes the second argument in dead end paths

$S_{\Sigma*}(x,y) = (y+(\Lambda+0+1)^n + (0+1)^{n+1}x) \qquad n = |y|$

in all cases the obvious $D(-)$ function works.

We now prove a metatheorem which extends the previous result to tape complete decision problems concerning regular expressions. Define

$$x\backslash L = \{w|xw \in L\} \quad \text{and} \quad L/x = \{w|wx \in L\}.$$

Theorem 11: Let P be any predicate in the regular sets over $\{0,1\}$ such that

1. $P(\{0,1\}^*) = \text{TRUE}$
2. $P_L = \bigcup_{x \in \{0,1\}^*} \{x\backslash L|P(L)=\text{TRUE}\}$ [or $P_R = \bigcup_{x \in \{0,1\}^*} \{L/x|P(L)=\text{T}\}$ is not the set of all regular sets over $\{0,1\}$
3. $L_P = \{R|R$ is a regular expression over $\{0,1\}$ and $P(L(R)) = \text{Fals}$ is in P-TAPE.

Then L_P is p-time isomorphic to $L_{\Sigma*}$.

Proof: Any L_p where P satisfies conditions 1 and 2 above is

In order to show the isomorphism we must find S and D.

Let ϕ be the p-time map such that $R \in L_P$ iff $\phi(R) \in L_{\Sigma *}$ and let L_o be regular set over $\{0,1\}$ not in P_L as in [HH] define $h_o(0) = 00$ and $h_o(1) = 01$. We note that the map

$$R_i \overset{\psi}{\to} h_o(R_i) \cdot 10 \cdot (0+1)^* + (00+01)^* \cdot 10 \cdot R_{L_o} + (00+01)^* \cdot [\Lambda+0+1+11(0+1)^*]$$

has the properties:

1. $R_i \in L_{\Sigma *}$ iff $\psi(R_i) \in L_P$
2. ψ is p-time invertible.

We now define $S_P(x,y) = \psi(S_{\Sigma *}(\phi(x),y))$ we note $D_P(x) = (D_{\Sigma *}(\psi^{-1}(x)))$ is the required D function. Since L_P is PSPACE complete and satisfies Lemma 5 the sets L_P and $L_{\Sigma}*$ are p-time isomorphic, as was to be shown.

IV. Density Considerations

We recall that a proof that no p-sparse set can be NP complete would imply that $P \neq NP$. We cannot solve this problem but we can show that some other complete and hardest sets cannot be p-sparse.

We prove next, as first observed by A. Meyer [M], that complete sets for EXPTIME and EXPSPACE cannot be p-sparse. Furthermore, using a very recent result from [HPV] we show that the hardest context-sensitive languages are not p-sparse. Thus showing that a single letter alphabet language cannot be a hardest csl. We also conjecture that the hardest context-free languages cannot be p-sparse.

Theorem 12: No p-sparse language A can be complete in EXPTIME or EXPTAPE. Thus A _ a* cannot be complete in EXPTIME or EXPTAPE.

Proof: We will prove this result by constructing a set A_o with the following properties:

a) A_o is in EXPTIME
b) A_o is not p-sparse
c) if A_o is p-reduced to a set B by the mapping ρ then ρ is a one-one mapping almost everywhere.

We not describe a TM, M, which accepts A_o. M will be a multitape TM which on input w computes as follows:

1. on one of its tapes M writes down 1#10#11#...#|w|#. Each integer will be treated as the encoding of a transducer, T_i and

$T_i(x)$ will be limited to $|x|^i$ steps so this list will eventually cover all polynomial time bounded transducers. The list can be written down in time $0(w^3)$ so there is some c such that for all $n \geq 1$ the time required to carry out step 1 on input of length n is less than 2^{cn}.

2. for i=1 to |w|
   for each x such that $2^{i+1} \leq x < w$ do
   a. compute $|x|^i$ for $2^{|x|}$ steps.
   b. if computation a. completed then compute $T_i(x)$ for $|x|^i$ simulated steps (call this case I)

      or

      compute $T_i(x)$ for $2^{|x|}$ actual steps (case II) which ever comes first
   c. if case I above occurred store $(x,i,T_i(x))$ on storage tape

(since a., b., and c. are each limited to $2^{|w|}$ steps and they must be carried out at most $w2^{|w|}$ times there is some c' such that 2. can be completed within $2^{c'|w|}$. Also for large |w| case II never occurs)

3. Construct two lists L_1 and L_2 as follows:

for i=1 to $|w|$

find (if it exists) the smallest $x < w$ which satisfies

a) there is some $y < x$ for which $T_i(x) = T_i(y)$

($T_i(x)$ and $T_i(y)$ must be listed in step 2)

b) for all z (x,z) is not on L_2 if such an x is found, take smallest y for which $T_i(x) = T_i(y)$ and put (x,y) on L_1 if no such x is found put i on L_2.

(The length of $T_i(x)$ is less that $2^{|x|}$ and comparisons on a multitape machine can be done in linear time so step 3 can be carried out in time $2^{c''|w|}$ for some c").

4. In ascending order, for each $i \in L_2$

for x=w perform steps 2a. and 2b. If computations complete find smallest $y < w$ for which $T_i(y) = T_i(w)$, determines length of longest chain. $(y,x_1)(x_1,x_2)\ldots(x_{\ell-1},x_\ell)$ entirely on L_1, M accepts w iff $\ell = 2m+1$ for some m if no such y has been found for any i on list L_2 then M accepts w.

This last step can also be carried out in exponential time and so the entire machine has T(M) in DEXPTIME.

It should be clear that for every polynomial time bounded machine T_i there is some integer n_i so that for all x, $|x| > n_i$, the simulation of T_i on x will be completed within $2^{|x|}$ steps and that therefore no p-time reduction, f, for which there are infinitely many pairs (x_i,y_i) with $f(x_i)= f(y_i)$ can reduce A_o.

Note that step 3. in the process constructs the past history of M relevant to M's action on w.

Since for every EXPTIME (and EXPTAPE) complete set, C, we know there must be some p-time reduction $f:A_0 \to C$ and since A_0 contains about 2^n-n elements of length less than or equal to n, we have that there is some n_c for which $|\{x | x \in C \text{ and } |x| < r^{n_c}\}| > 2^r$. This immediately implies, that no p-sparse set can be EXPTIME or EXPTAPE complete. This completes the proof.

We now turn our attention to hardest context-sensitive languages.

<u>Lemma 13</u>: There exists a recursive function σ such that for all linear time transducers M_i

$M_{\sigma(i)}$ is a 3 tape transducer satisfying:

1) $\forall x \; M_i(x) = M_{\sigma(i)}(x)$
2) $\exists c$ depending only on σ such that $M_{\sigma(i)}$ never scans more than $c|M_i|\frac{|x|}{\log|x|}$ squares on its work tape when processing x.

<u>Proof</u>: Follows from efficient simulation techniques of Hopcroft, Paul, and Valiant in [HPV].

We can now make use of the above transducer to enable us to diagonalize over linear time transductions on linear tape and get the following result.

<u>Theorem 14</u>: The hardest context-sensitive languages cannot be p-sparse. Thus no sla language can be a hardest csl.

<u>Proof</u>: We describe a Tm M which accepts a csl which is not p-sparse and such that any linear-time reduction of this language to another language must be one-one almost everywhere.

M behaves as follows on input w, $w \in \Sigma^*$, $|\Sigma| > 1$:

1. writes down as many of the $\frac{n}{\log n}$ space bounded transducers which simulate the linear-time transducers as possible on linear tape,

2. determines which of the transducers M_i have been eliminated while processing $x < w$, (i.e. M recomputes what it did for all previous inputs and keeps a list of the transducers which were eliminated).

3. for each M_i listed but not eliminated in increasing order M find smallest $x < w$ such that

1) $M_i(x) = M_i(w)$
2) $M_i(x) = M_i(w)$ can each be computed (although not necessarily written down) in $|w|$ tape.

If such an x is found, eliminate M_i and w is accepted iff x was rejected. If no such x is found for any i accept w.

The set accepted by this TM clearly has the property that if $f:T(M) \to A$ is a linear time reduction of T(M) to A then f is one-one a.e. Since for every n there are at least 2^n inputs of length n and at most n transducers have been checked, we see that T(M) is not a p-sparse set.

This shows that no hardest csl's can be sla languages nor can they by p-sparse, as was to be shown.

These results can easily be extended to the following.

Corollary 15: Let $L(n) \geq n$ be tape constructable. Then the hardest language for TAPE[L(n)] cannot be p-sparse. Let L(n) be tape constructable and such that for every k

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

then the complete languages of L(n) cannot be p-sparse.

V. Conclusion

A number of interesting and apparently difficult problems suggest themselves immediately from this work. As we have noted if all NP complete problems are p-isomorphic then $P \neq NP$ and if all PTAPE complete problems are p-isomorphic then $P \neq PTAPE$. Thus the question whether all NP and PTAPE complete sets, respectively, are p-isomorphic could be a very important and hard question. Similarly, the problem about the existance of sparse complete sets for NP and PTAPE seems very difficult and could help crack the P=NP=PTAPE? problem.

As a matter of fact, the sparsness question suggests a possible way of approaching the P = NP? problem. We know that if nondeterminism is used very little to accept a set in NP, say no more than logn times for inputs of length n, then the set is in P. Thus $P \neq NP$ only if nondeterminism must be and extensively during computations. We conjecture that there are no NP complete set such that nondeterminism is used only on a sparse set of inputs and furthermore, we conjecture that extensive use of nondeterminism must lead to many rejections and acceptances or the nondeterminism can be eliminated. In other words, if an essential use of nondeterminism is made in accepting a language then the nondeterministic choices must be "real" choices in the sense that many inputs are accepted and many others rejected. Thus neither L nor $\overline{L}$ could be sparse and therefore we would have shown that $P \neq NP$.

What about EXPTIME and EXPSPACE complete problems? We know that they cannot be sparse, are they all p-isomorphic? Similarly, are hardest context-sensitive languages all p-isomorphic?

We feel that these are important questions and that the techniques needed to solve them are likely to hold insights for other problems in computer science. We also believe that the study of the density properties of NP complete sets may yield a real insight in the nature of non-deterministic computations and can contribute to the solution of the P=NP? problem.

References

[AHU] Aho, A., Hopcroft, J.E., and Ullman, J. The Design and Analysis of Computer Algorithms, Addison-Wesley Publishing Company, Reading, Mass., 1974.

[C] Cook, S. "The Complexity of Theorems Proving Procedures", Proc. 3rd ACM Symposium Theory of Computing, 1974, 151-158.

[ET] Even, S. and Tarjan, R.E. "A Combinational Problem which is Complete in Polynomial Space", Proc. 7th ACM Symposium on Theory of Computing, (1975), 66-71.

[G] Galil, Z. "The Complexity of Resolution Procedures for Theorem Proving in the Propositional Calculus", Cornell University TR 75-239. 1975.

[GJS] Garey, M., Johnson, D., and Stockmeyer, L. "Some Simplified Polynomial Complete Problems", Proc. 6th ACM Symposium on Theory of Computing, (1974), 47-63.

[GR] Greibach, S.A. "The Hardest Context-Free Language", SICOMP(1973), 304-310.

[HH] Hartmanis, J. and Hunt III, H. "The LBA Problem and Its Importance in the Theory of Computing". SIAM-AMS Proceedings, Volume 7, 1974, 1-26.

[HPV] Hopcroft, J.E., Paul, W., and Valient, L. "On Time vs. Space and Related Problems". To Be Published.

[HS] Hartmanis, J. and Simon, J. "On the Structure of Feasible Computations". In Advances in Computers, Volume 14. (eds. M. Rubinoff and M.C. Yovits) Academic Press, New York, 1976.

[K] Karp, R. "Reducibilities Among Combinatorial Problems". In Complexity of Computer Computations (R. Miller and J. Thatcher, ed.) Plenum Press, 1972, 85-104.

[M] Meyer, A. Private communication.

[MS] Meyer, A. and Stockmeyer, L. "The Equivalence Problem for Regular Expressions with Squaring Requires Exponental

[SA] Sahni, S. "Some Related Problems from Network Flows, Same Theory, and Integer Programming". Conference Record IEEE 13th SWAT, (1972), 130-138.

[SI] Simon, J. "On Some Central Problems in Computational Complexity", Cornell University TR 75-224, 1975.

[U] Ullman, J. "Polynomial Complete Scheduling Problems", Proc. 4th Symposium on Operating Systems Principles, (1973), 96-101.