REDUCIBILITY AMONG COMBINATORIAL PROBLEMS[†]

Richard M. Karp

University of California at Berkeley

Abstract: A large class of computational problems involve the determination of properties of graphs, digraphs, integers, arrays of integers, finite families of finite sets, boolean formulas and elements of other countable domains. Through simple encodings from such domains into the set of words over a finite alphabet these problems can be converted into language recognition problems, and we can inquire into their computational complexity. It is reasonable to consider such a problem satisfactorily solved when an algorithm for its solution is found which terminates within a number of steps bounded by a polynomial in the length of the input. We show that a large number of classic unsolved problems of covering, matching, packing, routing, assignment and sequencing are equivalent, in the sense that either each of them possesses a polynomial-bounded algorithm or none of them does.

1. INTRODUCTION

All the general methods presently known for computing the chromatic number of a graph, deciding whether a graph has a Hamilton circuit, or solving a system of linear inequalities in which the variables are constrained to be 0 or 1, require a combinatorial search for which the worst case time requirement grows exponentially with the length of the input. In this paper we give theorems which strongly suggest, but do not imply, that these problems, as well as many others, will remain intractable perpetually.

[†]This research was partially supported by National Science Foundation Grant GJ-474.

R. E. Miller et al. (eds.), *Complexity of Computer Computations*

We are specifically interested in the existence of algorithms that are guaranteed to terminate in a number of steps bounded by a polynomial in the length of the input. We exhibit a class of well-known combinatorial problems, including those mentioned above, which are equivalent, in the sense that a polynomial-bounded algorithm for any one of them would effectively yield a polynomial-bounded algorithm for all. We also show that, if these problems do possess polynomial-bounded algorithms then all the problems in an unexpectedly wide class (roughly speaking, the class of problems solvable by polynomial-depth backtrack search) possess polynomial-bounded algorithms.

The following is a brief summary of the contents of the paper. For the sake of definiteness our technical development is carried out in terms of the recognition of languages by one-tape Turing machines, but any of a wide variety of other abstract models of computation would yield the same theory. Let Σ^* be the set of all finite strings of 0's and 1's. A subset of Σ^* is called a language. Let P be the class of languages recognizable in polynomial time by one-tape deterministic Turing machines, and let NP be the class of languages recognizable in polynomial time by one-tape nondeterministic Turing machines. Let Π be the class of functions from Σ^* into Σ^* computable in polynomial time by one-tape Turing machines. Let L and M be languages. We say that $L \propto M$ (L is reducible to M) if there is a function $f \in \Pi$ such that $f(x) \in M \Leftrightarrow x \in L$. If $M \in P$ and $L \propto M$ then $L \in P$. We call L and M equivalent if $L \propto M$ and $M \propto L$. Call L (polynomial) complete if $L \in NP$ and every language in NP is reducible to L. Either all complete languages are in P, or none of them are. The former alternative holds if and only if $P = NP$.

The main contribution of this paper is the demonstration that a large number of classic difficult computational problems, arising in fields such as mathematical programming, graph theory, combinatorics, computational logic and switching theory, are complete (and hence equivalent) when expressed in a natural way as language recognition problems.

This paper was stimulated by the work of Stephen Cook (1971), and rests on an important theorem which appears in his paper. The author also wishes to acknowledge the substantial contributions of Eugene Lawler and Robert Tarjan.

2. THE CLASS P

There is a large class of important computational problems which involve the determination of properties of graphs, digraphs, integers, finite families of finite sets, boolean formulas and

elements of other countable domains. It is a reasonable working hypothesis, championed originally by Jack Edmonds (1965) in connection with problems in graph theory and integer programming, and by now widely accepted, that such a problem can be regarded as tractable if and only if there is an algorithm for its solution whose running time is bounded by a polynomial in the size of the input. In this section we introduce and begin to investigate the class of problems solvable in polynomial time.

We begin by giving an extremely general definition of "deterministic algorithm", computing a function from a countable domain D into a countable range R.

For any finite alphabet A, let A^* be the set of finite strings of elements of A; for $x \in A^*$, let $lg(x)$ denote the length of x.

A deterministic algorithm A is specified by:

a countable set D (the domain)
a countable set R (the range)
a finite alphabet Δ such that $\Delta^* \wedge R = \phi$
an encoding function $E: D \to \Delta^*$
a transition function $\tau: \Delta^* \to \Delta^* \cup R$.

The computation of A on input $x \in D$ is the unique sequence $y_1, y_2, \ldots$ such that $y_1 = E(x)$, $y_{i+1} = \tau(y_i)$ for all i and, if the sequence is finite and ends with y_k, then $y_k \in R$. Any string occurring as an element of a computation is called an instantaneous description. If the computation of A on input x is finite and of length $t(x)$, then $t(x)$ is the running time of A on input x. A is terminating if all its computations are finite. A terminating algorithm A computes the function $f_A: D \to R$ such that $f_A(x)$ is the last element of the computation of A on x.

If $R = \{ACCEPT, REJECT\}$ then A is called a recognition algorithm. A recognition algorithm in which $D = \Sigma^*$ is called a string recognition algorithm. If A is a string recognition algorithm then the language recognized by A is $\{x \in \Sigma^* \mid f_A(x) = ACCEPT\}$. If $D = R = \Sigma^*$ then A is called a string mapping algorithm. A terminating algorithm A with domain $D = \Sigma^*$ operates in polynomial time if there is a polynomial $p(\cdot)$ such that, for every $x \in \Sigma^*$, $t(x) \leq p(lg(x))$.

To discuss algorithms in any practical context we must specialize the concept of deterministic algorithm. Various well known classes of string recognition algorithms (Markov algorithms, one-tape Turing machines, multitape and multihead Turing machines,

random access machines, etc.) are delineated by restricting the functions E and τ to be of certain very simple types. These definitions are standard [Hopcroft & Ullman (1969)] and will not be repeated here. It is by now commonplace to observe that many such classes are equivalent in their capability to recognize languages; for each such class of algorithms, the class of languages recognized is the class of recursive languages. This invariance under changes in definition is part of the evidence that recursiveness is the correct technical formulation of the concept of decidability.

The class of languages recognizable by string recognition algorithms which operate in polynomial time is also invariant under a wide range of changes in the class of algorithms. For example, any language recognizable in time $p(\cdot)$ by a multihead or multitape Turing machine is recognizable in time $p^2(\cdot)$ by a one-tape Turing machine. Thus the class of languages recognizable in polynomial time by one-tape Turing machines is the same as the class recognizable by the ostensibly more powerful multihead or multitape Turing machines. Similar remarks apply to random access machines.

Definition 1. P is the class of languages recognizable by one-tape Turing machines which operate in polynomial time.

Definition 2. Π is the class of functions from Σ^* into Σ^* defined by one-tape Turing machines which operate in polynomial time.

The reader will not go wrong by identifying P with the class of languages recognizable by digital computers (with unbounded backup storage) which operate in polynomial time and Π with the class of string mappings performed in polynomial time by such computers.

Remark. If $f: \Sigma^* \to \Sigma^*$ is in Π then there is a polynomial $p(\cdot)$ such that $lg(f(x)) \leq p(lg(x))$.

We next introduce a concept of reducibility which is of central importance in this paper.

Definition 3. Let L and M be languages. Then $L \propto M$ (L is reducible to M) if there is a function $f \in \Pi$ such that $f(x) \in M \Leftrightarrow x \in L$.

Lemma 1. If $L \propto M$ and $M \in P$ then $L \in P$.

Proof. The following is a polynomial-time bounded algorithm to decide if $x \in L$: compute $f(x)$; then test in polynomial time whether $f(x) \in M$.

We will be interested in the difficulty of recognizing subsets of countable domains other than Σ^*. Given such a domain D,

there is usually a natural one-one encoding $e: D \to \Sigma^*$. For example we can represent a positive integer by the string of 0's and 1's comprising its binary representation, a 1-dimensional integer array as a list of integers, a matrix as a list of 1-dimensional arrays, etc.; and there are standard techniques for encoding lists into strings over a finite alphabet, and strings over an arbitrary finite alphabet as strings of 0's and 1's. Given such an encoding $e: D \to \Sigma^*$, we say that a set $T \subseteq D$ is recognizable in polynomial time if $e(T) \in P$. Also, given sets $T \subseteq D$ and $U \subseteq D'$, and encoding functions $e: D \to \Sigma^*$ and $e': D' \to \Sigma^*$ we say $T \propto U$ if $e(T) \propto e'(U)$.

As a rule several natural encodings of a given domain are possible. For instance a graph can be represented by its adjacency matrix, by its incidence matrix, or by a list of unordered pairs of nodes, corresponding to the arcs. Given one of these representations, there remain a number of arbitrary decisions as to format and punctuation. Fortunately, it is almost always obvious that any two "reasonable" encodings e_0 and e_1 of a given problem are equivalent; i.e., $e_0(S) \in P \Leftrightarrow e_1(S) \in P$. One important exception concerns the representation of positive integers; we stipulate that a positive integer is encoded in a binary, rather than unary, representation. In view of the invariance of recognizability in polynomial time and reducibility under reasonable encodings, we discuss problems in terms of their original domains, without specifying an encoding into Σ^*.

We complete this section by listing a sampling of problems which are solvable in polynomial time. In the next section we examine a number of close relatives of these problems which are not known to be solvable in polynomial time. Appendix 1 establishes our notation.

Each problem is specified by giving (under the heading "INPUT") a generic element of its domain of definition and (under the heading "PROPERTY") the property which causes an input to be accepted.

SATISFIABILITY WITH AT MOST 2 LITERALS PER CLAUSE [Cook (1971)]
INPUT: Clauses $C_1, C_2, \ldots, C_p$, each containing at most 2 literals
PROPERTY: The conjunction of the given clauses is satisfiable; i.e., there is a set $S \subseteq \{x_1, x_2, \ldots, x_n, \bar{x}_1, \bar{x}_2, \ldots, \bar{x}_n\}$ such that

a) S does not contain a complementary pair of literals and
b) $S \cap C_k \neq \phi$, $k = 1,2,\ldots,p$.

MINIMUM SPANNING TREE [Kruskal (1956)]
INPUT: G, w, W
PROPERTY: There exists a spanning tree of weight $\leq W$.

SHORTEST PATH [Dijkstra (1959)]
INPUT: G, w, W, s, t
PROPERTY: There is a path between s and t of weight $\leq$ W.

MINIMUM CUT [Edmonds & Karp (1972)]
INPUT: G, w, W, s, t
PROPERTY: There is an s,t cut of weight $\leq$ W.

ARC COVER [Edmonds (1965)]
INPUT: G, k
PROPERTY: There is a set $Y \subseteq A$ such that $|Y| \leq k$ and every node is incident with an arc in Y.

ARC DELETION
INPUT: G, k
PROPERTY: There is a set of k arcs whose deletion breaks all cycles.

BIPARTITE MATCHING [Hall (1948)]
INPUT: $S \subseteq Z_p \times Z_p$
PROPERTY: There are p elements of S, no two of which are equal in either component.

SEQUENCING WITH DEADLINES
INPUT: $(T_1,\ldots,T_n) \in Z^n$, $(D_1,\ldots,D_n) \in Z^n$, k
PROPERTY: Starting at time 0, one can execute jobs 1,2,...,n, with execution times T_i and deadlines D_i, in some order such that not more than k jobs miss their deadlines.

SOLVABILITY OF LINEAR EQUATIONS
INPUT: (c_{ij}), (a_i)
PROPERTY: There exists a vector (y_j) such that, for each i, $\sum_j c_{ij} y_j = a_i$.

3. NONDETERMINISTIC ALGORITHMS AND COOK'S THEOREM

In this section we state an important theorem due to Cook (1971) which asserts that any language in a certain wide class *NP* is reducible to a specific set S, which corresponds to the problem of deciding whether a boolean formula in conjunctive normal form is satisfiable.

Let $P^{(2)}$ denote the class of subsets of $\Sigma^* \times \Sigma^*$ which are recognizable in polynomial time. Given $L^{(2)} \in P^{(2)}$ and a polynomial p, we define a language L as follows:

$L = \{x|$ there exists y such that $\langle x,y\rangle \in L^{(2)}$ and $\lg(y) \leq p(\lg(x))\}$.

We refer to L as the language derived from $L^{(2)}$ by p-bounded existential quantification.

Definition 4. NP is the set of languages derived from elements of $P^{(2)}$ by polynomial-bounded existential quantification.

There is an alternative characterization of NP in terms of nondeterministic Turing machines. A nondeterministic recognition algorithm A is specified by:

a countable set D (the domain)
a finite alphabet Δ such that $\Delta^* \cap \{\text{ACCEPT},\text{REJECT}\} = \phi$
an encoding function $E: D \to \Delta^*$
a transition relation $\tau \subseteq \Delta^* \times (\Delta^* \cup \{\text{ACCEPT},\text{REJECT}\})$

such that, for every $y_0 \in \Delta^*$, the set $\{\langle y_0,y\rangle \mid \langle y_0,y\rangle \in \tau\}$ has fewer than k_A elements, where k_A is a constant. A computation of A on input $x \in D$ is a sequence $y_1,y_2,\ldots$ such that $y_1 = E(x)$, $\langle y_i,y_{i+1}\rangle \in \tau$ for all i, and, if the sequence is finite and ends with y_k, then $y_k \in \{\text{ACCEPT},\text{REJECT}\}$. A string $y \in \Delta^*$ which occurs in some computation is an instantaneous description. A finite computation ending in ACCEPT is an accepting computation. Input x is accepted if there is an accepting computation for x. If $D = \Sigma^*$ then A is a nondeterministic string recognition algorithm and we say that A operates in polynomial time if there is a polynomial $p(\cdot)$ such that, whenever A accepts x, there is an accepting computation for x of length $\leq p(\lg(x))$.

A nondeterministic algorithm can be regarded as a process which, when confronted with a choice between (say) two alternatives, can create two copies of itself, and follow up the consequences of both courses of action. Repeated splitting may lead to an exponentially growing number of copies; the input is accepted if any sequence of choices leads to acceptance.

The nondeterministic 1-tape Turing machines, multitape Turing machines, random-access machines, etc. define classes of nondeterministic string recognition algorithms by restricting the encoding function E and transition relation τ to particularly simple forms. All these classes of algorithms, restricted to operate in polynomial time, define the same class of languages. Moreover, this class is NP.

Theorem 1. $L \in NP$ if and only if L is accepted by a nondeterministic Turing machine which operates in polynomial time.

Proof. $\Rightarrow$ Suppose $L \in NP$. Then, for some $L^{(2)} \in P^{(2)}$ and some polynomial p, L is obtained from $L^{(2)}$ by p-bounded existential quantification. We can construct a nondeterministic

machine which first guesses the successive digits of a string y of length $\leq p(\lg(y))$ and then tests whether $\langle x,y\rangle \in L^{(2)}$. Such a machine clearly recognizes L in polynomial time.

$\Leftarrow$ Suppose L is accepted by a nondeterministic Turing machine T which operates in time p. Assume without loss of generality that, for any instantaneous description Z, there are at most two instantaneous descriptions that may follow Z (i.e., at most two primitive transitions are applicable). Then the sequence of choices of instantaneous descriptions made by T in a given computation can be encoded as a string y of 0's and 1's, such that $\lg(y) \leq p(\lg(x))$.

Thus we can construct a deterministic Turing machine T', with $\Sigma^* \times \Sigma^*$ as its domain of inputs, which, on input $\langle x,y\rangle$, simulates the action of T on input x with the sequence of choices y. Clearly T' operates in polynomial time, and L is obtained by polynomial bounded existential quantification from the set of pairs of strings accepted by T'.

The class NP is very extensive. Loosely, a recognition problem is in NP if and only if it can be solved by a backtrack search of polynomial bounded depth. A wide range of important computational problems which are not known to be in P are obviously in NP. For example, consider the problem of determining whether the nodes of a graph G can be colored with k colors so that no two adjacent nodes have the same color. A nondeterministic algorithm can simply guess an assignment of colors to the nodes and then check (in polynomial time) whether all pairs of adjacent nodes have distinct colors.

In view of the wide extent of NP, the following theorem due to Cook is remarkable. We define the satisfiability problem as follows:

SATISFIABILITY
INPUT: Clauses $C_1,C_2,\ldots,C_p$
PROPERTY: The conjunction of the given clauses is satisfiable; i.e., there is a set $S \subseteq \{x_1,x_2,\ldots,x_n;\bar{x}_1,\bar{x}_2,\ldots,\bar{x}_n\}$ such that
a) S does not contain a complementary pair of literals and b) $S \cap C_k \neq \phi$, $k = 1,2,\ldots,p$.

Theorem 2 (Cook). If $L \in NP$ then $L \propto$ SATISFIABILITY.

The theorem stated by Cook (1971) uses a weaker notion of reducibility than the one used here, but Cook's proof supports the present statement.

Corollary 1. $P = NP \Leftrightarrow$ SATISFIABILITY $\in P$.

Proof. If SATISFIABILITY $\in P$ then, for each $L \in NP$, $L \in P$, since $L \propto$ SATISFIABILITY. If SATISFIABILITY $\notin P$, then, since clearly SATISFIABILITY $\in NP$, $P \neq NP$.

Remark. If $P = NP$ then NP is closed under complementation and polynomial-bounded existential quantification. Hence it is also closed under polynomial-bounded universal quantification. It follows that a polynomial-bounded analogue of Kleene's Arithmetic Hierarchy [Rogers (1967)] becomes trivial if $P = NP$.

Theorem 2 shows that, if there were a polynomial-time algorithm to decide membership in SATISFIABILITY then every problem solvable by a polynomial-depth backtrack search would also be solvable by a polynomial-time algorithm. This is strong circumstantial evidence that SATISFIABILITY $\notin P$.

4. COMPLETE PROBLEMS

The main object of this paper is to establish that a large number of important computational problems can play the role of SATISFIABILITY in Cook's theorem. Such problems will be called complete.

Definition 5. The language L is (polynomial) complete if

a) $L \in NP$

and b) SATISFIABILITY $\propto L$.

Theorem 3. Either all complete languages are in P, or none of them are. The former alternative holds if and only if $P = NP$.

We can extend the concept of completeness to problems defined over countable domains other than Σ^*.

Definition 6. Let D be a countable domain, e a "standard" one-one encoding $e: D \to \Sigma^*$ and T a subset of D. Then T is complete if and only if $e(D)$ is complete.

Lemma 2. Let D and D' be countable domains, with one-one encoding functions e and e'. Let $T \subseteq D$ and $T' \subseteq D'$. Then $T \propto T'$ if there is a function $F: D \to D'$ such that

a) $F(x) \in T' \Leftrightarrow x \in T$

and b) there is a function $f \in \Pi$ such that $f(x) = e'(F(e^{-1}(x)))$ whenever $e'(F(e^{-1}(x)))$ is defined.

The rest of the paper is mainly devoted to the proof of the following theorem.

Main Theorem. All the problems on the following list are complete.

1. SATISFIABILITY
   COMMENT: By duality, this problem is equivalent to determining whether a disjunctive normal form expression is a tautology.

2. 0-1 INTEGER PROGRAMMING
   INPUT: integer matrix C and integer vector d
   PROPERTY: There exists a 0-1 vector x such that $Cx = d$.

3. CLIQUE
   INPUT: graph G, positive integer k
   PROPERTY: G has a set of k mutually adjacent nodes.

4. SET PACKING
   INPUT: Family of sets $\{S_j\}$, positive integer ℓ
   PROPERTY: $\{S_j\}$ contains ℓ mutually disjoint sets.

5. NODE COVER
   INPUT: graph G', positive integer ℓ
   PROPERTY: There is a set $R \subseteq N'$ such that $|R| \leq \ell$ and every arc is incident with some node in R.

6. SET COVERING
   INPUT: finite family of finite sets $\{S_j\}$, positive integer k
   PROPERTY: There is a subfamily $\{T_h\} \subseteq \{S_j\}$ containing $\leq k$ sets such that $\cup T_h = \cup S_j$.

7. FEEDBACK NODE SET
   INPUT: digraph H, positive integer k
   PROPERTY: There is a set $R \subseteq V$ such that every (directed) cycle of H contains a node in R.

8. FEEDBACK ARC SET
   INPUT: digraph H, positive integer k
   PROPERTY: There is a set $S \subseteq E$ such that every (directed) cycle of H contains an arc in S.

9. DIRECTED HAMILTON CIRCUIT
   INPUT: digraph H
   PROPERTY: H has a directed cycle which includes each node exactly once.

10. UNDIRECTED HAMILTON CIRCUIT
    INPUT: graph G
    PROPERTY: G has a cycle which includes each node exactly once.

11. SATISFIABILITY WITH AT MOST 3 LITERALS PER CLAUSE
INPUT: Clauses $D_1, D_2, \ldots, D_r$, each consisting of at most 3 literals from the set $\{u_1, u_2, \ldots, u_m\} \cup \{\bar{u}_1, \bar{u}_2, \ldots, \bar{u}_m\}$
PROPERTY: The set $\{D_1, D_2, \ldots, D_r\}$ is satisfiable.

12. CHROMATIC NUMBER
INPUT: graph G, positive integer k
PROPERTY: There is a function $\phi: N \to Z_k$ such that, if u and v are adjacent, then $\phi(u) \neq \phi(v)$.

13. CLIQUE COVER
INPUT: graph G', positive integer ℓ
PROPERTY: N' is the union of ℓ or fewer cliques.

14. EXACT COVER
INPUT: family $\{S_j\}$ of subsets of a set $\{u_i, i = 1,2,\ldots,t\}$
PROPERTY: There is a subfamily $\{T_h\} \subseteq \{S_j\}$ such that the sets T_h are disjoint and $\cup T_h = \cup S_j = \{u_i, i = 1,2,\ldots,t\}$.

15. HITTING SET
INPUT: family $\{U_i\}$ of subsets of $\{s_j, j = 1,2,\ldots,r\}$
PROPERTY: There is a set W such that, for each i, $|W \cap U_i| = 1$.

16. STEINER TREE
INPUT: graph G, $R \subseteq N$, weighting function $w: A \to Z$, positive integer k
PROPERTY: G has a subtree of weight $\leq k$ containing the set of nodes in R.

17. 3-DIMENSIONAL MATCHING
INPUT: set $U \subseteq T \times T \times T$, where T is a finite set
PROPERTY: There is a set $W \subseteq U$ such that $|W| = |T|$ and no two elements of W agree in any coordinate.

18. KNAPSACK
INPUT: $(a_1, a_2, \ldots, a_r, b) \in Z^{n+1}$
PROPERTY: $\sum a_j x_j = b$ has a 0-1 solution.

19. JOB SEQUENCING
INPUT: "execution time vector" $(T_1, \ldots, T_p) \in Z^p$,
"deadline vector" $(D_1, \ldots, D_p) \in Z^p$
"penalty vector" $(P_1, \ldots, P_p) \in Z^p$
positive integer k
PROPERTY: There is a permutation π of $\{1,2,\ldots,p\}$ such that

$$\left(\sum_{j=1}^{p} [\text{if } T_{\pi(1)} + \cdots + T_{\pi(j)} > D_{\pi(j)} \text{ then } P_{\pi(j)} \text{ else } 0] \right) \leq k \ .$$

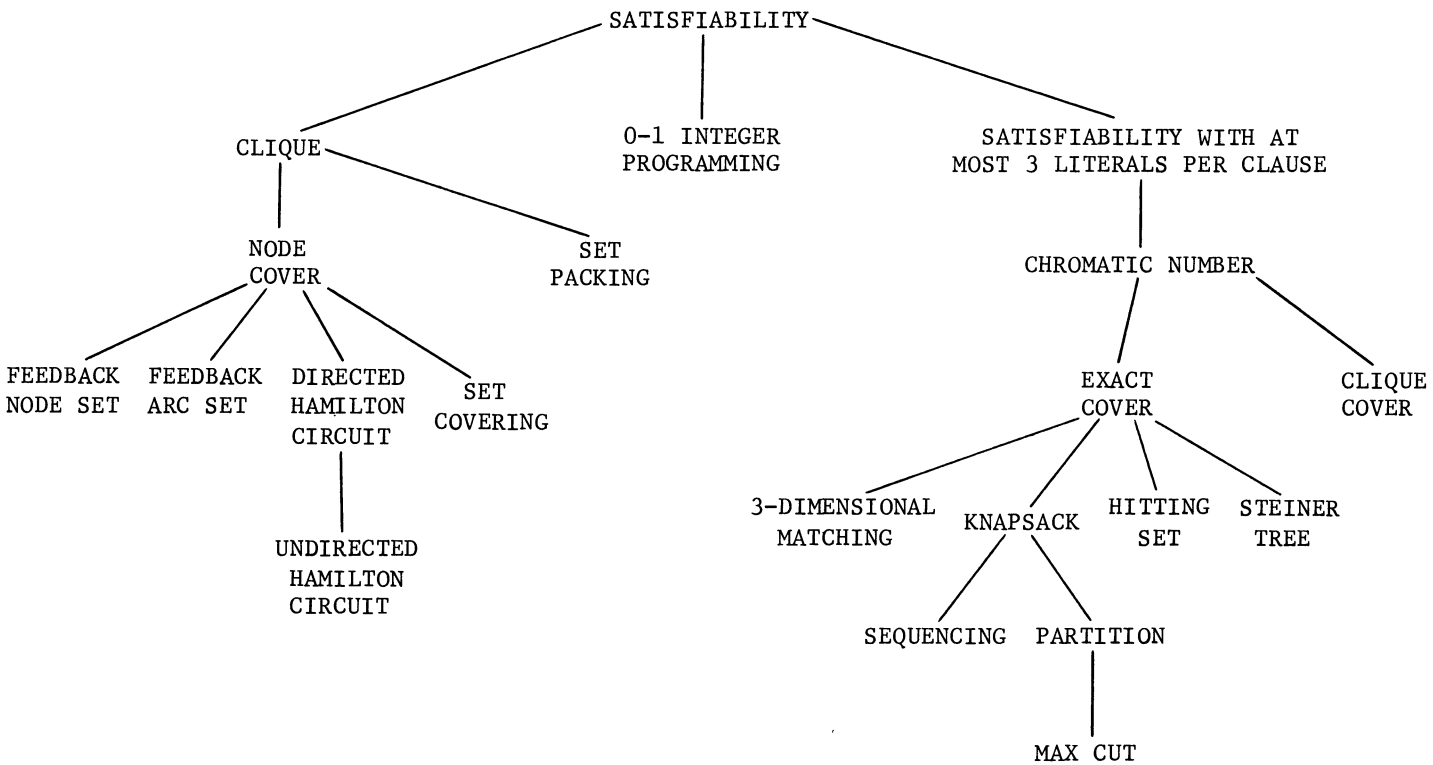

FIGURE 1 - Complete Problems

20. PARTITION
INPUT: $(c_1,c_2,\ldots,c_s) \in Z^s$
PROPERTY: There is a set $I \subseteq \{1,2,\ldots,s\}$ such that $\sum_{h\in I} c_h = \sum_{h\notin I} c_h$.

21. MAX CUT
INPUT: graph G, weighting function $w: A \to Z$, positive integer W
PROPERTY: There is a set $S \subseteq N$ such that

$$\sum_{\substack{\{u,v\}\in A \\ u\in S \\ v\notin S}} w(\{u,v\}) \geq W \quad .$$

It is clear that these problems (or, more precisely, their encodings into Σ^*), are all in NP. We proceed to give a series of explicit reductions, showing that SATISFIABILITY is reducible to each of the problems listed. Figure 1 shows the structure of the set of reductions. Each line in the figure indicates a reduction of the upper problem to the lower one.

To exhibit a reduction of a set $T \subseteq D$ to a set $T' \subseteq D'$, we specify a function $F: D \to D'$ which satisfies the conditions of Lemma 2. In each case, the reader should have little difficulty in verifying that F does satisfy these conditions.

SATISFIABILITY $\propto$ 0-1 INTEGER PROGRAMMING

$$c_{ij} = \begin{cases} 1 & \text{if } x_j \in C_i \\ -1 & \text{if } \bar{x}_j \in C_i \\ 0 & \text{otherwise} \end{cases} \qquad \begin{matrix} i = 1,2,\ldots,p \\ j = 1,2,\ldots,n \end{matrix}$$

$$b_i = 1 - (\text{the number of complemented variables in } C_i) ,$$
$$i = 1,2,\ldots,p.$$

SATISFIABILITY $\propto$ CLIQUE

$$N = \{\langle\sigma,i\rangle \mid \sigma \text{ is a literal and occurs in } C_i\}$$

$$A = \{\{\langle\sigma,i\rangle,\langle\delta,j\rangle\} \mid i \neq j \text{ and } \sigma \neq \bar{\delta}\}$$

$k = p$, the number of clauses.

CLIQUE $\propto$ SET PACKING

Assume $N = \{1,2,\ldots,n\}$. The elements of the sets $S_1,S_2,\ldots,S_n$ are those two-element sets of nodes $\{i,j\}$ not in A.

$$S_i = \{\{i,j\} \mid \{i,j\} \notin A\}, \quad i = 1,2,\ldots,n$$

$\ell = k$.

CLIQUE $\propto$ NODE COVER

G' is the complement of G.
$\ell = |N| - k$

NODE COVER $\propto$ SET COVERING

Assume $N' = \{1,2,\ldots,n\}$. The elements are the arcs of G'. S_j is the set of arcs incident with node j. $k = \ell$.

NODE COVER $\propto$ FEEDBACK NODE SET

$V = N'$
$E = \{\langle u,v\rangle \mid \{u,v\} \in A'\}$
$k = \ell$

NODE COVER $\propto$ FEEDBACK ARC SET

$V = N' \times \{0,1\}$
$E = \{\langle\langle u,0\rangle,\langle u,1\rangle\rangle \mid u \in N'\} \cup \{\langle\langle u,1\rangle,\langle v,0\rangle\rangle \mid \{u,v\} \in A'\}$
$k = \ell$.

NODE COVER $\propto$ DIRECTED HAMILTON CIRCUIT

Without loss of generality assume $A' = Z_m$.

$V = \{a_1,a_2,\ldots,a_\ell\} \cup \{\langle u,i,\alpha\rangle \mid u \in N'$ is incident with $i \in A'$ and $\alpha \in \{0,1\}\}$

$E = \{\langle\langle u,i,0\rangle,\langle u,i,1\rangle\rangle \mid \langle u,i,0\rangle \in V\}$
$\cup \{\langle\langle u,i,\alpha\rangle,\langle v,i,\alpha\rangle\rangle \mid i \in A'$, u and v are incident with i, $\alpha \in \{0,1\}\}$
$\cup \{\langle\langle u,i,1\rangle,\langle u,j,0\rangle\rangle \mid u$ is incident with i and j and $\nexists h$, $i < h < j$, such that u is incident with $h\}$
$\cup \{\langle\langle u,i,1\rangle,a_f\rangle \mid 1 \leq f \leq \ell$ and $\exists h > i$ such that u is incident with $h\}$
$\cup \{\langle a_f,\langle u,i,0\rangle\rangle \mid 1 \leq f \leq \ell$ and $\exists h < i$ such that u is incident with $h\}$.

DIRECTED HAMILTON CIRCUIT $\propto$ UNDIRECTED HAMILTON CIRCUIT

$N = V \times \{0,1,2\}$
$A = \{\{\langle u,0\rangle,\langle u,1\rangle\},\{\langle u,1\rangle,\langle u,2\rangle\} \mid u \in V\}$
$\cup \{\{\langle u,2\rangle,\langle v,0\rangle\} \mid \langle u,v\rangle \in E\}$

SATISFIABILITY $\propto$ SATISFIABILITY WITH AT MOST 3 LITERALS PER CLAUSE

Replace a clause $\sigma_1 \cup \sigma_2 \cup \cdots \cup \sigma_m$, where the σ_i are literals and $m > 3$, by

$$(\sigma_1 \cup \sigma_2 \cup u_1)(\sigma_3 \cup \cdots \cup \sigma_m \cup \bar{u}_1)(\bar{\sigma}_3 \cup u_1)\cdots(\bar{\sigma}_m \cup u_1) \quad ,$$

where u_1 is a new variable. Repeat this transformation until no clause has more than three literals.

SATISFIABILITY WITH AT MOST 3 LITERALS PER CLAUSE
$\propto$ CHROMATIC NUMBER

Assume without loss of generality that $m \geq 4$.

$N = \{u_1,u_2,\ldots,u_m\} \cup \{\bar{u}_1,\bar{u}_2,\ldots,\bar{u}_m\} \cup \{v_1,v_2,\ldots,v_m\}$
$\cup \{D_1,D_2,\ldots,D_r\}$

$A = \{\{u_i,\bar{u}_i\} \mid i=1,2,\ldots,n\} \cup \{\{v_i,v_j\} \mid i\neq j\} \cup \{\{v_i,\bar{v}_j\} \mid i\neq j\}$
$\cup \{\{v_i,\bar{v}_j\} \mid i\neq j\} \cup \{\{u_i,D_f\} \mid u_i \notin D_f\} \cup \{\{\bar{u}_i,D_f\} \mid \bar{u}_i \in D_f\}$

$k = r+1$

CHROMATIC NUMBER $\propto$ CLIQUE COVER

G' is the complement of G
$\ell = k$.

CHROMATIC NUMBER $\propto$ EXACT COVER

The set of elements is

$$N \cup A \cup \{\langle u,e,f\rangle \mid u \text{ is incident with } e \text{ and } 1 \leq f \leq k\} \quad .$$

The sets S_j are the following:

for each f, $1 \leq f \leq k$, and each $u \in N$,
$\{u\} \cup \{\langle u,e,f\rangle \mid e \text{ is incident with } u\}$;

for each $e \in A$ and each pair f_1, f_2 such that
$1 \leq f_1 \leq k$, $1 \leq f_2 \leq k$ and $f_1 \neq f_2$
$\{e\} \cup \{\langle u,e,f\rangle, f\neq f_1\} \cup \{\langle v,e,g\rangle \mid g\neq f_2\}$,

where u and v are the two nodes incident with e.

EXACT COVER $\propto$ HITTING SET

The hitting set problem has sets U_i and elements s_j, such that $s_j \in U_i \Leftrightarrow u_i \in S_j$.

EXACT COVER $\propto$ STEINER TREE

$N = \{n_o\} \cup \{S_j\} \cup \{u_i\}$
$R = \{n_o\} \cup \{u_i\}$
$A = \{\{n_o,S_j\}\} \cup \{\{S_j,u_i\} \mid u_i \in S_j\}$
$w(\{n_o,S_j\}) = |S_j|$
$w(\{S_j,u_i\}) = 0$
$k = |\{u_i\}|$.

EXACT COVER $\propto$ 3-DIMENSIONAL MATCHING

Without loss of generality assume $|S_j| \geq 2$ for each j.
Let $T = \{\langle i,j\rangle \mid u_i \in S_j\}$. Let α be an arbitrary one-one function

from $\{u_i\}$ into T. Let $\pi: T \to T$ be a permutation such that, for each fixed j, $\{\langle i,j\rangle \mid u_i \in S_j\}$ is a cycle of π.

$$U = \{\langle\alpha(u_i),\langle i,j\rangle,\langle i,j\rangle\rangle \mid \langle i,j\rangle \in T\}$$
$$\cup \{\langle\beta,\sigma,\pi(\sigma)\rangle \mid \text{for all } i,\ \beta \neq \alpha(u_i)\} \quad .$$

EXACT COVER $\propto$ KNAPSACK

Let $d = |\{S_j\}| + 1$. Let $\epsilon_{ji} = \begin{cases} 1 & \text{if } u_i \in S_j \\ 0 & \text{if } u_i \notin S_j \end{cases}$. Let

$r = |\{S_j\}|$, $a_j = \sum \epsilon_{ji} d^{i-1}$ and $b = \frac{d^t - 1}{d - 1}$.

KNAPSACK $\propto$ SEQUENCING

$$p = r, \quad T_i = P_i = a_i, \quad D_i = b.$$

KNAPSACK $\propto$ PARTITION

$$s = r + 2$$
$$c_i = a_i, \quad i = 1,2,\ldots,r$$
$$c_{r+1} = b + 1$$
$$c_{r+2} = \left(\sum_{i=1}^{r} a_i\right) + 1 - b$$

PARTITION $\propto$ MAX CUT

$$N = \{1,2,\ldots,s\}$$
$$A = \{\{i,j\} \mid i \in N,\ j \in N,\ i \neq j\}$$
$$w(\{i,j\}) = c_i \cdot c_j$$
$$W = \left\lceil \frac{1}{4} \Sigma\, c_i^2 \right\rceil$$

Some of the reductions exhibited here did not originate with the present writer. Cook (1971) showed that SATISFIABILITY $\propto$ SATISFIABILITY WITH AT MOST 3 LITERALS PER CLAUSE. The reduction

SATISFIABILITY $\propto$ CLIQUE

is implicit in Cook (1970), and was also known to Raymond Reiter. The reduction

NODE COVER $\propto$ FEEDBACK NODE SET

was found by the Algorithms Seminar at the Cornell University Computer Science Department. The reduction

NODE COVER $\propto$ FEEDBACK ARC SET

was found by Lawler and the writer, and Lawler discovered the reduction

EXACT COVER $\propto$ 3-DIMENSIONAL MATCHING

The writer discovered that the exact cover problem was reducible to the directed traveling-salesman problem on a digraph in which the arcs have weight zero or one. Using refinements of the technique used in this construction, Tarjan showed that

EXACT COVER $\propto$ DIRECTED HAMILTON CIRCUIT

and, independently, Lawler showed that

NODE COVER $\propto$ DIRECTED HAMILTON CIRCUIT .

The reduction

DIRECTED HAMILTON CIRCUIT $\propto$ UNDIRECTED HAMILTON CIRCUIT

was pointed out by Tarjan.

Below we list three problems in automata theory and language theory to which every complete problem is reducible. These problems are not known to be complete, since their membership in *NP* is presently in doubt. The reader unacquainted with automata and language theory can find the necessary definitions in Hopcroft and Ullman (1969).

EQUIVALENCE OF REGULAR EXPRESSIONS
INPUT: A pair of regular expressions over the alphabet $\{0,1\}$
PROPERTY: The two expressions define the same language.

EQUIVALENCE OF NONDETERMINISTIC FINITE AUTOMATA
INPUT: A pair of nondeterministic finite automata with input alphabet $\{0,1\}$
PROPERTY: The two automata define the same language.

CONTEXT-SENSITIVE RECOGNITION
INPUT: A context-sensitive grammar Γ and a string x
PROPERTY: x is in the language generated by Γ.

First we show that

SATISFIABILITY WITH AT MOST 3 LITERALS PER CLAUSE
$\propto$ EQUIVALENCE OF REGULAR EXPRESSIONS .

The reduction is made in two stages. In the first stage we construct a pair of regular expressions over an alphabet $\Delta = \{u_1,u_2,\ldots,u_n,\bar{u}_1,\bar{u}_2,\ldots,\bar{u}_n\}$. We then convert these regular expressions to regular expressions over $\{0,1\}$.

The first regular expression is $\Delta^n\Delta^*$ (more exactly, Δ is written out as $(u_1+u_2+\cdots+u_n+\bar{u}_1+\cdots+\bar{u}_n)$, and Δ^n represents n copies of the expression for Δ concatenated together). The second regular expression is

$$\Delta^n\Delta^* \cup \bigcup_{i=1}^{n} (\Delta^* u_i \Delta^* \bar{u}_i \Delta^* \cup \Delta^* \bar{u}_i \Delta^* u_i \Delta^*) \cup \bigcup_{h=1}^{r} \theta(D_h)$$

where

$$\theta(D_h) = \begin{cases} \Delta^*\bar{\sigma}_1\Delta^* & \text{if } D_h = \sigma_1 \\ \Delta^*\bar{\sigma}_1\Delta^*\bar{\sigma}_2\Delta^* \cup \Delta^*\bar{\sigma}_2\Delta^*\bar{\sigma}_1\Delta^* & \text{if } D_h = \sigma_1 \cup \sigma_2 \\ \Delta^*\bar{\sigma}_1\Delta^*\bar{\sigma}_2\Delta^*\bar{\sigma}_3\Delta^* \cup \Delta^*\bar{\sigma}_1\Delta^*\bar{\sigma}_3\Delta^*\bar{\sigma}_2\Delta^* & \\ \quad \cup \Delta^*\bar{\sigma}_2\Delta^*\bar{\sigma}_1\Delta^*\bar{\sigma}_3\Delta^* \cup \Delta^*\bar{\sigma}_2\Delta^*\bar{\sigma}_3\Delta^*\bar{\sigma}_1\Delta^* & \\ \quad \cup \Delta^*\bar{\sigma}_3\Delta^*\bar{\sigma}_1\Delta^*\bar{\sigma}_2\Delta^* \cup \Delta^*\bar{\sigma}_3\Delta^*\bar{\sigma}_2\Delta^*\bar{\sigma}_1\Delta^* & \\ & \text{if } D_h = \sigma_1 \cup \sigma_2 \cup \sigma_3 \ . \end{cases}$$

Now let m be the least positive integer $\geq \log_2 |\Delta|$, and let ϕ be a 1-1 function from Δ into $\{0,1\}^m$. Replace each regular expression by a regular expression over $\{0,1\}$, by making the substitution $a \to \phi(a)$ for each occurrence of each element of Δ.

EQUIVALENCE OF REGULAR EXPRESSIONS $\propto$ EQUIVALENCE OF NONDETERMINISTIC FINITE AUTOMATA

There are standard polynomial-time algorithms [Salomaa (1969)] to convert a regular expression to an equivalent nondeterministic automaton. Finally, we show that, for any $L \in NP$,

$$L \underline{\propto} \text{ CONTEXT-SENSITIVE RECOGNITION} \ .$$

Suppose L is recognized in time $p(\ )$ by a nondeterministic Turing machine. Then the following language $\tilde{L}$ over the alphabet $\{0,1,\#\}$ is accepted by a nondeterministic linear bounded automaton which simulates the Turing machine:

$$\tilde{L} = \{\#^{p(\lg(x))}x\#^{p(\lg(x))} \mid x \in L\} \ .$$

Hence $\tilde{L}$ is context-sensitive and has a context-sensitive grammar $\tilde{\Gamma}$. Thus $x \in L$ iff

$$\tilde{\Gamma}, \#^{p(\lg(x))}x\#^{p(\lg(x))}$$

is an acceptable input to CONTEXT-SENSITIVE RECOGNITION.

We conclude by listing the following important problems in NP which are not known to be complete.

GRAPH ISOMORPHISM
INPUT: graphs G and G'
PROPERTY: G is isomorphic to G'.

NONPRIMES
INPUT: positive integer k
PROPERTY: k is composite.

LINEAR INEQUALITIES
INPUT: integer matrix C, integer vector d
PROPERTY: $Cx \geq d$ has a rational solution.

APPENDIX I

Notation and Terminology Used in Problem Specification

PROPOSITIONAL CALCULUS

$x_1,x_2,\ldots,x_n$	$u_1,u_2,\ldots,u_m$	propositional variables
$\bar{x}_1,\bar{x}_2,\ldots,\bar{x}_n$	$\bar{u}_1,\bar{u}_2,\ldots,\bar{u}_m$	complements of propositional variables
σ,σ_i		literals
$C_1,C_2,\ldots,C_p$	$D_1,D_2,\ldots,D_r$	clauses

$C_k \subseteq \{x_1,x_2,\ldots,x_n,\bar{x}_1,\bar{x}_2,\ldots,\bar{x}_n\}$

$D_\ell \subseteq \{u_1,u_2,\ldots,u_m,\bar{u}_1,\bar{u}_2,\ldots,\bar{u}_m\}$

A clause contains no complementary pair of literals.

SCALARS, VECTORS, MATRICES

Z	the positive integers
Z^p	the set of p-tuples of positive integers
Z_p	the set $\{0,1,\ldots,p-1\}$
k,W	elements of Z
$\langle x,y\rangle$	the ordered pair $\langle x,y\rangle$
(a_i) (y_j) d	vectors with nonnegative integer components
(c_{ij}) C	matrices with integer components

GRAPHS AND DIGRAPHS

$G = (N,A)$ $G' = (N',A')$ finite graphs

N,N' sets of nodes A,A' sets of arcs

s,t,u,v nodes $e,\{u,v\}$ arcs

$(X,\bar{X}) = \{\{u,v\} \mid u \in X \text{ and } v \in \bar{X}\}$ cut

If $s \in X$ and $t \in \bar{X}$, $(X,\bar{X})$ is a s-t cut.

$w: A \to Z$ $w': A' \to Z$ weight functions

The weight of a subgraph is the sum of the weights of its arcs.

$H = (V,E)$ digraph V set of nodes, E set of arcs

$e,\langle u,v\rangle$ arcs

SETS

ϕ	the empty set
$\lvert S\rvert$	the number of elements in the finite set S
$\{S_j\}$ $\{T_h\}$ $\{U_i\}$	finite families of finite sets