34 The Complexity of Theorem-Proving Procedures (1971)

Stephen A. Cook

While discussing maximal matchings in graphs a decade after Kruskal's unelaborated claim that his spanning tree algorithm was "practical" (chapter 17), the computer scientist Jack Edmonds paused to draw the distinction between polynomial- and exponential-time algorithms (as we now call them). A "digression" near the beginning of Edmonds (1965) states, "An explanation is due on the use of the words 'efficient algorithm.'... For practical purposes the difference between algebraic and exponential order is often more crucial than the difference between finite and non-finite." At about the same time, the mathematician Alan Cobham (1965) formulated a machine-model-independent class he called $\mathcal{L}$, the functions computable in time "bounded by polynomials in the lengths of the numbers involved" represented in digital form.

In this seminal paper, Stephen Cook (b. 1939) defines as $\mathcal{L}_*$ the class that Edmonds refers to as problems with "algebraic order" solutions and which are today known as $\mathcal{P}$ or PTIME. That is, $\mathcal{L}_*$ or $\mathcal{P}$ is the set analog of Cobham's class $\mathcal{L}$ of functions. Cook then provides the crucial definition of polynomial-time reducibility between problems, and proves that all problems accepted in polynomial time by a nondeterministic Turing machine are polynomial-time reducible to the set of satisfiable formulas of the propositional calculus in conjunctive normal form. (The $\mathcal{NP}$ notation for this class was adopted soon after publication of this paper—see chapter 36 and Knuth (1974b). Also, the theorem is stated in terms of tautologyhood, which is in co-$\mathcal{NP}$, rather than—as would be customary today—satisfiability, which is in $\mathcal{NP}$.)

Cook received his PhD in mathematics at Harvard in 1966, working under the direction of the logician Hao Wang on the complexity of multiplication and other mathematical functions. He joined the mathematics department at Berkeley as assistant professor. In what can only be considered a blunder, Cook was denied tenure at Berkeley in 1970 and moved to the University of Toronto. He presented this short paper at a major computer science theory conference, demonstrating that time-bounded nondeterministic computations could be described succinctly by boolean formulas, and therefore that a polynomial-time algorithm for boolean satisfiability would immediately result in polynomial-time algorithms for every $\mathcal{NP}$ problem. The proof is easy—the brilliant insight was to describe computations by formulas in a distant echo of Turing

Reprinted from Cook (1971b), with permission from the Association for Computing Machinery.

(1936, here page 60). Cook's result set off the still unfinished search for an answer to the question of whether or not $\mathcal{P} = \mathcal{NP}$ and hundreds of other research threads in complexity theory.

There is one more important historical aspect of the $\mathcal{P} = \mathcal{NP}$ question. Working independently and in relative isolation in the Soviet Union, Leonid Levin (b. 1948) had identified the same class of "Universal Search Problems" (Levin, 1973) that came to be known as the $\mathcal{NP}$-complete problems. The discoveries by Cook and Levin were nearly simultaneous, though Levin's publication was delayed. The existence of $\mathcal{NP}$-complete problems is now known as the Cook–Levin Theorem. Levin emigrated to the US in 1978 and is now a professor at Boston University.

34.0 Summary

It is shown that any recognition problem solved by a polynomial time-bounded nondeterministic Turing machine can be "reduced" to the problem of determining whether a given propositional formula is a tautology. Here "reduced" means, roughly speaking, that the first problem can be solved deterministically in polynomial time provided an oracle is available for solving the second. From this notion of reducible, polynomial degrees of difficulty are defined, and it is shown that the problem of determining tautologyhood has the same polynomial degree as the problem of determining whether the first of two given graphs is isomorphic to a subgraph of the second. Other examples are discussed. A method of measuring the complexity of proof procedures for the predicate calculus is introduced and discussed.

Throughout this paper, a *set of strings* means a set of strings on some fixed, large, finite alphabet Σ. This alphabet is large enough to include symbols for all sets described here. All Turing machines are deterministic recognition devices, unless the contrary is explicitly stated.

34.1 Tautologies and Polynomial Reducibility

Let us fix a formalism for the propositional calculus in which formulas are written as strings on Σ. Since we will require infinitely many proposition symbols (atoms), each such symbol will consist of a member of Σ followed by a number in binary notation to distinguish that symbol. Thus a formula of length n can only have about $n/\log n$ distinct function and predicate symbols. The logical connectives are $\wedge$ (and), $\vee$ (or), and $\neg$ (not).

The set of tautologies (denoted by {tautologies}) is a certain recursive set of strings on this alphabet, and we are interested in the problem of finding a good lower bound on its possible recognition times. We provide no such lower bound here, but Theorem 1 will give evidence that {tautologies} is a difficult set to recognize, since many apparently difficult problems can be reduced to determining tautologyhood. By *reduced* we mean, roughly speaking, that if tautologyhood could be decided instantly (by an "oracle") then these problems could be decided in polynomial time. In order to make this notion precise, we introduce query machines, which are like Turing machines with oracles in Kreider and Ritchie (1964).

A *query machine* is a multitape Turing machine with a distinguished tape called the *query tape*, and three distinguished states called the *query state*, *yes state*, and *no state*, respectively. If M is a query machine and T is a set of strings, then a *T-computation* of M is a computation of M in which initially M is in the initial state and has an input string w on its input tape, and each time M assumes the query state there is a string u on the query tape, and the next state M assumes is the yes state if $u \in T$ and the no state if $u \notin T$. We think of an "oracle," which knows T, placing M in the yes state or no state.

Definition. A set S of strings is *P-reducible* (P for polynomial) to a set T of strings iff there is some query machine M and a polynomial $Q(n)$ such that for each input string w, the T-computation of M with input w halts within $Q(|w|)$ steps ($|w|$ is the length of w) and ends in an accepting state iff $w \in S$.

It is not hard to see that P-reducibility is a transitive relation. Thus the relation E on sets of strings, given by $(S, T) \in E$ iff each of S and T is P-reducible to the other, is an equivalence relation. The equivalence class containing a set S will be denoted by $\deg(S)$ (the polynomial degree of difficulty of S).

Definition. We will denote $\deg(\{0\})$ by $\mathcal{L}_*$, where 0 denotes the zero function.

Thus $\mathcal{L}_*$ is the class of sets recognizable in polynomial time. $\mathcal{L}_*$ was discussed in Cook (1971a, p. 5), and is the string analog of Cobham's class $\mathcal{L}$ of functions (Cobham, 1965).

We now define the following special sets of strings.

1. *The subgraph problem* is the problem given two finite undirected graphs, determine whether the first is isomorphic to a subgraph of the second. A graph G can be represented by a string $\overline{G}$ on the alphabet $\{0, 1, *\}$ by listing the successive rows of its adjacency matrix, separated by *s. We let {subgraph pairs} denote the set of strings $\overline{G_1} ** \overline{G_2}$ such that G_1 is isomorphic to a subgraph of G_2.
2. *The graph isomorphism problem* will be represented by the set, denoted by {isomorphic graph pairs}, of all strings $\overline{G_1} ** \overline{G_2}$ such that G_1 is isomorphic to G_2.
3. The set {primes} is the set of all binary notations for prime numbers.
4. The set {DNF tautologies} is the set of strings representing tautologies in disjunctive normal form.
5. The set D_3 consists of those tautologies in disjunctive normal form in which each disjunct has at most three conjuncts (each of which is an atom or negation of an atom).

Theorem 1. If a set S of strings is accepted by some nondeterministic Turing machine within polynomial time, then S is P-reducible to {DNF tautologies}.

Corollary. Each of the sets in definitions 1–5 is P-reducible to {DNF tautologies}.

This is because each set, or its complement, is accepted in polynomial time by some nondeterministic Turing machine.

Proof of Theorem 1. Suppose a nondeterministic Turing machine M accepts a set S of strings within time $Q(n)$, where $Q(n)$ is a polynomial. Given an input w for M, we will construct a proposition formula $A(w)$ in conjunctive normal form such that $A(w)$ is satisfiable iff M accepts w.

Thus $\neg A(w)$ is easily put in disjunctive normal form (using De Morgan's laws), and $\neg A(w)$ is a tautology if and only if $w \notin S$. Since the whole construction can be carried out in time bounded by a polynomial in $|w|$ (the length of w), the theorem will be proved.

We may as well assume the Turing machine M has only one tape, which is infinite to the right but has a left-most square. Let us number the squares from left to right 1, 2, ... Let us fix an input w to M of length n, and suppose $w \in S$. Then there is a computation of M with input w that ends in an accepting state within $T = Q(n)$ steps. The formula $A(w)$ will be built from many different proposition symbols, whose intended meanings, listed below, refer to such a computation.

Suppose the tape alphabet for M is $\{\sigma_1, \ldots, \sigma_\ell\}$, and the set of states is $\{q_1, \ldots, q_r\}$. Notice that since the computation has at most $T = Q(n)$ steps, no tape square beyond T is scanned.

Proposition symbols:

$P^i_{s,t}$ for $1 \le i \le \ell$, $1 \le s, t \le T$. $P^i_{s,t}$ is true iff tape square number s at step t contains the symbol σ_i.

Q^i_t for $1 \le i \le r$, $1 \le t \le T$. Q^i_t is true iff at step t the machine is in state q_i.

$S_{s,t}$ for $1 \le s, t \le T$ is true iff at time t square number s is scanned by the tape head.

The formula $A(w)$ is a conjunction $B \wedge C \wedge D \wedge E \wedge F \wedge G \wedge H \wedge I$ formed as follows. Notice $A(w)$ is in conjunctive normal form.

B will assert that at each step t, one and only one square is scanned. B is a conjunction $B_1 \wedge B_2 \wedge \cdots \wedge B_T$, where B_t asserts that at time t one and only one square is scanned:

$$B_t = (S_{1,t} \vee S_{2,t} \vee \cdots \vee S_{T,t}) \wedge \left(\bigwedge_{1 \le i < j \le T} (\neg S_{i,t} \vee \neg S_{j,t}) \right).$$

For $1 \le s \le T$ and $1 \le t \le T$, $C_{s,t}$ asserts that at square s and time t there is one and only one symbol. C is the conjunction of all the $C_{s,t}$.

D asserts that for each t there is one and only one state.

E asserts the initial conditions are satisfied:

$$E = Q^0_1 \wedge S_{1,1} \wedge P^{i_1}_{1.1} \wedge P^{i_2}_{2,2} \wedge \cdots \wedge P^{i_n}_{n,1} \wedge P^1_{n+1,1} \wedge \cdots \wedge P^1_{T,1}$$

where $w = \sigma_{i_1} \ldots \sigma_{i_n}$, q_0 is the initial state and σ_1 is the blank symbol.

F, G, and H assert that for each time t the values of the P's, Q's and S's are updated properly. For example, G is the conjunction over all t, i, j of $G^t_{i,j}$, where $G^t_{i,j}$ asserts that if at time t the machine is in state q_i scanning symbol σ_j, then at time $t+1$ the machine is in state q_k, where q_k is the state given by the transition function for M. [EDITOR: The version of the original paper posted on Cook's website includes a handwritten note here: "(or states: nondeterminism)." Indeed, the construction as presented is correct only if the machine is deterministic; it needs to be done somewhat differently for nondeterministic machines.]

$$G^t_{i,j} = \bigwedge_{s=1}^{T} (\neg Q^i_t \vee \neg S_{s,t} \vee \neg P^j_{s,t} \vee Q^k_{t+1})$$

Finally, the formula I asserts that the machine reaches an accepting state at some time. The machine M should be modified so that it continues to compute in some trivial fashion after reaching an accepting state, so that $A(w)$ will be satisfied.

It is now straightforward to verify that $A(w)$ has all the properties asserted in the first paragraph of the proof.

Theorem 2. The following sets are P-reducible to each other in pairs (and hence each has the same polynomial degree of difficulty): {tautologies}, {DNF tautologies}, D_3, {subgraph pairs}.

Remark. We have not been able to add either {primes} or {isomorphic graph pairs} to the above list. To show {tautologies} is P-reducible to {primes} would seem to require some deep results in number theory, while showing {tautologies} is P-reducible to {isomorphic graph pairs} would probably upset a conjecture of Corneil's (Corneil and Gotlieb, 1970) from which he deduces that the graph isomorphism problem can be solved in polynomial time.

Incidentally, it is not hard to see from the Davis–Putnam procedure (Davis and Putnam, 1960) that the set D_2 consisting of all DNF tautologies with at most two conjuncts per disjunct, is in $\mathcal{L}_*$. Hence D_2 cannot be added to the list in Theorem 2 (unless all sets in the list are in $\mathcal{L}_*$).

Proof of Theorem 2. By the corollary to Theorem 1, each of the sets is P-reducible to {DNF tautologies}. Since obviously {DNF tautologies} is P-reducible to {tautologies}, it remains to show {DNF tautologies} is P-reducible to D_3 and D_3 is P-reducible to {subgraph pairs}.

To show {DNF tautologies} is P-reducible to D_3, let A be a proposition formula in disjunctive normal form. Say $A = B_1 \vee B_2 \vee \cdots \vee B_k$, where $B_1 = R_1 \wedge \cdots \wedge R_s$, and each R_i is an atom or negation of an atom, and $s > 3$. Then A is a tautology if and only if A' is a tautology where

$$A' = P \wedge R_3 \wedge \cdots \wedge R_s \vee \neg P \wedge R_1 \wedge R_2 \vee B_2 \vee \cdots \vee B_k,$$

where P is a new atom. Since we have reduced the number of conjuncts in B_1, this process may be repeated until eventually a formula is found with at most three conjuncts per disjunct. Clearly the entire process is bounded in time by a polynomial in the length of A.

It remains to show that D_3 is P-reducible to {subgraph pairs}. Suppose A is a formula in disjunctive normal form with three conjuncts per disjunct. Thus $A = C_1 \vee \cdots \vee C_k$,where $C_i = R_{i1} \wedge R_{i2} \wedge R_{i3}$, and each R_{ij} is an atom or a negation of an atom. Now let G_1 be the complete graph with vertices $\{v_1, v_2, \ldots, v_k\}$, and let G_2 be the graph with vertices $\{u_{ij}\}$, $1 \leq i \leq k$, $1 \leq j \leq 3$, such that u_{ij} is connected by an edge to u_{rs} if and only if $i \neq r$ and the two literals (R_{ij}, R_{rs}) do not form an opposite pair (that is they are neither of the form $(P, \neg P)$ nor of the form $(\neg P, P)$). Thus there is a falsifying truth assignment to the formula A iff there is a graph homomorphism $\phi : G_1 \rightarrow G_2$ such that for each i, $\phi(v_i) = u_{ij}$ for some j. (The homomorphism tells for each i which of R_{i1}, R_{i2}, R_{i3} should be falsified, and the selective lack of edges in G_2 guarantees that the resulting truth assignment is consistently specified.)

In order to guarantee that a one-one homomorphism $\phi : G_1 \rightarrow G_2$ has the property that for each i, $\phi(v_i) = u_{ij}$ for some j, we modify G_1 and G_2 as follows. We select graphs $H_1, H_2, \ldots, H_k$ which are sufficiently distinct from each other that if G_1' is formed from G_1 by attaching H_i to v_i,

$1 \leq i \leq k$, and G_2' is formed from G_2 by attaching H_i to each of u_{i1} and u_{i2} and u_{i3}, $1 \leq i \leq k$, then every one-one homomorphism $\phi : G_1' \rightarrow G_2'$ has the property just stated. It is not hard to see such a construction can be carried out in polynomial time. Then G_1' can be embedded in G_2' if and only if $A \notin D_3$. This completes the proof of Theorem 2.

34.2 Discussion

Theorem 1 and its corollary give strong evidence that it is not easy to determine whether a given proposition formula is a tautology, even if the formula is in normal disjunctive form. Theorems 1 and 2 together suggest that it is fruitless to search for a polynomial decision procedure for the subgraph problem, since success would bring polynomial decision procedures to many other apparently intractable problems. Of course the same remark applies to any combinatorial problem to which tautologies is P-reducible.

Furthermore, the theorems suggest that {tautologies} is a good candidate for an interesting set not in $\mathcal{L}_*$, and I feel it is worth spending considerable effort trying to prove this conjecture. Such a proof would be a major breakthrough in complexity theory.

In view of the apparent complexity of {DNF tautologies}, it is interesting to examine the Davis–Putnam procedure (Davis and Putnam, 1960). This procedure was designed to determine whether a given formula in conjunctive normal form is satisfiable, but of course the "dual" procedure determines whether a given formula in disjunctive normal form is a tautology. I have not yet been able to find a series of examples showing the procedure (treated sympathetically to avoid certain pitfalls) must require more than polynomial time. Nor have I found an interesting upper bound for the time required.

If we let strings represent natural numbers (or k-tuples of natural numbers) using m-adic or other suitable notation, then the notions in the preceding sections can be made to apply to sets of numbers (or k-place relations on numbers). It is not hard to see that the set of relations accepted in polynomial time by some nondeterministic Turing machine is precisely the set $\mathcal{L}^+$ of relations of the form

$$(\exists y \leq g_k(\overline{x}))\, R(\overline{x}, y) \tag{34.1}$$

where $g_k(\overline{x}) = 2^{(\ell(\max \overline{x}))^k}$, $\ell(z)$ is the dyadic length of z, and $R(\overline{x}, y)$ is an $\mathcal{L}_*$ relation. ($\mathcal{L}^+$ is the class of extended positive rudimentary relations of Bennett (1962).) If we remove the bound on the quantifier in formula (34.1), the class $\mathcal{L}^+$ would become the class of recursively enumerable sets. Thus if $\mathcal{L}^+$ is the analog of the class of r.e. sets, then determining tautologyhood is the analog of the halting problem; since, according to Theorem 1, {tautologies} has the complete $\mathcal{L}^+$ degree just as the halting problem has the complete r.e. degree. Unfortunately, the diagonal argument which shows the halting problem is not recursive apparently cannot be adapted to show {tautologies} is not in $\mathcal{L}_*$. ...

This is a section of doi:10.7551/mitpress/12274.001.0001

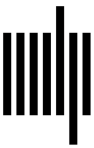
Ideas That Created the Future

Classic Papers of Computer Science

Edited by: Harry R. Lewis

Citation:

Ideas That Created the Future: Classic Papers of Computer Science
Edited by: Harry R. Lewis
DOI: 10.7551/mitpress/12274.001.0001
ISBN (electronic): 9780262363174
Publisher: The MIT Press
Published: 2021



The MIT Press

This book was set in Times New Roman by the editor using LaTeX.

Library of Congress Cataloging-in-Publication Data

Names: Lewis, Harry R., editor.
Title: Ideas that created the future : classic papers of computer science / edited by Harry R. Lewis.
Description: Cambridge, Massachusetts : The MIT Press, [2021] | Includes bibliographical references and index.
Identifiers: LCCN 2020018950 | ISBN 9780262045308 (paperback)
Subjects: LCSH: Computer science.
Classification: LCC QA76 .I34 2020 | DDC 004–dc23
LC record available at https://lccn.loc.gov/2020018950