The History and Status of the P versus NP Question

Michael Sipser[*]
Department of Mathematics
Massachusetts Institute of Technology
Cambridge MA 02139

As long as a branch of science offers an abundance of problems, so long it is alive; a lack of problems foreshadows extinction or the cessation of independent development.

— David Hilbert, from a lecture delivered before the International Congress of Mathematicians at Paris in 1900.

When?

— Juris Hartmanis

1 Significance

The P versus NP question grew out of developments in mathematical logic and electronic technology during the middle part of the twentieth century. It is now considered to be one of the most important problems in contemporary mathematics and theoretical computer science. The literature on this subject is diverse but not huge — it is possible for the student to acquaint himself with all of the important relevant work in a semester or two. In writing this article, I have attempted to organize and describe this literature, including an occasional opinion about the most fruitful directions, but no technical details.

In the first half of this century, work on the power of formal systems led to the formalization of the notion of algorithm and the realization that certain problems are algorithmically unsolvable. At around this time, forerunners of the programmable computing machine were beginning to appear. As mathematicians contemplated the practical capabilities and limitations of such devices, computational complexity theory emerged from the theory of algorithmic unsolvability.

Early on, a particular type of computational task became evident, where one is seeking an object which lies somewhere in a large space of possibilities. Computers may greatly speed up the search, but the extremely large spaces that really do occur in cases of interest would still require geologic time to search exhaustively even on the fastest machines imaginable. The only way to solve such cases practically would be to find a method which avoided searching by brute force. Roughly speaking, the P versus NP question asks whether, in general, such a method exists.

This question has attracted considerable attention. Its intuitive statement is simple and accessible to non-specialists, even those outside of science. By embracing issues in the foundations of mathematics as well as in the practice of computing, it gains something in character beyond that of a mere puzzle, but rather with apparently deeper significance and broader consequences.

Church and Turing, in their work on Hilbert's *entschiedungsproblem*, have shown that testing whether an assertion has a proof is algorithmically unsolvable. The P versus NP question can be viewed as a more finitary version of the *entschiedungsproblem*. Suppose we wish to test whether an assertion has a *short* proof. Of course this can be solved by a brute force algorithm, but can it be done efficiently?

The *entschiedungsproblem* does have practical importance in addition to it's philosophical significance. Mathematical proof is a codification of more general human reasoning. An automatic theorem prover would have wide application within computer science, if it operated efficiently enough. Even though this is hopeless

[*]Supported by NSF Grant 8912586 CCR and Air Force Contract AFOSR 89-0271.

in general, there may be important special cases which are solvable. It would be nice if Church's or Turing's proofs gave us some information about where the easier cases might lie. Unfortunately, their arguments rest on "self-reference," a contrived phenomenon which never appears spontaneously. This does not tell us what makes the problem hard in interesting cases. Conceivably, a proof that P is not equal to NP would be more informative.

2 History

Among the many equivalent formulations of the P versus NP question, the following one is particularly simple to describe.

Is there an algorithm which, when presented with an undirected graph, determines whether it contains a complete subgraph on half its nodes, and whose running time is bounded above by a polynomial in the number of nodes?

By the term *algorithm* we mean a finite set of instructions on any of a number of related models of computation, *e.g.,* the multi-tape Turing machine. The term P, for Polynomial time, refers to the class of languages languages decidable on a Turing machine in time bounded above by a polynomial. It is invariant over reasonable computational models. NP, for Nondeterministic Polynomial time, refers to the analogous class for nondeterministic Turing machines. NP consists of those languages where membership is *verifiable* in polynomial time. The question of whether P is equal to NP is equivalent to whether an *NP-complete* problem, such as the clique problem described above, can be solved in polynomial time. The books by Hopcroft and Ullman [HU79] and Garey and Johnson [GJ79] contain excellent introductions to this subject.

2.1 Brute Force Search

The idea of using brute force search to solve certain problems is certainly very old. In principle, many naturally occurring problems may be solved in this way, though if the search space is large, this becomes obviously impractical. People realized this difficulty in particular cases, and occasionally were able to find alternative methods. One of the early accomplishments of the theory of algorithms was to recognize brute force search as a phenomenon independent of any specific problem. A number of researchers, working separately, hit upon this idea.

Gödel, in a recently discovered 1956 letter to von Neumann,[1] wrote about the issue in a remarkably modern way, formulating it in terms of the time required on a Turing machine to test whether a formula in the predicate calculus has a proof of length n. He also asked how much we can improve upon brute force, in general, when solving combinatorial problems. It is not known whether von Neumann, who was dying of cancer at the time, thought about Gödel's letter or responded to it. Von Neumann does appear to have had some awareness of the undesirability of brute force search. A couple of years earlier he had written a paper giving an algorithm avoiding brute force search for the assignment problem [Vo53] .

Edmonds [Ed65] gave the first lucid account of the issue of brute force search appearing in western literature. The main contribution of his paper was a polynomial time algorithm for the maximum matching problem. This algorithm itself still stands as a beautiful example of how brute force can sometimes be avoided, if one is clever enough. Edmonds felt the need to explain why his was a result at all, when there is an obvious finite algorithm for solving the problem. In so doing he discussed the problem of brute force search in general as an algorithmic method.

In the 1950's, researchers in the Soviet Union were aware of the issue of brute force search, called *perebor* in Russian. Yablonski [Ya59a] described the issue for general problems and focused on the specific problem of constructing the smallest circuit for a given function. Curiously, in a later paper [Ya59b] he obscured the issue by claiming to prove that perebor is unavoidable for this problem. That paper, whose main theorem makes no reference to the notion of algorithm, certainly did not do as claimed, at least in the sense we would mean it today. Trakhtenbrot sheds some light on this murky situation in his survey of early Russian work on perebor [Tr84].

2.2 P and NP

Several early papers, notably those of Rabin [Ra60], Hartmanis and Stearns [HS65], and Blum [Bl67], proposed and developed the notion of measuring the complexity of a problem in terms of the number of steps required to solve it with an algorithm. Cobham [Co64], Edmonds [Ed65], and Rabin [Ra66] suggested the class P as a reasonable approximation to the class of "realistically solvable" problems.

The importance of the class NP stems from the large number of computational problems contained within. Cook [Co71] and Levin [Le73] first defined this class and proved the existence of complete problems. Somewhat

[1] A translation of Gödel's letter along with quotations from a number of historical papers are included in an appendix.

earlier, Edmonds [Ed65b] captured the essence of the notion of NP with his "good characterization" whereby an assistant can convince his supervisor of the solution to a problem. Since the solution may be either positive or negative in the case of a problem involving language membership, Edmonds's notion is generally associated with the class NP $\cap$ coNP. Karp [Ka72] demonstrated the surprisingly wide extent of NP-completeness when he showed that many familiar problems have this property.

2.3 Explicit Conjecture

Cook first precisely formulated the P versus NP conjecture in his 1971 paper. More or less close approximations to it appeared somewhat earlier in the papers of Edmonds, Levin, Yablonski, and in Gödel's letter.

3 Status

3.1 Diagonalization and Relativization

Rabin [Ra60] and Hartmanis and Stearns [HS65] proved the existence of decidable problems with arbitrarily high computational complexity. They used a classical diagonalization, giving an algorithm which, by design, accepts a language differing from any language of low complexity. This technique allows us to conclude, for example, that there exist problems requiring exponential time. Problems constructed in this way are artificial, having no meaning or significance outside of their role in this theorem. Meyer and Stockmeyer [MS72] used this theorem to demonstrate natural problems provably outside of P by showing that they are complete for high complexity classes.

A priori, it might seem possible to use diagonalization to prove that some problem in NP requires exponential time. One would give a nondeterministic polynomial time Turing machine which, over the course of its inputs, has an opportunity to run each of the deterministic polynomial time Turing machines and arrange to accept a differing language. This simple argument fails because the nondeterministic machine, running in some fixed n^k time, is not able to simulate a deterministic machine whose time bound is a larger polynomial. Possibly there is a more clever nondeterministic simulation of the deterministic machine which would allow the nondeterministic machine to carry out the diagonalization process. The method of relativization, described below, suggests that this will not be straightforward.

In a relativized computation the machine is provided with a set, called an *oracle*, and the capability to determine membership in this set without cost. For each oracle, there is associated the relativized complexity class of languages efficiently computable in the presence of that oracle. Baker, Gill, and Solovay [BGS75] demonstrated an oracle relative to which P is equal to NP and another relative to which they differ. A step-by-step simulation of one machine by another, such as that used in a conventional diagonalization, still succeeds in the presence of an oracle, as the simulator needs only query the oracle whenever the simulated machine does. Any argument which relies only on step-by-step simulation to collapse or separate P and NP would thus also apply in the presence of any oracle. As this would contradict the BGS theorem, step-by-step simulation will not be sufficient to settle the question.

The above is a heuristic argument as the notion of "step-by-step simulation" is not precisely defined. In recent papers, Lund, Fortnow, Karloff, and Nisan, and Shamir [LFKN90, Sh90] introduced a new type of simulation to prove equivalence for the complexity classes IP and PSPACE. This simulation is not of the step-by-step sort, but is rather indirect, and indeed does not relativize to all oracles, as can be seen from the paper of Fortnow and Sipser [FS88]. Conceivably, an indirect simulation such as this could be combined with the diagonalization method to separate certain complexity classes.

3.2 Independence

Following the celebrated work of Gödel and Cohen we know that certain mathematical questions, such as the Continuum Hypothesis, cannot be answered within accepted formal systems of mathematical reasoning. There has been speculation that the P versus NP question may be unresolvable in a similar sense. This would mean that if there were no polynomial time algorithm for the clique problem, we would never be able to prove the nonexistence. On the other hand if such an algorithm were to exist, we would not be able to prove its performance. I will mention here some of the literature on independence, though I state at the outset my opinion that our current inability to resolve this question derives from insufficient brain power, rather than any lack of power in our formal systems.

In a sense, the work on relativization already suggests a sort of limited independence for the P versus NP question. One might be able to formulate an axiomatic system corresponding to pure recursive function theory, in which the results of Baker, Gill, and Solovay presumably would show that the question is unresolvable. However, this would be very far from establishing independence with respect to strong systems such as number theory or set theory.

In the direction of stronger independence, a number of papers have attempted to offer a beginning, notably: [HH76] giving an oracle under which the P versus NP question is independent of set theory and [Li78, DL79, Sa80, Bu86, CU88, BH91] establishing independence and related results within weak fragments of number theory or related systems. While I do not profess expertise in this area, I doubt whether the results obtained so far in this vein do more than raise the possibility of meaningful independence.

3.3 Expressibility in Logic

Another potential application of mathematical logic arises because it provides a different way of defining familiar complexity classes. Fagin [Fa74] showed that NP equals the class of languages representing structures definable by Σ_1^1 second-order sentences. He and Jones and Selman [JS74] showed that the class NEXPTIME equals the class of languages representing the collections of cardinalities of universes of such structures. Much earlier, Scholz [Sc52] had considered such collections and asked whether they could be characterized (see also Asser [As55]). Immerman [Im87] has extended the correspondence with finite model theory to P, NL, and other complexity classes, using first-order calculus adorned with various operators. Hope that this recoding into logic may help to settle questions in complexity theory has been partly born out by Immerman's discovery [Im88] that NL=coNL, inspired by certain observations concerning first-order definability following an earlier weaker result of [LJK87]. Szelepcsényi [Sz88] independently obtained NL=coNL at around the same time.

3.4 Restricted Systems

The difficulty in proving lower bounds on complexity stems from the richness of the class of algorithms. One approach that allows for partial progress is to restrict the class of algorithms considered. There is a substantial literature on bounds in various restricted models. These fall into two categories, which may be called *natural models* and *handicapped models*. In the former, a model is selected which allows only for operations specially tailored to the problem at hand. This includes sorting models, polynomial evaluation models, various models oriented towards data structures, and others. In the latter, we seek to maintain the generality of the model, but weaken it sufficiently so that an interesting lower bound is obtainable. Both approaches bear a connection to separating complexity classes. We will treat handicapped models shortly in the section on circuit complexity. Here we consider restricted proof systems as a form of natural model.

A *proof system* is a collection of axioms and rules of inference. *Resolution* is a proof system for showing that formulas are unsatisfiable. The axioms are the clauses of the formula. In the inference rule, if clauses $(x \wedge \alpha)$ and $(\overline{x} \wedge \beta)$ are already present, then the new clause $(\alpha \wedge \beta)$ may be added. Unsatisfiability is proved when the empty clause is derived. The complexity of the proof is the number of clauses added. Any unsatisfiable formula has a resolution proof. Haken, solving a question that had been open for many years, proved that there are formulas requiring exponential size proofs [Ha85]. His work followed that of Tseitin [Ts62] who had solved regular resolution, a further restricted form (see also [Ga77]). Extended resolution, a generalized form, remains open (see [Co75]). Kozen [Ko77] considered other proof systems and established lower bounds for them.

Proof systems are related to the class NP. If all unsatisfiable formulas had polynomial length resolution proofs then it would easily follow that NP=coNP. Haken's result proves a special case of the NP$\neq$coNP conjecture, showing that one particular nondeterministic algorithm is superpolynomial.

3.5 Circuit Complexity

Shannon [Sh49] proposed the size of Boolean circuits as a measure of the complexity of functions. Savage [Sa72] demonstrated a close relationship between circuit size and time on a Turing machine. Circuits are an appealing model of computation for proving lower bounds. Their relatively simple definition renders them more amenable to combinatorial analysis and allows for natural variations and restrictions. This has been key to achieving a number of important results. Many researchers consider circuit complexity to be the most viable direction for resolving the P versus NP question. We will survey recent work in this area, with an eye towards this goal. The survey paper of Boppana and Sipser [BS90] contains details of much of this work, described in a readable fashion. Dunne [Du88] and Wegener [We87] have recently written books on Boolean complexity.

A *circuit* (or *straight line program*) over a basis (typically AND, OR and NOT) is a sequence of instructions, each producing a function by applying an operation from the basis to previously obtained functions. Initially, we start out with the functions naturally associated with each of the input variables. If each function is used at most once, then the circuit is called a *formula*.

By Savage's theorem, any problem in P has a polynomial size family of circuits. Thus, to show that a problem is outside of P it would suffice to show that its circuit complexity is superpolynomial. A simple counting argument shows that most Boolean functions on n variables require exponential circuit size [Sh49, Mu56]. The best lower bound we have to date for a problem in NP is $3n - o(n)$, due to N. Blum [Bl84]. It does not seem likely that the techniques used to get this result will extend to significantly better bounds.

In [Va90], Valiant suggests that a direct attempt to prove lower bounds on the size of Boolean circuits may not be the right approach. There are natural algebraic generalizations of certain NP-complete problems, such as the Hamiltonian cycle problem, to an arbitrary ring. We may also generalize circuits over the basis AND, XOR for such problems to arbitrary rings. Circuits which solve the generalized problem form a proper subclass of the circuits which merely work correctly in the Boolean case. As such it may be easier to prove strong lower bounds in this restricted case. Valiant points out that our failure to do this even then argues poorly for our chances in the Boolean case. He suggests directing our energies toward solving the algebraic case first.

There are merits to this line or reasoning. One of the impediments in the lower bounds area is a shortage of problems of intermediate difficulty which lend insight into the harder problems. The algebraic generalizations may be important steps towards the Boolean goal. On the other hand, I do not believe, as Valiant argues, that the algebraic case is prerequisite. True, the algebraic case is more restrictive, and hence "easier." The same may be said for uniform models of computation versus nonuniform models. Nonuniform models, such as circuits, are more powerful than uniform models, such as Turing machines, because the algorithm used may change arbitrarily depending upon the length of the input. Accordingly it may be more difficult to prove lower bounds on these more powerful models. Nevertheless, the "harder" problem may allow one to see more easily the heart of the matter, unobscured by unnecessary features.

3.6 Bounded Depth Circuits

The next few sections treat handicapped variants of the circuit model. By placing various restrictions on structure, such as limiting the depth or restricting the basis, it has been possible to obtain several strong lower bounds. The first result of this kind is due to Furst, Saxe, and Sipser [FSS84], and independently Ajtai [Aj83], who proved that certain simple functions, such as the parity function, require superpolynomial circuit size if the depth is held fixed.

These two proofs of this theorem use different, though related, methods. Furst, Saxe, and Sipser introduced the notion of *probabilistic restriction*, a randomly selected assignment to some of the variables. They showed that for any circuit of the form AND of small ORs, the circuit induced by a random restriction may likely be put in the form OR of small ANDs. This interchange allows adjacent levels of ORs to follow one another and therefore to merge. A circuit may thus be simplified inductively, while preserving the fact that it computes a parity function. Ajtai's argument is also an induction. He showed that any set definable by a low depth circuit is well approximated by a set of a very special form, a disjoint union of large cylinders. A *cylinder* is a maximal collection of strings which all agree on some collection of variables. Clearly the parity function cannot be approximated in this way. His main combinatorial lemma shows that the property of being well approximable is closed under polynomial union and intersection.

Yao [Ya85] combined probabilistic restriction with a type of approximation to give an exponential lower bound for constant depth parity circuits. Håstad [Hå86] simplified and strengthened this bound by improving the core of the FSS argument using a type of analysis of conditional probabilities developed by Yao.

Sipser [Si83] showed that there are functions computable by depth d linear size circuits and which require superpolynomial size for depth $d-1$, by a technique very similar to that used by FSS. Yao claimed without proof an exponential bound for this depth hierarchy. Using his simplified method, Håstad presented a proof of this result.

Razborov [Ra87] introduced a method of obtaining lower bounds on the size of limited depth circuits over a larger basis. He proved that the majority function requires exponential size for circuits having PARITY gates as well as AND, OR, and NOT gates. He showed that any set definable with a small, shallow circuit over this basis is well approximated by a polynomial of low degree over the two element field, and argues that the majority function cannot be well approximated in this way. Barrington [Ba86a] extended this to circuits with MOD_q gates for any prime q instead of only PARITY (*i.e.,* MOD_2) gates. Smolensky [Sm87] further simplified and improved this, showing that computing the MOD_p function requires exponential size when computed by shallow circuits having AND, OR, and MOD_q gates, for p and q powers of different primes.

3.7 Monotone Circuit Size

A *monotone* circuit is one having AND and OR gates but no negations. A function is *monotone* if $x \leq y$ implies $f(x) \leq f(y)$ under the usual Boolean ordering. It is easy to see that monotone circuits compute exactly the class of monotone functions.

The first strong lower bound concerning this model is due to Razborov [Ra85a], who proved that the clique function has superpolynomial monotone complexity. Shortly thereafter Andreev [An85], using similar methods, proved an exponential lower bound, further tightened by Alon and Boppana [AB87].

Razborov's theorems on monotone and bounded depth circuits, as well as the aforementioned proof of Ajtai, rely upon a technique which has come to be called the *approximation method.* One of the difficulties in attempting to analyze the computation of a circuit (restricted or general) for the purposes of proving lower bounds is that the subfunctions computed by the circuit may be complicated and hard to grasp. In the approximation method we show that the subfunctions are in a certain sense near functions which are much simpler. Take, for example Razborov's lower bound for the clique function. Consider any monotone circuit computing this function. The plan is to adjust the result of each of the AND and OR gates in such a way that (1) each adjustment alters the result only slightly, while (2) the adjusted output is far from the clique function. This means that there must have been many adjustments and hence many gates.

If general circuits computing monotone functions could be converted into monotone circuits with only a polynomial increase in size then strong monotone lower bounds would yield separations of complexity classes in general complexity theory. Razborov [Ra85b] showed that this is not true when he used the approximation to show that the that the problem of testing for a perfect matching requires superpolynomial size monotone circuits. As this problem is in P [Ed65], we know that it has polynomial size non-monotone circuits. Using a similar method, Tardos [Ta88] showed that there is a monotone problem in P which requires exponential size monotone circuits.

There is a class of functions where the monotone and non-monotone complexities are polynomially related. These are the slice functions, introduced by Berkowitz [Be82]. A function f is a *slice function* if for some k the value of $f(x)$ is 0 when the number of 1s in x is fewer than k, and the value is 1 when the number of 1s in x is more than k. He showed that any general circuit computing a slice function may be converted into a monotone function by adding only a polynomial number of gates. Hence if one were able to prove a strong lower bound on the monotone circuit complexity of a slice function then this would show that P≠NP.

3.8 Monotone Circuit Depth

Circuits over the standard basis restricted to fan-in two with $O(\log n)$ depth form an important subclass of the polynomial size circuits. Such circuits are equivalent in power to polynomial size formulas. Proving that a language is not definable with a circuit of this kind would show that it is not in the class NC^1. This would be weaker than showing it is outside of P, but we are still unable to do this for any language in NP.

Karchmer and Wigderson [KW88] gave a very nice characterization of languages definable with such circuits, which may be useful in obtaining lower bounds on depth. Let A be such a language. In the *communication game* for A, there are two players, one having a string in A and the other having a string of the same length not in A. The players communicate with each other to find a position where their two strings differ. The minimum number of bits that they require to do this over all strings of length n is the complexity of the game. Karchmer and Wigderson show that this exactly equals the minimum circuit depth necessary for this language.

A monotone variant of this game has played an important role in the discovery of lower bounds on monotone circuit depth. If one demands in addition that the found position is 1 for the string in A, then the complexity of the game is the minimum monotone circuit depth. This characterization of monotone depth complexity led them to a $\Omega(\log^2 n/\log\log n)$ lower bound on the monotone depth for the st-connectivity function. This was subsequently improved to $\Omega(\log^2 n)$ by Boppana [BS90] who avoided the communication game by arguing directly on the monotone circuit (independently obtained by Håstad [unpublished]). Raz and Wigderson [RW90] used the communication game to obtain a linear depth lower bound for monotone circuits computing the matching function. Their proof uses a lower bound on the probabilistic communication complexity of the set disjointness problem due to Kalyanasundaram and Schnitger [KS87] (simplified by Razborov [Ra90a]). Because of this it seems that it will be difficult to phrase the Raz-Wigderson argument without appealing to the communication game. Razborov [Ra90b] and Goldmann and Håstad [GH89] obtained additional lower bounds using the communication game.

Grigni and Sipser [GS90] point out that many of the results in monotone complexity can be viewed in terms of monotone analogs, mP, mNP, etc., of the familiar complexity classes P, NP, AC^k, NC^k, and others. Thus we already know that mP≠mNP [Ra85a], mP≠ {P ∩ **mono**} [Ra85b], and mNC^1≠mNL [KW88]. We use **mono** to designate the class of all monotone languages. Grigni and Sipser demonstrate that mNL≠m-coNL using the technique of Karchmer and Wigderson. This shows the inherently nonmonotone nature of the Immerman-Szelepcseny simulation. In a later paper [GS91], they strengthen the KW result by separating mNC^1 from mL. There are a number of open questions in monotone complexity, for example, whether Barrington's beautiful simulation of log-depth circuits by polynomial-size, width 5, branching programs, is inherently nonmonotone (see [GS90]).

3.9 Active Research Directions

In this section we will examine a few directions which appear to be plausible means of separating complexity classes. These are the *approximation method*, which has been used successfully in analyzing restricted circuit models; the *function composition method*, which may be useful to prove functions require large circuit depth; and an *infinite analog*, which has led to lower bounds in the bounded depth circuit model.

There is a possibility that the approximation method, which has been used so successfully to obtain lower bounds for monotone and bounded depth circuits, may apply in the general case as well. In [Ra89], Razborov considers this question and gives both a positive and negative answer. He shows that for one formalization of the method, it is too weak to give interesting bounds. On the other hand a generalization of the method is, in principle, strong enough, though there are much greater technical difficulties when applying it. Roughly speaking, these two versions of the approximation method differ in the way the class of approximating functions are chosen. In the weaker version, the class is selected in advance and applies to all circuits. In the stronger version the class depends upon the circuit.

Karchmer, Raz, and Wigderson [KRW91] proposed a direction for investigating the NC^1 versus P question. Let B_n denote the set of all Boolean functions on n variables. Given $f \in B_n$ and $g \in B_m$ we define the composition $f \circ g : \{0,1\}^{nm} \to \{0,1\}$ by

$$f \circ g(\vec{X}_1, ..., \vec{X}_n) = f(g(\vec{X}_1), ..., g(\vec{X}_n))$$

where $\vec{X}_i \in \{0,1\}^m$ for $i = 1, ..., n$. It is clear that $d(f \circ g) \leq d(f) + d(g)$. KRW conjecture that these two quantities are close to each other and argue that, if so, $\mathrm{NC}^1 \neq \mathrm{P}$. Edmonds, Impagliazzo, Rudich and Sgall [EIRS91] (and slightly later, Håstad and Wigderson [HW91] in a somewhat different way) proved a weak form of this conjecture.

In [Si81], I suggest a way to use ideas from descriptive set theory (Moschovakis [Mo80] has an excellent text on this subject) to gain insight into certain problems in circuit complexity theory. This stems from a proposed analogy between polynomiality and countability. A justification for this analogy is that uncountability and superpolynomiality often have a common origin: the power set operation. In this way certain problems in circuit complexity have infinitary analogs, often easier to solve. When this occurs, it may lead to an approach in the finitary case.

A successful application of this strategy occurred in proving the lower bound of Furst, Saxe, and myself, mentioned earlier [FSS84]. The infinite analog to the theorem that there are no polynomial-size, depth d circuits computing the parity function on n variables, is the theorem that there are no countable-size, depth d, circuits computing a parity function on ω many variables. (A Boolean function on ω many Boolean variables is a *parity function* if the output changes whenever any single input changes.) The proof of the infinitary theorem did precede and motivate the proof in the finite case. The two proofs are very similar in structure. A second, related application of this analogy appears in the constant-depth hierarchy theorem [Si83].

This same approach suggests an infinite analog to the class NP: the analytic sets (also called Σ^1_1 sets). The classical theorem due to Lebesgue [Le05] stating that the there is an analytic set whose complement is not analytic may thus be taken as an infinite analog to the statement that NP$\neq$coNP. Lebesgue's proof is a diagonalization, and it does not seem likely that it has a finite counterpart. In [Si84] I give a different proof of Lebesgue's theorem. This new proof does not rest upon the notion of universal set, essential to the diagonalization in Lebesgue's proof. It offers more information of a combinatorial nature that may be useful in the finite case.

Acknowledgments

I wish to thank Mauricio Karchmer and Alexander Razborov who read an earlier draft of this paper and offered several suggestions and corrections.

I am grateful to Juris Hartmanis for providing me with a copy of Gödel's letter and to Sorin Istrail and Arthur S. Wensinger for it's translation. It appears here with the permission of the Institute for Advanced Study.

Finally, it is wonderful to thank my wife Ina, for putting up with the writing of this paper as our daughter Rachel was being born, and for other assistance at critical moments.

Appendix: Historical Quotes

Shannon, *discussing circuit design methods [Sh49]:*

The problem of synthesizing non-series-parallel circuits is exceedingly difficult. It is even more difficult to show that a circuit found in this way is the *most* economical one to realize a given function. The difficulty springs from the large number of essentially different networks available and more particularly from the lack of a simple mathematical idiom for representing these circuits.

von Neumann, *presenting an algorithm for solving the assignment problem [Vo53]:*

It turns out that this number [of steps required by the algorithm] is a moderate power of n, i.e., considerably smaller than the "obvious" estimate $n!$ mentioned earlier.

Yablonski, *discussing alternatives to perebor in designing circuits [Ya59a]:*

At present there is an extensive field of problems in cybernetics where the existence of certain objects or facts may be established quite trivially and, within the limits of the classical definition of algorithms, completely effectively, yet a solution is, in practice, often impossible because of its cumbersome nature. Such, for example, are some of the problems involving information coding and the analysis and synthesis of circuits. It is here that the necessity of making the classical definition an algorithm more precise naturally arises. It is to be expected that this will, to a greater extent than at present, take into account the peculiarities of certain classes of problems, and may, possibly, lead to such developments in the concept of algorithm that different types of algorithms will not be comparable. At the moment it is too early to make predictions of how the notion of an algorithm may be modified, for we have, as yet, very little idea of how the various classes of problems should be specified. In the present article we attempt to explore the algorithmic difficulties arising in the solution of cybernetic problems which, while admitting of a trivial solution, on the basis of the classical definition of an algorithm, are not practically solvable because of the massive nature of that solution.

Cobham, *studying why certain problems, such a multiplication, are computationally more difficult than others, such as addition [Co64]:*

Thus the process of adding m and n can be carried out in a number of steps bounded by a linear polynomial in $l(m)$ and $l(n)$. Similarly, we can multiply m and n in a number of steps bounded by a quadratic polynomial in $l(m)$ and $l(n)$. So, too, the number of steps involved in the extraction of square roots, calculation of quotients, etc., can be bounded by polynomials in the lengths of the numbers involved, and this seems to be a property of simple function in general. This suggests that we consider the class, which I will call $\mathcal{L}$, of all functions having this property.

For several reasons the class $\mathcal{L}$ seems a natural one to consider. For one thing, if we formalize the above definition relative to various general classes of computing machines we seem always to end up with the same well-defined class of functions. Thus we can give a mathematical characterization of $\mathcal{L}$ having some confidence that it characterizes correctly our informally defined class.

Edmonds, *a section marked "Digression" in a paper giving a polynomial time algorithm for the maximum matching problem [Ed65]:*

An explanation is due on the use of the words "efficient algorithm." First, what I present is a conceptual description of an algorithm and not a particular formalized algorithm or "code."

For practical purposes computational details are vital. However, my purpose is only to show as attractively as I can that there is an efficient algorithm. According to the dictionary, "efficient" means "adequate in operation or performance." This is roughly the meaning I want — in the sense that it is conceivable for maximum matching to have no efficient algorithm. Perhaps a better word is "good."

I am claiming, as a mathematical result, the existence of a *good* algorithm for finding a maximum cardinality matching in a graph.

There is an obvious finite algorithm, but that algorithm increases in difficulty exponentially with the size of the graph. It is by no means obvious whether *or not* there exists an algorithm whose difficulty increases only algebraically with the size of the graph.

The mathematical significance of this paper rests largely on the assumption that the two preceding sentences have mathematical meaning. I am not prepared to set up the machinery necessary to give them formal meaning, nor is the present context appropriate for doing this, but I should like to explain the idea a little further informally. It may be that since one is customarily concerned with existence, convergence, finiteness, and so forth, one is not inclined to take seriously the question of the existence of a *better-than-finite* algorithm.

The relative cost, in time or whatever, of the various applications of a particular algorithm is a fairly clear notion, at least as a natural phenomenon. Presumably, the notion can be formalized. Here "algorithm" is used in the strict sense to mean the idealization of some physical machinery which gives a definite output, consisting of cost plus the desired result, for each member of a specified domain of inputs, the individual problems.

The problem-domain of applicability for an algorithm often suggests for itself possible measures of size for the individual problems — for maximum matching, for example, the number of edges or the number of vertices in the graph. Once a measure of problem-size is chosen, we can define $F_A(N)$ to be the least upper bound on the cost of applying algorithm A to problems of size N.

When the measure of problem-size is reasonable and when the sizes assume values arbitrarily large, an asymptotic estimate of $F_A(N)$ (let us call it *the order of difficulty of algorithm* A) is theoretically important. It cannot be rigged by making the algorithm artificially difficult for smaller sizes. It is one criterion showing how good the algorithm is — not merely in comparison with other given algorithms for the same class of problems, but also on the whole how good in comparison with itself. There are, of course, other equally valuable criteria. And in practice this one is rough, one reason being that the size of a problem which would ever be considered is bounded.

It is plausible to assume that any algorithm is equivalent, both in the problems to which it applies and in the costs of its applications, to a "normal algorithm" which decomposes into elemental steps of certain prescribed types, so that the costs of the steps of all normal algorithms are comparable. That is, we may use something like Church's thesis in logic. Then, it is possible to ask: Does there or does there not exist an algorithm of given order of difficulty for a given class of problems?

One can find many classes of problems, besides maximum matching and its generalizations, which have algorithms of exponential order but seemingly none better. An example known to organic chemists is that of deciding whether two given graphs are isomorphic. For practical purposes the difference between algebraic and exponential order is often more crucial than the difference between finite and non-finite.

It would be unfortunate for any rigid criterion to inhibit the practical development of algorithms which are either not known or known not to conform nicely to the criterion. Many of the best algorithmic ideas known today would suffer by such theoretical pedantry. In fact, an outstanding open question is, essentially: "how good" is a particular algorithm for linear programming, the simplex method? And, on the other hand, many important algorithmic ideas in electrical switching theory are obviously not "good" in our sense.

However, if only to motivate the search for good, practical algorithms, it is important to realize that it is mathematically sensible even to question their existence. For one thing the task can then be described in terms of concrete conjectures.

Fortunately, in the case of maximum matching the results are positive. But possibly this favourable position is very seldom the case. Perhaps the twoness of edges makes the algebraic order for matching rather special for matching in comparison with the order of difficulty for more general combinatorial extremum problems.

Edmonds, *discussing the matroid partition problem [Ed65b]:*

We seek a good characterization of the minimum number of independent sets into which the columns of a matrix of Π can be partitioned. As the criterion of "good" for the characterization we apply the "principle of the absolute supervisor." The good characterization will describe certain information about the matrix which the supervisor can require his assistant to search out along with a minimum partition and which the supervisor can then use "with ease" to verify with mathematical certainty that the partition is indeed minimum. Having a good characterization does not mean necessarily that there is a good algorithm. The assistant may have to kill himself with work to find the information and the partition.

Rabin, *in a survey paper on automata theory [Ra66]:*

We shall consider an algorithm to be practical if, for automata with n states, it requires at most cn^k (k is a fixed integer and c a fixed constant) computational steps. This stipulation is, admittedly, both vague and arbitrary. We do not, in fact cannot, define what is meant by a computational step, thus have no precise and general measure for the complexity of algorithms. Furthermore, there is no compelling reason to classify algorithms requiring cn^k steps as practical.

Several points may be raised in defense of the above stipulation. In every given algorithm the notion of a computational step is quite obvious. Hence there is not that much vagueness about the measure of complexity of existing algorithms. Another significant pragmatic fact is that all existing algorithms either require up to about n^4 steps or else require 2^n or worse steps. Thus drawing the line of practicality between algorithms requiring n^k steps and algorithms for which no such bound exists seems to be reasonable.

Cook, *considering the complexity of theorem proving [Co71]. The class P is denoted by $\mathcal{L}^*$:*

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

Levin, *from Trakhtenbrot's survey on perebor [Le73, Tr84]:*

This is the situation with so called exhaustive search problems, including: the minimization of Boolean functions, the search for proofs of finite length, the determination of the isomorphism of graphs, etc. All of these problems are solved by trivial algorithms entailing the sequential scanning of all possibilities. The operating time of the algorithm is, however, exponential, and mathematicians nurture the conviction that it is impossible to find simpler algorithms.

Gödel, *in a letter to von Neumann. Translated by Arthur S. Wensinger with Sorin Istrail's guidance. The original letter is in the* Manuscript Division of the Library of Congress, Washington, D.C. *The translator had access to Juris Hartmanis' article [Ha89]. Discussions* *with Hartley Rogers and Philip Scowcroft provided contributions to the translation:*

Dear Mr. von Neumann,

To my great regret I have heard about your illness. The news reached me most unexpectedly. Morgenstern had, to be sure, told me in the summer about some infirmity you had been suffering, but at the time he thought that no major significance was to be attached to it. I hear that you have been undergoing radical treatment in the past several months, and I am happy[2] that this has achieved the desired results and that things are now going better for you. I hope and wish that your condition continues to improve and that the latest medical advances, if possible, can lead to a complete recovery.

Since I have heard that you are feeling stronger now, I take the liberty[3] of writing to you about a mathematical problem about which your views would interest me greatly. It is evident that one can easily construct a Turing machine which, for each formula F of the predicate calculus[4] and for every natural number n, will allow one to decide if F has a proof of length n [length = number of symbols]. Let $\Psi(F, n)$ be the number of steps that the machine requires for that and let $\varphi(n) = \max_F \Psi(F, n)$. The question is, how fast does $\varphi(n)$ grow for an optimal machine. One can show that $\varphi(n) \geq Kn$. If there actually were a machine with $\varphi(n) \sim Kn$ (or even only[5] with $\sim Kn^2$), this would have consequences of the greatest magnitude[6]. That is to say, it would clearly indicate that, despite the unsolvability of the Entscheidungsproblem, the mental effort of the mathematician in the case of yes-or-no questions could be completely (**Gödel's footnote**: apart from the postulation of axioms) replaced by machines. One would indeed have to simply select an n so large[7] that, if the machine yields no result, there would then also be no reason to think further about the problem. Now, it seems to me, however, to be totally within the realm of possibility that $\varphi(n)$ grows slowly.[8] For 1.) it seems that $\varphi(n) \geq Kn$ is the only estimate that can be derived from a generalization of the proof for the unsolvability of the Entscheidungsproblem; 2.) $\varphi(n) \sim Kn$ (or $\sim Kn^2$) means, of course, simply that the number of steps vis-à-vis *dem blossen Probieren*[9] can be reduced from N to $\log N$ (or $(\log N)^2$)[10] Such strong reductions[11] do indeed occur, however, in the case of other finite problems, e.g., in the case of calculating a quadratic residue[12] by means of repeated application of the law of reciprocity. It would be interesting to know, for example, what the situation is in the case of determining whether a number is a prime number, and in the case of finite combinatorial problems, how strongly[13] *in general* the number of steps vis-à-vis the blossen Probieren can be reduced.

I wonder if you have heard that Post's problem (as to whether there are degrees of unsolvability among the problems $(\exists y)\varphi(y, x)$, where φ is recursive) has been solved in the positive by a very young man named Richard Friedberg. The solution is very elegant[14]. Unfortunately, Friedberg does not want to pursue graduate work in mathematics, but in medicine (evidently under the influence of his father).

What is your opinion by the way, of the attempts recently in vogue again to base Analysis[15] on ramified type-theory[16] ? You are probably aware that in connection with this, Paul Lorenzen has progressed as far as Lebesgue's Measure Theory. But I believe that in important aspects of Analysis non-predicative methods of proof cannot be eliminated.

I would be very happy to hear from you personally; and please let me know if I can do anything at all for you.

With best greetings and wishes, also to your wife[17],
Your very devoted,
Kurt Gödel

P.S. I congratulate you heartily on the ...[18]

[2] to hear

[3] lit.: I should like to allow myself

[4] des engeren Funktionenkalküls, lit.: of the narrower function-calculus

[5] It is not completely clear if the word is "nur": only, or "nun": now; however, "only" seems to fit the context better.

[6] "Tragweite" suggests great breadth of range or impact, lit.: carrying power far into many fields or areas where it would have its effect.

[7] or: large enough

[8] Gödel says "so langsam wächst," which does not mean "grows so slowly," but more something like "accordingly grows slowly," where "accordingly" means "considering the type of growth of $\varphi(n)$ I have been discussing."

[9] lit.: "the simply testing or trying out"; "exhaustive search" or "brute force" are the currently established terms

[10] The exponent here is possibly an "n."

[11] Verringerungen

[12] Restsymbol

[13] severely, radically; "how much" is not quite adequate as a translation

[14] It is worth remarking that this work was done by Friedberg in his undergraduate senior honors thesis.

[15] Gödel uses the English word "Analysis" here, not the German term "Analytik."

[16] verzweigte Typentheorie

[17] "an ihre Frau Gemahlin" is an unusually polite and old-fashioned formula, quite in accord with the formal tone of this letter to a man clearly highly esteemed by Gödel.

[18] ... ? The balance of the postscript is missing from the photocopy of the document available to us.

References

[Aj83] M. Ajtai, Σ_1^1-formulae on finite structures, *Annals of Pure and Applied Logic* 24, 1–48, 1983.

[AB87] N. Alon and R. B. Boppana, The monotone circuit complexity of Boolean functions, *Combinatorica* 7:1, 1–22, 1987.

[An85] A. E. Andreev, On a method for obtaining lower bounds for the complexity of individual monotone functions, *Doklady Akademii Nauk SSSR* 282:5, 1033–1037 (in Russian). English translation in *Soviet Mathematics Doklady* 31:3, 530–534, 1985.

[As55] G. Asser, Das Repräsentatenproblem im Prädikatenkalkül der ersten Stufe mit identität, *Zeitschrift für Matemitische Logic und Grundlagen der Mathematik*, 1, 252–263, 1955.

[BGS75] T. P. Baker, J. Gill, and R. Solovay, Relativizations of the P=?NP question, *SIAM Journal on Computing* 4:4, 431-442, 1975.

[Ba86a] D. A. Barrington, Bounded-width polynomial-size branching programs recognize exactly those languages in NC^1, *Journal of Computer and System Sciences 38*, 150–164, 1986.

[Ba86a] D. A. Barrington, A note on a theorem of Razborov, manuscript, 1986.

[BH91] S. Ben-David and S. Halevi, On the Independence of P versus NP, Tech. Report #699, Technion, 1991.

[Be82] S. J. Berkowitz, On some relationships between monotone and non-monotone circuit complexity, Technical Report, Computer Science Department, University of Toronto, 1982.

[Bl67] M. Blum, A machine-independent theory of the complexity of recursive functions, *Journal of the ACM, 14*, no. 2, 322–336, 1967.

[Bl84] N. Blum, A Boolean function requiring $3n$ network size, *Theoretical Computer Science* 28, 337–345, 1984.

[Bo86] R. B. Boppana, Threshold functions and bounded depth monotone circuits, *Journal of Computer and System Sciences* 32:2, 222–229, 1986.

[BS90] R. B. Boppana and M. Sipser, The complexity of finite functions, *Handbook of Theoretical Computer Science*, Elsevier, 758–804, 1990.

[Bu86] S. R. Buss, *Bounded Arithmetic*, Bibliopolis, 1986.

[Co64] A. Cobham, The intrinsic computational difficulty of functions, *Proc. of the 1964 International Congress for Logic, Methodology, and the Philosophy of Science*, edited by Y. Bar-Hillel, North-Holland, 24–30, 1964.

[CSV84] A. K. Chandra, L. J. Stockmeyer, and U. Vishkin, Constant depth reducibility, *SIAM Journal on Computing* 13:2, 423–439, 1984.

[Co71] S. A. Cook, The complexity of theorem proving procedures, *Proceedings of the 3rd Annual ACM Symposium on Theory of Computing*, 151–158, 1971.

[Co75] S. A. Cook, Feasably constructive proofs and the propositional calculus, *Proceedings of the 7th Annual ACM Symposium on Theory of Computing*, 83–97, 1975.

[CU88] S. A. Cook and A. Urquhart, Functional interpretations of feasibly constructive arithmetic, Technical Report 210/88, Computer Science Department, University of Toronto, 1988.

[DL79] R. E. DeMillo and R. J. Lipton, The consistency of P=NP and related problems within fragments of number theory, *Proceedings of the 11th ACM Symposium on Theory of Computing*, 153–159, 1979.

[Du88] P. E. Dunne, *The Complexity of Boolean Networks*, Academic Press, 1988.

[Ed65] J. Edmonds, Paths, trees, and flowers, *Canadian Journal of Mathematics*, 17, 449–467, 1965.

[Ed65b] J. Edmonds, Minimum partition of a matroid into independent sets, *Journal of Research of the National Bureau of Standards (B)*, 69, 67–72, 1965.

[EIRS91] J. Edmonds, R. Impagliazzo, S. Rudich, J. Sgall, Communication complexity towards lower bounds on circuit depth, *Proceedings of the 32nd Annual Symposium on Foundations of Computer Science*, 249–257, 1991.

[Fa74] R. Fagin, Generalized first-order spectra and polynomial-time recognizable sets, *Complexity of Computation, SIAM-AMS Proceedings 7*, 43–73, 1974.

[FS88] L. Fortnow and M. Sipser, Are there interactive protocols for co-NP languages? *Information Processing Letters 28*, 249–251, 1988.

[FSS84] M. Furst, J. Saxe, and M. Sipser, Parity, circuits and the polynomial time hierarchy, *Mathematical Systems Theory 17*, 13–27, 1984.

[Ga77] Z. Galil, On the complexity of regular resolution and the Davis-Putnam procedure, *Theoretical Computer Science 4*, 23–46, 1977.

[GJ79] M. Garey and D. S. Johnson, *Computers and Intractability: a guide to the theory of NP-completeness*, Freeman, 1979.

[GH89] M. Goldmann, J. Håstad, A lower bound for monotone clique using a communication game, *manuscript*, 1989.

[GS90] M. Grigni and M. Sipser, Monotone complexity, *Proceedings of LMS workshop on Boolean Function Complexity*, Durham, edited by M. S. Paterson, Cambridge University Press, 1990.

[GS91] M. Grigni and M. Sipser, Monotone separation of NC^1 from logspace, *Proceedings of the 6th Annual Symposium on Structure in Complexity Theory,* 294–298, 1991.

[HS72] L. H. Harper and J. E. Savage, The complexity of the marriage problem, *Advances in Mathematics* 9, 299–312, 1972.

[Ha89] J. Hartmanis, Gödel, von Neumann and the P=?NP Problem, *Bulletin of the European Association for Theoretical Computer Science*, 101–107, June 1989.

[HH76] J. Hartmanis and J. Hopcroft, Independence results in computer science, *ACM SIGACT News 8*, no. 4, 13–24, 1976.

[HS65] J. Hartmanis and R. E. Stearns, On the computational complexity of algorithms, *Transactions of the AMS* 117, 285–306, 1965.

[Ha85] A. Haken, The intractability of resolution, *Theoretical Computer Science 39*, 297–308, 1985.

[Hå86] J. Håstad, *Computational Limitations for Small Depth Circuits*, MIT Press, 1986.

[HW91] J. Håstad, A. Wigderson, Composition of the Universal Relation, manuscript, 1991.

[HU79] J. E. Hopcroft and J. D. Ullman, *Introduction to Automata Theory, Languages, and Computation*, Addison-Wesley, 1979.

[Im87] N. Immerman, Languages that capture complexity classes, *SIAM Journal on Computing* 16:4, 760–778, 1987.

[Im88] N. Immerman, Nondeterministic space is closed under complement, *SIAM Journal on Computing*, 935–938, 1988.

[JS74] N. D. Jones and A. L. Selman, Turing machines and the spectra of first-order formulas, *Journal of Symbolic Logic 39*, 139–150, 1974.

[KS87] B. Kalyanasundaram and G. Schnitger, The probabilistic communication complexity of set intersection, *Proceedings of the Second Annual Conference on Structure in Complexity Theory*, 41–49, 1987.

[KRW91] M. Karchmer, R. Raz, A. Wigderson, On proving super-logarithmic depth lower bounds via the direct sum in communication complexity, *Proceedings of the Sixth Annual Conference on Structure in Complexity Theory,* 1991.

[KW88] M. Karchmer and A. Wigderson, Monotone circuits for connectivity require super-logarithmic depth, *Proceedings of the 20th Annual ACM Symposium on Theory of Computing*, 539–550, 1988.

[Ka72] R. M. Karp, Reducibility among combinatorial problems, *Complexity of Computer Computations*, Plenum Press, 85–103, 1972.

[Ko77] D. Kozen, Lower bounds for natural proof systems, *Proceedings of the 18th Annual Symposium on Foundations of Computer Science*, 254–266, 1977.

[LJK87] K. J. Lange, B. Jenner, B, Kirsig, The logarithmic hierarchy collapses: $A\Sigma_2^L = A\Pi_2^L$, *Proceedings of the 14th International Conference on Automata*, Languages, and Programming, 1987.

[Le05] H. Lebesgue, Sur les fonctions représentables analytiquement, *Journal de Math.* 6^e *serie 1*, 139–216, 1905.

[Le73] L. Levin, Universal sequential search problems, *Probl. Pered. Inform. IX 3*, 1973; translation in *Problems of Information Trans 9*, 3, 265–266; corrected translation in Trakhtenbrot [Tr84].

[Li78] R. J. Lipton, Model theoretic aspects of complexity theory, *Proceeding of the 19th Annual Symposium on Foundations of Computer Science*, 193–200, 1978.

[LFKN90] C. Lund, L. Fortnow, H. Karloff, and N. Nisan, Algebraic methods for interactive proof systems, *Proceedings of 22th Annual ACM Symposium on Theory of Computing*, 2–10, 1990; to appear in the *Journal of the ACM.*

[MS72] A. R. Meyer, and L. J. Stockmeyer, The equivalence problem for regular expressions with squaring requires exponential time, *Proceedings of the 13th Annual Symposium on Switching and Automata Theory*, 125–129, 1972.

[Mo80] Y. N. Moschovakis, *Descriptive Set Theory*, North-Holland, 1980.

[Mu56] D. E. Muller, Complexity in electronic switching circuits, *IRE Transactions on Electronic Computers* 5, 15–19, 1956.

[Pa76] M. S. Paterson, An introduction to Boolean function complexity, *Astérisque 38–39*, 183–201, 1976.

[Ra60] M. O. Rabin, Degree of difficutly of computing a function and a partial ordering of recursive sets, Tech. Rep. No. 2, Hebrew University, 1960.

[Ra66] M. O. Rabin, Mathematical theory of automata, *Proceedings of 19th ACM Symposium in Applied Mathematics*, 153–175, 1966.

[RW90] R. Raz and A. Wigderson, Monotone circuits for matching require linear depth, *Proceedings of the 22nd Annual ACM Symposium on Theory of Computing*, 287–292, 1990.

[Ra85a] A. A. Razborov, A lower bound on the monotone network complexity of the logical permanent, *Matematicheskie Zametki* 37:6, 887–900 (in Russian). English translation in *Mathematical Notes of the Academy of Sciences of the USSR* 37:6, 485–493, 1985.

[Ra85b] A. A. RAZBOROV, A lower bound on the monotone network complexity of the logical permanent, *Matematicheskie Zametki* 37:6, 887–900 (in Russian). English translation in *Mathematical Notes of the Academy of Sciences of the USSR* 37:6, 485–493, 1985.

[Ra87] A. A. RAZBOROV, Lower bounds on the size of bounded depth networks over a complete basis with logical addition, *Matematicheskie Zametki* 41:4, 598–607 (in Russian). English translation in *Mathematical Notes of the Academy of Sciences of the USSR* 41:4, 333–338, 1987.

[Ra89] A. A. RAZBOROV, On the method of approximations, *Proceedings of the 21st Annual ACM Symposium on Theory of Computing*, 168–176, 1989.

[Ra90a] A. A. RAZBOROV, On the distributional complexity of disjointness, *Proceeding of the International Conference on Automata, Languages, and Computation*, 1990; to appear in *Theoretical Computer Science.*

[Ra90b] A. A. RAZBOROV, Applications of matrix methods for the theory of lower bounds in computational complexity, *Combinatorica 10*, 81–93, 1990.

[Sa72] J. E. SAVAGE, Computational work and time on finite machines, *Journal of the ACM* 19:4, 660–674, 1972.

[Sa80] V. Y. SAZANOV, A logical approach to the problem "P=NP?" *Mathematical Foundations of Computer Science, Lecture notes in Computer Science #88,* 562–575, 1980.

[Sc52] H. SCHOLZ, Problem #1, *Journal of Symbolic Logic*, 17, 160, 1952.

[Sh90] A. SHAMIR, IP=PSPACE, *Proceedings of 22th Annual ACM Symposium on Theory of Computing*, 11–15, 1990; to appear in the *Journal of the ACM.*

[Sh49] C. E. SHANNON, The synthesis of two-terminal switching circuits, *Bell Systems Technical Journal* 28:1, 59–98, 1949.

[Si81] M. SIPSER, On polynomial vs. exponential growth, unpublished, 1981.

[Si83] M. SIPSER, Borel sets and circuit complexity, *Proceedings of 15th Annual ACM Symposium on Theory of Computing*, 61–69, 1983.

[Si84] M. SIPSER, A topological view of some problems in complexity theory, *Colloquia Mathematica Societatis János Bolyai* 44, 387–391, 1984.

[Sm87] R. SMOLENSKY, Algebraic methods in the theory of lower bounds for Boolean circuit complexity, *Proceedings of 19th Annual ACM Symposium on Theory of Computing*, 77–82, 1987.

[Sz88] R. SZELEPCSÉNYI, The method of forced enumeration for nondeterministic automata, *Acta Informatica 26*, 279–284, 1988.

[Ta88] É. TARDOS, The gap between monotone and non-monotone circuit complexity is exponential, *Combinatorica* 8:1, 141–142, 1988.

[Tr84] B. A. TRAKHTENBROT, A survey of Russian approaches to Perebor (brute-force search) algorithms, *Annals of the History of Computing 6*, no. 4, 384–400, 1984.

[Ts62] G. S. TSEITIN, On the complexity of derivation in propositional calculus, *Studies in Constructive Mathematics and Mathematical Logic, Part 2*, Consultants Bureau, New Youk-London, 115–125, 1962.

[Va90] L. G. VALIANT, Why is Boolean complexity theory difficult? *Proceedings of LMS workshop on Boolean Function Complexity*, Durham, edited by M. S. Paterson, Cambridge University Press, 1990.

[Vo53] J. VON NEUMANN, A certain zero-sum two-person game equivalent to the optimal assignment problem, *Contributions to the Theory of Games 2*, 5–12, 1953.

[We87] I. WEGENER, *The Complexity of Boolean Functions*, Wiley-Teubner, 1987.

[Ya59a] S. YABLONSKI, The algorithmic difficulties of synthesizing minimal switching circuits, *Problemy Kibernetiki 2,* Moscow, Fizmatgiz, 75–121, 1959; translation in *Problems of Cybernetics 2,* Pergamon Press, 401–457, 1961.

[Ya59b] S. YABLONSKI, On the impossibility of eliminating brute force search in solving some problems of circuit theory, *Doklady, AN SSSR 124*, 44–47, 1959.

[Ya85] A. C. YAO, Separating the polynomial-time hierarchy by oracles, *Proceedings of 26th Annual IEEE Symposium on Foundations of Computer Science*, 1–10, 1985.