The Computational Complexity Column

by

Lance FORTNOW

NEC Laboratories America
4 Independence Way, Princeton, NJ 08540, USA
fortnow@nec-labs.com
http://www.neci.nj.nec.com/homepages/fortnow/beatcs

Every third year the Conference on Computational Complexity is held in Europe and this summer the University of Aarhus (Denmark) will host the meeting July 7-10. More details at the conference web page

http://www.computationalcomplexity.org

This month we present a historical view of computational complexity written by Steve Homer and myself. This is a preliminary version of a chapter to be included in an upcoming North-Holland Handbook of the History of Mathematical Logic edited by Dirk van Dalen, John Dawson and Aki Kanamori.

A Short History of Computational Complexity

Lance Fortnow[1]	Steve Homer[2]
NEC Research Institute	Computer Science Department
4 Independence Way	Boston University
Princeton, NJ 08540	111 Cummington Street
	Boston, MA 02215

1 Introduction

It all started with a machine. In 1936, Turing developed his theoretical computational model. He based his model on how he perceived mathematicians think. As digital computers were developed in the 40's and 50's, the Turing machine proved itself as the right theoretical model for computation.

Quickly though we discovered that the basic Turing machine model fails to account for the amount of time or memory needed by a computer, a critical issue today but even more so in those early days of computing. The key idea to measure time and space as a function of the length of the input came in the early 1960's by Hartmanis and Stearns. And thus computational complexity was born.

In the early days of complexity, researchers just tried understanding these new measures and how they related to each other. We saw the first notion of efficient computation by using time polynomial in the input size. This led to complexity's most important concept, **NP**-completeness, and its most fundamental question, whether $\mathbf{P} = \mathbf{NP}$.

The work of Cook and Karp in the early 70's showed a large number of combinatorial and logical problems were **NP**-complete, i.e., as hard as any problem computable in nondeterministic polynomial time. The $\mathbf{P} = \mathbf{NP}$ question is equivalent to an efficient solution of any of these problems. In the thirty years hence this problem has become one of the outstanding open questions in computer science and indeed all of mathematics.

In the 70's we saw the growth of complexity classes as researchers tried to encompass different models of computations. One of those models, probabilistic computation, started with a probabilistic test for primality, led to probabilistic complexity classes and a new kind of interactive proof system that itself led to hardness results for approximating certain **NP**-complete problems. We have also seen strong evidence that we can remove the randomness from computations and most recently a deterministic algorithm for the original primality problem.

In the 80's we saw the rise of finite models like circuits that capture computation in an inherently different way. A new approach to problems like $\mathbf{P} = \mathbf{NP}$ arose from these circuits and though they have had limited success in separating complexity classes, this approach brought combinatorial techniques into the area and led to a much better understanding of the limits of these devices.

In the 90's we have seen the study of new models of computation like quantum computers and propositional proof systems. Tools from the past have greatly helped our understanding of these new areas.

One cannot in the short space of this article mention all of the amazing research in computational complexity theory. We survey various areas in complexity choosing papers more for their historical value than necessarily the importance of the results. We hope that this gives an insight into the richness and depth of this still quite young field.

2 Early History

While we can trace the idea of "efficient algorithms" to the ancient Greeks, our story starts with the seminal 1965 paper of Hartmanis and Stearns, "On the Computational Complexity of Algorithms" [HS65]. This paper laid out the definitions of quantified time and space complexity on multitape Turing machines and showed the first results of the form given more time (or space) one can compute more things.

A multitape Turing machine consists of some fixed number of "tapes" each of which contains an infinite number of tape cells. The contents of a tape cell comes from a finite set of symbols called the tape alphabet of the Turing machine. All the tapes initially contain only a special "blank" character except for the finite input written at the beginning of the first tape. Each tape has a tape head sitting on the first character on each tape. The Turing machine also has a finite state memory to control its operations. In each step, it can move each tape independently one character left or right, read and possibly change the characters under each head, change its current state and decide whether to halt and accept or reject the input. Time is measured by the number of steps before halting as a function of the length of the input. Space is measured as the number of different character locations touched by the various heads.

The Hartmanis-Stearns paper did not develop in a vacuum. Turing [Tur36], of course, developed his notion of a computational device back in 1936. This machine model did and still does form the basis for most of computational complexity. Slightly earlier, Yamada [Yam62] studied "real-time computable functions", Myhill [Myh60] looked at linear bounded automata and Smullyan [Smu61] considered rudimentary sets. These models looked at specific time and space-bounded machines but did not give a general approach to measuring complexity.

After Hartmanis and Stearns developed the general method for measuring computational resources, one can ask how the different variations of Turing machines affect the complexity of problems. Rabin [Rab63] shows problems solvable faster by two-tape machine than by one-tape machines. Hennie and Stearns [HS66] show that a 2-tape Turing machine can simulate any constant tape machine taking only a logarithmic factor more time.

Hartmanis and Stearns show that given space functions s_1 and s_2 with a "constructibility" condition and $s_1(n) = o(s_2(n))$, i.e., $s_1(n)/s_2(n)$ goes to zero, then there are problems computable in space $s_2(n)$ but not space $s_1(n)$. The Hennie-Stearns result gives the best known time-hierarchy, getting a separation if $t_1(n) \log t_1(n) = o(t_2(n))$. These proofs use straightforward diagonalization arguments that go back to Cantor [Can74].

Nondeterministic computation allows a Turing machine to make a choice of several possible transitions. We say the machine accepts if any collection of choices leads to an accepting state. Nondeterministic time and space hierarchies are much trickier to prove because one cannot do straightforward diagonalization on nondeterministic computations.

Savitch [Sav70] showed that problems computable in nondeterministic space $s(n)$ are computable in deterministic space $s^2(n)$. In 1972, Ibarra [Iba72] using translational techniques of Ruby and Fischer [RF65] used Savitch's theorem to show that there exist problems computable in nondeterministic space n^a but not space n^b for $a > b \geq 1$. Sixteen years later, Immerman [Imm88] and Szelepcsényi [Sze88] independently showed that nondeterministic space is closed under complement. The Immerman-Szelepcsényi result immediately gives a nondeterministic space hierarchy as tight as the the deterministic hierarchy.

For nondeterministic time, Cook [Coo73] uses a more careful translation argument to show problems computable in nondeterministic time n^a but not time n^b for $a > b \geq 1$. Seiferas, Fischer and Meyer [SFM78] give the current best known nondeterministic time hierarchy, getting a separation if $t_1(n+1) = o(t_2(n))$.

In 1967, Blum [Blu67] had his speed-up theorem: For any computable unbounded function $r(n)$ there exists a computable language L such that for any Turing machine accepting L in time $t(n)$ there is another Turing machine accepting L in time $r(t(n))$. This seems to violate the time hierarchy mentioned earlier but one must realize $t(n)$ will not necessarily be time-constructible.

Blum's speed-up theorem holds not only for time but also for space and any other measure fulfilling a small list of axioms, which we now call Blum complexity measures.

Soon after we saw two other major results that we will state for time but also hold for all Blum complexity measures. Independently Borodin [Bor72] and Trakhtenbrot [Tra64] proved the gap theorem: For any computable unbounded $r(n)$ there exist a computable time bound $t(n)$ such that any language computable in time $t(n)$ is also computable in time $r(t(n))$. McCreight and Meyer [MM69] showed the union theorem: Given any computably presentable list of computable time bounds $t_1, t_2, \dots$ such that $t_{i+1} > t_i$ for all i then there exist a time bound t such that a problem is computable in time t if and only if it is computable in time t_i for some i.

In 1964, Cobham [Cob64] noted that the set of problems computable in polynomial time remains independent of the particular deterministic machine model. He also showed that many common mathematical functions can be computed in polynomial time.

In 1965, Edmonds [Edm65b] in his paper showing that the matching problem has a polynomial-time algorithm, argues that polynomial-time gives a good formalization of efficient computation. He noted the wide range of problems computable in polynomial time and as well the fact that this class of problems remains the same under many different reasonable models of computation. In another paper, Edmonds [Edm65a] gave an informal description of nondeterministic polynomial-time. This set the stage for the $\mathbf{P} = \mathbf{NP}$ question, the most famous problem in theoretical computer science that we discuss in Section 3.

Several Russians, notably Barzdin and Trakhtenbrot, independently developed several of these notions of complexity during the sixties though their work was not known to the West until the seventies.

3 NP-completeness

It was in the early 1970's that complexity theory first flowered, and came to play a central role in computer science. It did so by focusing on one fundamental concept and on the results and ideas stemming from it. This concept was **NP**-completeness and it has proved to be one of the most insightful and fundamental theories in the mathematics of the last half century. **NP**-completeness captures the combinatorial difficulty of a number of central problems which resisted efficient solution and provides a method for proving that a combinatorial problem is as intractable as any **NP** problem.

By the late 1960's, a sizable class of very applicable and significant problems which resisted polynomial time solution was widely recognized. These problems are largely optimization problems such as the traveling salesman problem, certain scheduling problems, or linear programming problems. They all have a very large number of possible solution where there is no obvious way to find an optimal solution other than a brute force search. As time passed and much effort was expended on attempts at efficiently solving these problems, it began to be suspected that there was no such solution. However, there was no hard evidence that this was the case nor was there any reason to suspect that these problems were in any sense difficult for the same reasons or in the same ways. The theory of **NP**-completeness provided precisely this evidence.

Proving a problem in **NP** to be **NP**-complete tells us that it is as hard to solve as any other **NP** problem. Said another way, if there is any **NP**-

complete problem that admits an efficient solution then every **NP** problem does so. The question of whether every **NP** problem has an efficient solution has resisted the efforts of computer scientists since 1970. It is known as the **P** versus **NP** problem and is among the most central open problems of mathematics. The fact that a very large number of fundamental problems have been shown to be **NP**-complete and that the problem of proving that **P** is not **NP** has proved to be so difficult has made this problem and the connected theory one of the most celebrated in contemporary mathematics. The $\mathbf{P} = \mathbf{NP}$ problem is one of the seven Millennium Prize Problems and solving it brings a \$1,000,000 prize from the Clay Mathematics Institute [Cla00].

Quite surprisingly, one of the earliest discussions of a particular **NP**-complete problem and the implications of finding an efficient solution came from Kurt Gödel. In a 1956 letter to von Neumann [Har86, Sip83] Gödel asks von Neumann about the complexity of what is now known to be an **NP**-complete problem concerning proofs in first-order logic and asks if the problem can be solved in linear or quadratic time. In fact, Gödel seemed quite optimistic about finding an efficient solution. He fully realized that doing so would have significant consequences.

It is worth noting that in about the same period there was considerable effort by Russian mathematicians working on similar combinatorial problems trying to prove that brute force was needed to solve them. Several of these problems eventually turned out to be **NP**-complete as well [Tra64].

The existence of **NP**-complete problems was proved independently by Stephen Cook in the United States and Leonid Levin in the Soviet Union. Cook, then a graduate student at Harvard, proved that the satisfiability problem is **NP**-complete [Coo71]. Levin, a student of Kolmogorov at Moscow State University, proved that a variant of the tiling problem is **NP**-complete [Lev73].

Researchers strove to show other interesting, natural problems **NP**-complete. Richard Karp, in a tremendously influential paper [Kar72], proved that eight central combinatorial problems are all **NP**-complete. These problems included the the clique problem, the independent set problem, the set cover problem, and the traveling salesman problem, among others.

Karp's paper presented several key methods to prove **NP**-completeness using reductions from problems previously shown to be **NP**-complete. It set up a general framework for proving **NP**-completeness results and established several useful techniques for such proofs. In the following years, and continuing until today, literally thousands of problems have been shown to be **NP**-complete. A proof of **NP**-completeness has come to signify the (worst case) intractability of a problem. Once proved **NP**-complete, researchers turn to other ways of trying to solve the problem, usually using approximation algorithms to give an approximate solution or probabilistic methods to solve the problem in "most" cases.

Another fundamental step was taken around 1970 by Meyer and Stockmeyer [MS72], [Sto76]. They defined the polynomial hierarchy in analogy with the arithmetic hierarchy of Kleene. This hierarchy is defined by iterating the notion of polynomial jump, in analogy with the Turing jump operator. This hierarchy has proven useful in classifying many hard combinatorial problems which do not lie in **NP**. It is explored in more detail in Section 4.2.

Of course, all problems in the polynomial hierarchy are recursive and in fact very simple problems within the vast expanse of all recursive sets. So are there natural problems which are recursive and are not captured by the hierarchy? The answers is yes and results in the exploration of several important larger complexity classes which contain the polynomial hierarchy. One such class is **PSPACE**, those problems which can be solved using work space which is of polynomial length relative to the length of the problem's input. Just as with **P** and **NP**, the full extent of **PSPACE** is not known. **PSPACE** contains **P** and **NP**. It is not known if either of these conclusions are proper. Settling these questions would again be significant steps forward in this theory.

The notion of **PSPACE**-completeness is defined very similarly to **NP**-completeness, and has been studies alongside the the **NP**-completeness notion. Namely, a problem C is **PSPACE**-complete if it is in **PSPACE** and if any other **PSPACE** problem can be reduced to it in polynomial time. As is the case with **NP**-complete problems, **PSPACE**-complete problems are quite common and often arise quite naturally. Typical **PSPACE**-complete problems are or arise from generalized games such as hex or checkers played on boards of unbounded finite size (see [GJ79]). Beyond **PSPACE** lie the exponential time (**EXPTIME**) and exponential space complexity classes. A small number of natural problems have been shown complete for these classes (see [GJ79]), and as well **EXPTIME** is the smallest deterministic class which has been proved to contain **NP**.

4 Structural Complexity

By the early 1970's, the definitions of time and space-bounded complexity classes were precisely established and the import of the class **NP** and of **NP**-

complete problems realized. At this point effort turned to understanding the relationships between complexity classes and the properties of problems within the principal classes. In particular, attention was focused on **NP**-complete problems and their properties and on the structure of complexity classes between **LOGSPACE** and **PSPACE**. We briefly survey some of these studies here.

4.1 The Isomorphism Conjecture

In the mid-70's, building on earlier work on Gödel numberings [HB75, Har82] and in analogy with the well-known result of Myhill from computability theory [Myh55], Berman and Hartmanis [BH77, HB78] formulated their isomorphism conjecture. The conjecture stated that all **NP**-complete sets are **P**-isomorphic (that is, isomorphic via polynomial time computable and invertible isomorphisms). This conjecture served as a springboard for the further study of the structure of **NP**-complete sets. As evidence for their conjecture, Berman and Hartmanis and others [MY85, KMR87] were able to give simple, easily checkable properties of **NP**-complete sets which implied they were isomorphic. Using these, they proved that all of the **known NP**-complete sets were in fact **P**-isomorphic. This conjecture remains an open question today. A positive resolution of the conjecture would imply that **P** is not equal to **NP**. Much effort was focused on proving the converse, that assuming **P** is not **NP** then the isomorphism conjecture holds. This remains an open question today.

As the number of known **NP**-complete problems grew during the 1970's, the structure and properties of these problems began to be examined. While very disparate, the **NP**-complete sets have certain common properties. For example, they are all rather dense sets. Density of a set is measured here simply in the sense of how many string of a given length are in the set. So (assuming a binary encoding of a set) there are 2^n different strings of length n. We say that set S is sparse if there is a polynomial p(n) which bounds the number of strings in S of length n, for every n. It is dense otherwise. All known **NP**-complete sets are dense.

One consequence of the isomorphism conjecture is that no **NP**-complete set can be sparse. As with the isomorphism conjecture, this consequence implies that **P** is not **NP** and so it is unlikely that a proof of this consequence will soon be forthcoming. Berman and Hartmanis also conjectured that if **P** in not equal to **NP** there are no sparse **NP**-complete sets. This conjecture was settled affirmatively by the famous result of Mahaney [Mah82]. Mahaney's elegant proof used several new counting techniques and had a lasting impact on work in structural complexity theory.

4.2 The Polynomial Hierarchy

While numerous hard decision problems have been proved **NP**-complete, a small number are outside **NP** and have escaped this classification. An extended classification, the polynomial time hierarchy (**PH**), was provided by Meyer and Stockmeyer [Sto76]. They defined the hierarchy, a collection of classes between **P** and **PSPACE**, in analogy with Kleene's arithmetic hierarchy.

The polynomial time hierarchy (**PH**) consists of an infinite sequence of classes within **PSPACE**. The bottom (0^{th}) level of the hierarchy is just the class **P**. The first level is the class **NP**. The second level are all problems in **NP** relative to an **NP** oracle, etc. Iterating this idea to all finite levels yields the full hierarchy.

If **P**=**PSPACE** then the whole **PH** collapses to the class **P**. However, quite the opposite is believed to be the case, namely that the **PH** is strict in the sense that each level of the hierarchy is a proper subset of the next level. While every class in the **PH** is contained in **PSPACE**, the converse is not true if the hierarchy is strict. In this case, **PSPACE** contains many problems not in the **PH** and in fact has a very complex structure (see, for example, [AS89]).

4.3 Alternation

Another unifying and important thread of results which also originated during the 1970's was the work on **alternation** initiated out by Kozen, Chandra and Stockmeyer [CKS81]. The idea behind alternation is to classify combinatorial problems using an alternating Turing machine, a generalization of a nondeterministic Turing machine. Intuitively a nondeterministic Turing machine can be thought of as having an existential acceptance criterion. That is, an input to the TM is accepted if there *exists* a computation path of the machine which results in acceptance. Similarly, we could consider a universal acceptance criterion whereby a nondeterministic machine accepts of all computation paths lead to acceptance. Restricting ourselves to polynomial length alternation, we see that **NP** can be characterized as those problems accepted by nondeterministic TM running in polynomial time using the existential acceptance criterion. Similarly, the universal acceptance criterion with the same type of machines defines the class co-**NP** consisting of problems whose complements are in **NP**. Furthermore, we can iterate these two acceptance methods, for example asking that there exist an path of a TM such that for all paths extending that path there exists an extension of that path which accepts. This idea gives a machine implementation of the notion of alternations of universal and existential quantifiers. It is not hard to see that finitely many alternations results in the finite levels of the polynomial time hierarchy and that alternating polynomial time is the same thing as **PSPACE**. Other relationship between time and space classes defined using alternation can be found in [CKS81], for example, alternating log space = **P** and alternating **PSPACE** = **EXPTIME**.

4.4 Logspace

To this point all the complexity classes we have considered contain the class **P** of polynomial time computable problems. For some interesting problems it is useful to consider classes within **P** and particularly the seemingly smaller space classes of deterministic log space, denoted **L**, and nondeterministic log space, denoted **NL**. These classes provide a measure with which to distinguish between some interesting problems within **P**, and present interesting issues in their own right.

At first glance logarithmic space is a problematic notion at best. An input of length n takes n squares by itself, so how can a computation on such an input take only log n space? The answer lies in changing our computation model slightly to only count the space taken by the computation and not the space of the input. Formally, this is done by considering an "off-line Turing machine." This is a (deterministic or nondeterministic) Turing machine whose input is written on a special read-only input tape. Computation is carried out on read-write work tapes which are initially blank. The space complexity of the computation is then taken to be the amount of space used on the work tapes. So in particular this space can be less than n, the length of the input to the computation. We define logspace, **L**, to be the class of languages decided by deterministic Turing machines which use at most $O(\log n)$ tape squares. Similarly, **NL** is defined using nondeterministic Turing machines with the same space bound.

It is straightforward to check that $\mathbf{L} \subseteq \mathbf{NL} \subseteq \mathbf{P}$, and these three classes are thought to be distinct. There are a number of nontrivial problems solvable in **L** (for example see [LZ77]) as well as problems known to be in **NL** which are not believed to be in **L** (for example see [Sav73, Jon75]). Numerous problems in **P** are thought to lie outside of **L** or **NL**. For example, one such problem is the circuit value problem, the problem of determining the value of a Boolean circuit, given inputs to the circuit. The circuit value problem is one of many problems in **P** which is known to be **P** complete. These are problems in **P** which are proved to be complete with respect to log-space bounded reductions, reductions defined analogously to polynomial time bounded reduction in the previous section. Proving a **P**-complete problem is in **L** would imply that **L** = **P**.

4.5 Oracles

Oracle results play a unique role in complexity theory. They are metamathematical results delineating the limitations of proof techniques and indicating what results might be possible to achieve and which are likely beyond our current reach. Oracle results concern relativized computations. We say that a computation is carried out "relative to an oracle set O" if the computation has access to the answers to membership queries of O. That is, the computation can query the oracle O about whether or not a string x is in O. The computation obtains the answer (in one step) and proceeds with the computation, which may depend on the answer to the oracle query.

The first, and still most fundamental oracle results in complexity were carried out by Baker, Gill and Solovay [BGS75]. They proved that there is an oracle reactive to which **P**=**NP** and another oracle relative to which P and **NP** differ.

What do these results say about the **P** vs **NP** question? They say little about the actual answer to this question. The existence of an oracle making a statement S true is simply a kind of consistency result about S. It says that the statement is true in one particular model or "world" (that is, the oracle set itself). As such, we can conclude that a proof of the negation of S will not itself relativize to any oracle. Thus, as many proof methods do relativize to every oracle, an oracle result provides a limitation to the possible methods used to prove S and hence are evidence that the result is, in this sense, hard. Oracle results have been most useful in delineating theorems which are difficult to prove (i.e., those which do no relativize), from those which might more likely be settled by well-understood, relativizing proof techniques. In particular, the Baker, Gill and Solovay results concerning **P** and **NP** question indicate that a proof will be difficult to come by, as has indeed been the case.

Since 1978 numerous other oracle results have been proved. Techniques used to achieve these results have become quite sophisticated and strong. For instance, Fenner, Fortnow and Kurtz [FFK94] gave a relativized world where the isomorphism conjecture holds where Kurtz, Mahaney and Royer [KMR89] had showed that it fails relative to most oracles. They were the culmination of a long series of partial results addressing this question.

There are a few results in complexity that do not relativize, mostly relating to interactive proof systems (see Section 6.1) but these tend to be the exception and not the rule.

5 Counting Classes

Another way to study **NP** computations is to ask how many computations paths in the computation lead to acceptance. For example, consider the satisfiability problem. Given an instance of this problem, that is a propositional formula, instead of asking if the formula has a solution (a truth assignment making the formula true), ask how many such assignments there are. In 1979, Valiant [Val79] defined the complexity class #**P** as the class of functions computing the number of accepting paths of a nondeterministic Turing machine. Every #**P** function is computable in polynomial space. Valiant used this class to capture the complexity of the counting version of satisfiability as well as other interesting problems such as computing the permanent function.

Counting complexity has since played an important role in computational complexity theory and theoretical computer science. The techniques used in counting complexity have significant applications in circuit complexity and in the series of recent results on interactive proof systems. (See the next section.) The two most important counting function classes are #**P**, described above, and **GapP**. **GapP** consists of the class of functions which compute the difference between the number of accepting paths and the number of rejecting paths of a nondeterministic Turing machine. For example, the function which tells, for any propositional formula, computes the difference of the number of accepting and rejecting truth assignments is in the class **GapP**.

Perhaps the two most important recent results in counting complexity are Toda's theorem [Tod91] and the closure theorem of Beigel, Reingold and Spielman's [BRS95]. Toda's theorem asserts that one can reduce any language in the polynomial-time hierarchy to a polynomial time computation which uses a #**P** function as an oracle. Hence, that in terms of complexity, hard functions in #**P** lie above any problem in the polynomial time hierarchy. In 1994, Beigel, Reingold and Spielman [BRS95] proved that **PP** is closed under union. This result solved a longstanding open problem in this area, first posed by Gill in 1977 [Gil77] in the initial paper on probabilistic classes. It implies that **PP** is also closed under intersection Those interested in further exploring counting classes and the power of counting in complexity theory should consult the papers of Schöning [Sch90] and Fortnow [For97].

6 Probabilistic Complexity

In 1977, Solovay and Strassen [SS77] gave a new kind of algorithm for testing whether a number is prime. Their algorithm flipped coins to help search for a counterexample to primality. They argued that if the number was not prime then with very high confidence a counterexample could be found.

This algorithm suggested that we should revise our notion of "efficient computation". Perhaps we should now equate the efficiently computable problems with the class of problems solve in probabilistic polynomial time. A whole new area of complexity theory was developed to help understand the power of probabilistic computation.

Gill [Gil77] defined the class **BPP** to capture this new notion. Adleman and Manders [AM77] defined the class **R** that represented the set of problems with one-sided randomness–the machine only accepts if the instance is guaranteed to be in the language. The Solovay-Strassen algorithm puts compositeness in **R**.

Babai introduced the concept of a "Las Vegas" probabilistic algorithm that always gives the correct answer and runs in expected polynomial time. This class **ZPP** is equivalent to those problems with both positive and negative instances in **R**. Adleman and Huang [AH87] building on work of Goldwasser and Kilian [GK99] show that primality is in **R** and thus **ZPP**.

Very recently, Agrawal, Kayal and Saxena [AKS02] gave a deterministic polynomial-time algorithm for primality. If this result was known in the 70's, perhaps the study of probabilistic algorithms would not have progressed as quickly.

In 1983, Sipser [Sip83] showed that **BPP** is contained in the polynomial-time hierarchy. Gács (see [Sip83]) improves this result to show **BPP** is in the second level of the hierarchy and Lautemann [Lau83] gives a simple proof of this fact.

One can also consider probabilistic space classes. Aleliunas, Karp, Lipton, Lovász and Rackoff [AKL+79] show that undirected graph connectivity can be computed in one-sided randomized logarithmic space, a class called **RL**. Similarly one can define the classes **BPL** and **ZPL**. Borodin, Cook, Dymond, Ruzzo and Tompa [BCD+89] showed that undirected graph nonconnectivity also sits in **RL** and thus **ZPL**. Nisan and Ta-Shma [NT95] showed that the connectivity question reduced directly to the nonconnectivity question.

6.1 Interactive Proof Systems

One can think of the class **NP** as a *proof system*: An arbitrarily powerful prover gives a proof that say a formula is satisfiable. One can generalize this notion of proof system by allowing probabilistic verification of the proof. This yields the complexity class **MA**. One can also consider interaction where the verifier sends messages based on her random coins. The bounded round version of this class is **AM** and the unbounded round version is **IP**. The incredible power of these interactive proof systems has led to several of the most surprising and important recent results in computational complexity theory.

Babai [Bab85] defined interactive proof systems to help classify some group questions. An alternative interactive proof system was defined by Goldwasser, Micali and Rackoff [GMR89] as a basis for the cryptographic class zero-knowledge. Zero-knowledge proof systems have themselves played a major role in cryptography.

The two models differed on whether the prover could see the verifier's random coins, but Goldwasser and Sipser [GS89] showed the two models equivalent. Babai and Moran [BM88] showed that any bounded-round protocol needs only one question from the verifier followed by a response from the prover. Fürer, Goldreich, Mansour, Sipser and Zachos [FGM$^+$89] showed that one can assume that for positive instances the prover can succeed with no error.

Goldreich, Micali and Wigderson [GMW91] show that the set of pairs of nonisomorphic graphs has a bounded-round interactive proof system. Boppana, Håstad and Zachos [BHZ87] show that if the complement of any **NP**-complete language has bounded-round interactive proofs than the polynomial-time hierarchy collapses. This remains the best evidence that the graph isomorphism problem is probably not **NP**-complete.

In 1990, Lund, Fortnow, Karloff and Nisan [LFKN92] showed that the complements of **NP**-complete languages have unbounded round interactive proof systems. Shamir [Sha92] quickly extended their techniques to show that every language in **PSPACE** has interactive proof system. Feldman [Fel86] had earlier shown that every language with interactive proofs lies in **PSPACE**.

Interactive proofs are notable in that in general proofs concerning them do not relativize, that is they are not true relative to every oracle. The classification of interactive proofs turned out not to be the end of the story but only the beginning of a revolution connecting complexity theory with approximation algorithms. For the continuation of this story we turn to probabilistically checkable proofs.

6.2 Probabilistically Checkable Proofs

In 1988, Ben-Or, Goldwasser, Kilian and Wigderson [BGKW88] developed the multiprover interactive proof system. This model has multiple provers who cannot communicate with each other or see the conversations each has with the verifier. This model allows the verifier to play one prover off another.

Fortnow, Rompel and Sipser [FRS94] show this model is equivalent to probabilistically checkable proofs, where the prover writes down a possibly exponentially long proof that the verifier spot checks in probabilistic polynomial time. They also show that every language accepted by these proof systems lie in **NEXP**, nondeterministic exponential time.

In 1990, Babai, Fortnow and Lund [BFL91] show the surprising converse–that every language in **NEXP** has probabilistically checkable proofs. Babai, Fortnow, Levin and Szegedy [BFLS91] scale this proof down to develop "holographic" proofs for **NP** where, with a properly encoded input, the verifier can check the correctness of the proof in very short amount of time.

Feige, Goldwasser, Lovász, Safra and Szegedy [FGL$^+$96] made an amazing connection between probabilistically checkable proofs and the clique problem. By viewing possible proofs as nodes of a graph, they showed that one cannot approximate the size of a clique well without unexpected collapses in complexity classes.

In 1992, Arora, Lund, Motwani, Sudan and Szegedy [ALM$^+$98] building on work of Arora and Safra [AS98] showed that every language in **NP** has a probabilistically checkable proof where the verifier uses only a logarithmic number of random coins and a constant number of queries to the proof.

The Arora et. al. result has tremendous implications for the class **MAXSNP** of approximation problems. This class developed by Papadimitriou and Yannakakis [PY91] has many interesting complete problems such as max-cut, vertex cover, independent set, traveling salesman on an arbitrary metric space and maximizing the number of satisfiable clauses of a formula.

Arora et. al. show that, unless $\mathbf{P} = \mathbf{NP}$, every **MAXSNP**-complete set does not have a polynomial-time approximation scheme. For each of these problems there is some constant $\delta > 1$ such that they cannot be approximated within a factor of δ unless $\mathbf{P} = \mathbf{NP}$.

Since these initial works on probabilistically checkable proofs, we have seen a large number of outstanding papers improving the proof systems and getting stronger hardness of approximation results. Håstad [Hås97] gets tight results for some approximation problems. Arora [Aro98] after failing to achieve lower bounds for traveling salesman in the plane, has developed a polynomial-time approximation algorithm for this and related problems.

A series of results due to Cai, Condon, Lipton, Lapidot, Shamir, Feige and Lovász [CCL92, CCL90, CCL91, Fei91, LS91, FL92] have modified the protocol of Babai, Fortnow and Lund [BFL91] to show that every language in **NEXP** has a two-prover, one-round proof systems with an exponentially small error. This problem remained so elusive because running these proof systems in parallel does not have the expected error reduction [FRS94]. In 1995, Raz [Raz98] showed that the error does go done exponentially when these proofs systems are run in parallel.

6.3 Derandomization

If you generate a random number on a computer, you do not get a truly random value, but a pseudorandom number computed by some complicated function on some small, hopefully random seed. In practice this usually works well so perhaps in theory the same might be true. Many of the exciting results in complexity theory in the 1980's and 90's consider this question of derandomization–how to reduce or eliminate the number of truly random bits to simulate probabilistic algorithms.

The first approach to this problem came from cryptography. Blum and Micali [BM84] first to show how to create randomness from cryptographically hard functions. Yao [Yao90] showed how to reduce the number of random bits of any algorithm based on any cryptographically secure one-way permutation. Håstad, Impagliazzo, Levin and Luby [HILL99] building on techniques of Goldreich and Levin [GL89] and Goldreich, Krawczyk and Luby [GKL93] show that one can get pseudorandomness from any one-way function.

Nisan and Wigderson [NW94] take a different approach. They show how to get pseudorandomness based on a language hard against nonuniform computation. Impagliazzo and Wigderson [IW97] building on this result and Babai, Fortnow, Nisan and Wigderson [BFNW93] show that **BPP** equals **P** if there exists a language in exponential time that cannot be computed by any subexponential circuit.

For derandomization of space we have several unconditional results. Nisan [Nis92] gives general tools for derandomizing space-bounded computation. Among the applications, he gets a $O(\log^2 n)$ space construction for universal traversal sequences for undirected graphs.

Saks and Zhou [SZ99] show that every probabilistic logarithmic space algorithm can be simulated in $O(\log^{3/2} n)$ deterministic space. Armoni, Ta-Shma, Wigderson and Zhou [ATWZ97] building on work of Nisan, Szemerédi, and Wigderson [NSW92] show that one can solve undirected graph connectivity in $O(\log^{4/3} n)$ space.

7 Descriptive Complexity

Many of the fundamental concepts and methods of complexity theory have their genesis in mathematical logic, and in computability theory in particular. This includes the ideas of reductions, complete problems, hierarchies and logical definability. It is a well-understood principle of mathematical logic that the more complex a problem's logical definition (for example, in terms of quantifier alternation) the more difficult its solvability. Descriptive complexity aims to measure the computational complexity of a problem in terms of the complexity of the logical language needed to define it. As is often the case in complexity theory, the issues here become more subtle and the measure of the logical complexity of a problem more intricate than in computability theory. Descriptive complexity has its beginnings in the research of Jones, Selman, Fagin [JS74, Fag73, Fag74] and others in the early 1970's. More recently descriptive complexity has had significant applications to database theory and to computer-aided verification.

The ground breaking theorem of this area is due to Fagin [Fag73]. It provided the first major impetus for the study of descriptive complexity. Fagin's Theorem gives a logical characterization of the class **NP**. It states that **NP** is exactly the class of problems definable by existential second order Boolean formulas. This result, and others that follow, show that natural complexity classes have an intrinsic logical complexity.

To get a feel for this important idea, consider the **NP**-complete problem of 3 colorability of a graph. Fagin's theorem says there is a second order existential formula which holds for exactly those graphs which are 3-colorable. This formula can be written as $(\exists A, B, C)(\forall v)[(A(v) \vee B(v) \vee C(v)) \wedge (\forall w)(E(v,w) \rightarrow \neg(A(v) \wedge A(w)) \wedge \neg(B(v) \wedge B(w)) \wedge \neg(C(v) \wedge C(w)))]$. Intuitively this formula states that every vertex is colored by one of three colors A, B, or C and no two adjacent vertices have the same color. A graph, considered as a finite model, satisfies this formula if and only if it is 3-colorable.

Fagin's theorem was the first in a long line of results which prove that complexity classes can be given logical characterizations, often very simply and elegantly. Notable among these is the theorem of Immerman and Vardi [Imm82, Var82] which captures the complexity of polynomial time. Their theorem states that the class of problems definable in first order logic with the addition of the least fixed point operator is exactly the complexity class P. Logspace can be characterized along these same lines, but using the transitive closure (TC) operator rather than least fixed point. That is, nondeterministic logspace is the class of problems definable in first order logic with the addition of TC (see Immerman [Imm88]). And if one replaces first order logic with TC with second order logic with TC the result is **PSPACE** (see Immerman [Imm83]). Other, analogous results in this field go on to characterize various circuit and parallel complexity classes, the polynomial time hierarchy, and other space classes, and even yield results concerning counting classes.

The intuition provided by looking at complexity theory in this way has proved insightful and powerful. In fact, one proof of the famous Immerman-Szelepcsenyi Theorem [Imm88, Sze88] (that by Immerman) came from these logical considerations. This theorem say that any nondeterministic space class which contains logspace is closed under complement. An immediate consequence is that the context sensitive languages are closed under complement, answering a question which had been open for about 25 years.

To this point we have considered several of the most fully developed and fundamental areas of complexity theory. We now survey a few of the more central topics in the field dealing with other models of computation and their complexity theory. These include circuit complexity, communication complexity and proof complexity.

8 Finite Models

8.1 Circuit Complexity

The properties and construction of efficient Boolean circuits are of practical importance as they are the building block of computers. Circuit complexity studies bounds on the size and depth of circuits which compute a given Boolean functions. Aside from their practical value, such bounds are closely tied to important questions about Turing machine computations.

Boolean circuits are directed acyclic graphs whose internal nodes (or "gates") are Boolean functions, most often the "standard" Boolean functions, *and, or* and *not*. In a circuit, the nodes of in-degree 0 are called input nodes and labeled with input variables. The nodes with out-degree 0 are called output nodes. The value of the circuit is computed in the natural way by giving values to the input variables, applying the gates to these values, and computing the output values.

The **size**, $s(C)$, of a circuit C is the number of gates it contains. The **depth**, $d(C)$, of a circuit C is the length of the longest path from an input to an output node.

A circuit with n inputs can be thought of as a recognizer of a set of strings of length n, namely those which result in the circuit evaluating to 1. In order to consider circuits as recognizing an infinite set of strings, we consider circuit families which are infinite collections of circuits, C_n, one for each input length. In this way a circuit family can recognize a language just as a Turing machine can.

A circuit family is a nonuniform model, the function taking n to C_n may not be computable. A nonuniform circuit family can recognize noncomputable sets. We can measure the size and depth of circuit families using asymptotic notation. So, for example, we say that a circuit family has polynomial size if s(C_n) is O(p(n)), for some polynomial p(n). Any language in P has polynomial size circuits. That is, it is recognized by a circuit family which has polynomial size. And so proving that some **NP** problem does not have polynomial size circuits would imply that $\mathbf{P} \neq \mathbf{NP}$. Largely because of many such implications for complexity classes, considerable effort has been devoted to proving circuit lower bounds. However, to this point this effort has met with limited success.

In an early paper, Shannon [Sha49] showed that most Boolean functions require exponential size circuits. This proof was nonconstructive and proving bounds on particular functions is more difficult. In fact, no non-linear lower bound is known for the circuit size of a concrete function.

To get more positive results one needs to restrict the circuit families being considered. This can be done by requiring some uniformity in the function mapping n to C_n, or it can be done by restricting the size or depth of the circuits themselves. For example, the class $\mathbf{AC^0}$ consists of those languages recognized by uniform, constant depth, polynomial size circuits with and, or and not gates which allow unbounded fan-in. One early and fundamental results, due to Furst, Saxe and Sipser [FFS88] and Ajtai [Ajt83] is that the parity function is not in $\mathbf{AC^0}$, and in fact requires exponential size $\mathbf{AC^0}$-type circuits [Yao90]. This immediately implies that $\mathbf{AC^0}$ differs from the class **ACC** of languages which have circuit families made from $\mathbf{AC^0}$ circuits with the addition of Mod_m gates, with m fixed for the circuit family. It also can be shown to imply the existence of an oracle separating the polynomial hierarchy from **PSPACE**.

It is also known that the classes **ACC(p)** are all distinct, where only Mod_p gates are allowed, for p a prime. This was shown by Smolensky [Smo87] and Razborov [Raz98]. **ACC** itself has resisted all lower bound techniques and in fact it is not even know to be properly contained in **NP**.

Razborov [Raz85b] showed that clique does not have small monotone circuits, i.e., just AND and OR gates without negations. However, this result says more about the limitations of monotone circuits as Razborov [Raz85a] showed that the matching problem, known to be in **P**, also does not have small monotone circuit.

8.2 Communication Complexity

Much of modern computer science deals with the speed and efficiency at which digital communication can take place. Communication complexity is an attempt to model the efficiency and intrinsic complexity of communication between computers. It studies problems which model typical communication needs of computations and attempts to determine the bounds on the amount of communication between processors that these problems require.

The basic question of communication complexity is, how much information do two parties need to exchange in order to carry out a computation? We assume both parties have unlimited computational power.

For example, consider the case where both parties have n input bits and they want to determine if there is a position $i \leq n$ where the two bits in position i match. It is not hard to see that the communication complexity of this problem is n, as the n bits are independent and in the worst case, all n bits of one party have to be transmitted to the other.

Now consider the problem of computing the parity of a string of bits where 1/2 of the bits are given to party 1 and the other half to party 2. In this case, party 1 need only compute the parity of her bits and send this parity to party 2 who can then compute the parity of the whole bit string. So in this case the communication complexity is a single bit.

Communication complexity has provided upper and lower bounds for the complexity of many fundamental communication problems. It has clarified the role which communication plays in distributed and parallel computation as well as in the performance of VLSI circuits. It also applies and has had an impact on the study of interactive protocols. For a good survey of the major results in this field, consult Nisan and Kushelevitz [KN96].

8.3 Proof Complexity

The class **NP** can be characterized as those problems which have short, easily verified membership proofs. Dual to **NP**-complete problems, like SAT, are **co**$-$**NP**-complete problems, such as TAUT (the collection of propositional tautologies). TAUT is not known to have short, easily verified membership proofs, and in fact if it did then **NP** $=$ **co**$-$**NP** (see Cook and Reckhow [CR73]). Proof complexity studies the lengths of proofs in propositional logic and the connections between propositional proofs and computational complexity theory, circuit complexity and automated theorem proving. In the last decade there have been significant advances in lower bounds for propositional proof complexity as well as in the study of new and interesting proof systems.

Cook and Reckhow [CR73] were the first to make the notion of a propositional proof system precise. They realized that to do this they needed to specify exactly what a proof is and to give a general format for presenting and efficiently verifying a proof p. They defined a propositional proof system S to be a polynomial-time computable predicate, R, such that for all propositional formulas, F, $F \in TAUT \iff \exists p \; S(F,p)$. The complexity of S is then defined to be the smallest function $f : N \longrightarrow N$ which bounds the lengths of the proofs of S as a function of the lengths of the tautologies being proved. Efficient proof systems, those with complexity bounded by some polynomial, are called polynomial-bounded proof systems.

Several natural proof systems have been defined and their complexity and relationship explored. Among the most studied are Frege and extended-Frege Proof systems [Urq87] and [KP89], refutation systems, most notably resolution [Rob65] and circuit based proof systems [Ajt83] and [Bus87]. We briefly discuss the complexity of resolution systems here, but see Beame and Pitassi [BP98] for a nice overview of results concerning these other proof systems.

Resolution proof systems are the most well-studied model. Resolution is a very restricted proof system and so has provided the setting for the first lower bound proofs. Resolution proof systems are refutation systems where a statement D is proved by assuming its negation and deriving a contradiction from this negation. In a resolution proof system there is a single rule of inference, resolution, which is a form of cut. In its propositional form it say that is $F \vee x$ and $G \vee \neg x$ are true then $F \vee G$ follows. A restricted form of resolution, called regular resolution, was proved to have a superpolynomial lower bound by Tseitin [Tse68] on certain tautologies representing graph properties. The first superpolynomial lower bound for general resolution was achieved by Haken [Hås89] who in 1985 proved an exponential lower bound for the pigeonhole principle. Since then several other classes of tautologies have been shown to require superpolynomial long resolution proofs.

9 Quantum Computing

The mark of a good scientific field is its ability to adapt to new ideas and new technologies. Computational complexity reaches this ideal. As we have developed new ideas of probabilistic and parallel computation, the complexity community has not thrown out the previous research, rather they have modified the existing models to fit these new ideas and have shown how to connect the power of probabilistic and parallel computation to our already rich theory. Most recently complexity theorists have begun to analyze the computational power of machines based on quantum mechanics.

In 1982, Richard Feynman [Fey82], the physicist, noted that current computer technology could not efficiently simulate quantum systems. He suggested the possibility that computers built on quantum mechanics might be able to perform this task. David Deutch [Deu85] in 1985 developed a theoretical computation model based on quantum mechanics and suggested that such a model could efficiently compute problems not computable by a traditional computer.

Two quantum algorithms have received quite a bit of notice: Shor's [Sho97] procedure for factoring integers in polynomial time on a quantum computer and Grover's [Gro96] technique for searching a database of n elements in $O(\sqrt{n})$ time.

We know surprisingly little about the computational complexity of quantum computing. Bernstein and Vazirani [BV97] give a formal definition of the class **BQP** of language efficiently computable by quantum computers. They show the surprising robustness of **BQP** which remains unscathed under variations of the model such as restricting to a small set of rational amplitudes, allowing quantum subroutines and a single measurement at the end of the computation.

Bernstein and Vazirani show that **BQP** is contained in **PSPACE**. Adleman, DeMarrais and Huang [ADH97] show that **BQP** is contained in the counting class **PP**. Bennett, Bernstein, Brassard and Vazirani [BBBV97] give a relativized world where **NP** is not contained in **BQP**. We do not know any nonrelativized consequences of **NP** in **BQP** or if **BQP** lies in the polynomial-time hierarchy.

What about quantum variations of **NP** and interactive proof systems?

Fenner, Green, Homer and Pruim [FGHP99] consider the class consisting of the languages L such that for some polynomial-time quantum Turing machine, x is in L when $M(x)$ accepts with positive probability. They show the equivalence of this class to the counting class $\mathbf{co{-}C_{=}P}$.

Watrous [Wat99] shows that every language in **PSPACE** has a bounded-round quantum interactive proof system. Kitaev and Watrous [KW00] show that every quantum interactive proof system has an equivalent bounded-round proof system and every such language sits in deterministic exponential time.

We have seen quite a bit of progress on quantum decision tree complexity. In this model we count the number of queries made to a black-box database of size n. Quantum queries can be made in superposition.

Deutsch and Jousza [DJ92] gave an early example of a simple function that can be solved with one query quantumly but requires $\Omega(n)$ queries deterministically or probabilistically with no error. Bernstein and Vazirani [BV97] give the first example of a problem that can be solved with polynomial number of queries quantumly but requires a superpolynomial number of queries probabilistically with bounded error. Simon [Sim97] gives another example with an exponential gap. Brassard and Høyer [BH97] gave a zero-error quantum algorithms for Simon's problem. Shor's factoring algorithm [Sho97] can be viewed as an extension of Simon's problem that finds the period in a periodic black-box function.

All of these examples require a promise, i.e., restricting the allowable inputs to be tested. Fortnow and Rogers [FR99] and Beals, Buhrman, Cleve, Mosca and de Wolf [BBC$^+$98] show that a promise is necessary to get a superpolynomial separation.

10 Future Directions

Despite the plethora of exciting results in computational complexity over the past forty years, true complexity class separations have remained beyond our grasp. Tackling these problems, especially showing a separation of **P** and **NP**, is our greatest challenge for the future.

How will someone prove that **P** and **NP** differ? As of this writing, we have no serious techniques that could help separate these classes. What kind of future ideas could lead us to answer this difficult question? Some possibilities:

- A unexpected connection to other areas of mathematics such as algebraic geometry or higher cohomology. Perhaps even an area of mathematics not yet developed. Perhaps someone will develop a whole new direction for mathematics in order to handle the **P** versus **NP** question.

- New techniques to prover lower bounds for circuits, branching programs and/or proof systems in models strong enough to give complexity class separations.

- A new characterization of **P** or **NP** that makes separation more tractable.

- A clever twist on old-fashioned diagonalization, still the only techniques that has given any lower bounds on complexity classes.

Complexity theory will progress in areas beyond class separation. Still, quite a few interesting questions remain in many areas, even basic questions in quantum computational complexity remain. Complexity theorists will continue to forge new ground and find new and exciting results in these directions.

As with probabilistic, parallel and quantum complexity, new models of computation will be developed. Computational complexity theorists will be right on top of these developments leading the way to understand the inherent efficient computational power of these models.

We have seen many books and popular news stories about the other "complexity", complex systems that occur in many aspects of society and nature such as financial markets, the internet, biological systems, the weather and debatably even physical systems. This theory suggests that such systems have a very simple set of rules that when combined produce quite a complex behavior. Computer programs exhibit a very similar behavior. We will see computational complexity techniques used to help understand the efficiency of the complex behavior of these systems.

Finally, computational complexity will continue to have the Big Surprise. No one can predict the next big surprise but it will happen as it always does.

Let us end this survey with a quote from Juris Hartmanis' notebook (see [Har81]) in his entry dated December 31, 1962

This was a good year.

This was a good forty years and complexity theory is only getting started.

11 Further Reading

There have been several articles on various aspects of the history of complexity theory, many of which we have used as source material for this article. We give a small sampling of pointers here:

- [Har81] Juris Hartmanis reminisces on the beginnings of complexity theory.
- [Tra84] Boris Trakhtenbrot describes the development of **NP**-completeness from the Russian perspective.
- [Sip92] Michael Sipser gives a historical account of the **P** versus **NP** question including a copy and translation of Gödel's historic letter to von Neumann.
- [GJ79] Michael Garey and David Johnson give a "terminological history" of **NP**-completeness and a very readable account of the basic theory of **NP**-completeness.
- The collection of papers edited by Hochbaum [Hoc95] is a good overview of progress made in approximating solutions to **NP**-hard problems.
- Consult the book by Greenlaw, Hoover and Ruzzo [RGR95] to learn more of complexity theory within **P** and for many more **P**-complete problems.
- The Turing award lectures of Cook [Coo83], Karp [Kar86], Hartmanis [Har94] and Stearns [Ste94] give interesting insights into the early days of computational complexity.
- The textbook of Homer and Selman [HS00] contains a careful development of the definitions and basic concepts of complexity theory, and proofs of many central facts in this field.
- The complexity columns of SIGACT news and the Bulletin of the EATCS have had a number of excellent surveys on many of the areas described in this article.
- The two collections Complexity Theory Retrospective [Sel88] and Complexity Theory Retrospective II [HS97] contain some excellent recent surveys of several of the topics mentioned here.

Acknowledgments

The authors would like to thank their colleagues, far too numerous to mention, whom we have had many wonderful discussions about complexity over the past few decades. Many of these discussions have affected how we have produced various aspects of this article.

We would like to thank the editors of this book, particularly Aki Kanamori, for inviting us to write this article. The first author would like to thank the program committee of the 17th Annual Conference on Computational Complexity for inviting him to present some of this material at that conference.

References

[ADH97] L. Adleman, J. DeMarrais, and M. Huang. Quantum computability. *SIAM Journal on Computing*, 26(5):1524–1540, 1997.

[AH87] L. Adleman and M. Huang. Recognizing primes in random polynomial time. In *Proceedings of the 19th ACM Symposium on the Theory of Computing*, pages 462–469. ACM, New York, 1987.

[Ajt83] M. Ajtai. σ_1^1 formulea on finite structures. *Journal of Pure and Applied Logic*, 24:1–48, 1983.

[AKL$^+$79] R. Aleliunas, R. Karp, R. Lipton, L. Lovász, and C. Rackoff. Random walks, universal traversal sequences, and the complexity of maze problems. In *Proceedings of the 20th IEEE Symposium on Foundations of Computer Science*, pages 218–223. IEEE, New York, 1979.

[AKS02] M. Agrawal, N. Kayal, and N. Saxena. PRIMES is in P. Unpublished manuscript, Indian Institute of Technology Kanpur, 2002.

[ALM$^+$98] S. Arora, C. Lund, R. Motwani, M. Sudan, and M. Szegedy. Proof verification and the hardness of approximation problems. *Journal of the ACM*, 45(3):501–555, May 1998.

[AM77] L. Adleman and K. Manders. Reducibility, randomness, and intractibility. In *Proceedings of the 9th ACM Symposium on the Theory of Computing*, pages 151–163. ACM, New York, 1977.

[Aro98] S. Arora. Polynomial time approximation schemes for Euclidean traveling salesman and other geometric problems. *Journal of the ACM*, 45(5):753–782, September 1998.

[AS89] K. Ambos-Spies. On the relative complexity of hard problems for complexity classes without complete problems. *TCS*, 64:43–61, 1989.

[AS98] S. Arora and S. Safra. Probabilistic checking of proofs: A new characterization of NP. *Journal of the ACM*, 45(1):70–122, January 1998.

[ATWZ97] R. Armoni, A. Ta-Shma, A. Wigderson, and S. Zhou. $SL \subseteq L^{\frac{4}{3}}$. In *Proceedings of the 29th ACM Symposium on the Theory of Computing*, pages 230–239. ACM, New York, 1997.

[Bab85] L. Babai. Trading group theory for randomness. In *Proceedings of the 17th ACM Symposium on the Theory of Computing*, pages 421–429. ACM, New York, 1985.

[BBBV97] C. Bennett, E. Bernstein, G. Brassard, and U. Vazirani. Strengths and weaknesses of quantum computing. *SIAM Journal on Computing*, 26(5):1510–1523, 1997.

[BBC+98] R. Beals, H. Buhrman, R. Cleve, M. Mosca, and R. de Wolf. Quantum lower bounds by polynomials. In *Proceedings of the 39th IEEE Symposium on Foundations of Computer Science*, pages 352–361. IEEE, New York, 1998.

[BCD+89] A. Borodin, S. A. Cook, P. W. Dymond, W. L. Ruzzo, and M. Tompa. Two applications of inductive counting for complementaion problems. *SIAM J. Computing*, 13:559–578, 1989.

[BFL91] L. Babai, L. Fortnow, and C. Lund. Non-deterministic exponential time has two-prover interactive protocols. *Computational Complexity*, 1(1):3–40, 1991.

[BFLS91] L. Babai, L. Fortnow, L. Levin, and M. Szegedy. Checking computations in polylogarithmic time. In *Proceedings of the 23rd ACM Symposium on the Theory of Computing*, pages 21–31. ACM, New York, 1991.

[BFNW93] L. Babai, L. Fortnow, N. Nisan, and A. Wigderson. BPP has subexponential simulations unless EXPTIME has publishable proofs. *Computational Complexity*, 3:307–318, 1993.

[BGKW88] M. Ben-Or, S. Goldwasser, J. Kilian, and A. Wigderson. Multi-prover interactive proofs: How to remove intractability assumptions. In *Proceedings of the 20th ACM Symposium on the Theory of Computing*, pages 113–131. ACM, New York, 1988.

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

[BH77] L. Berman and H. Hartmanis. On isomorphisms and density of NP and other complete sets. *SIAM Journal on Comput.*, 6:305–322, 1977.

[BH97] G. Brassard and P. Høyer. An exact quantum polynomial-time algorithm for Simon's problem. In *Proceedings of the 5th Israeli Symposium on Theory of Computing and Systems (ISTCS'97)*, pages 12–23. IEEE, New York, 1997.

[BHZ87] R. Boppana, J. Håstad, and S. Zachos. Does co-NP have short interactive proofs? *Information Processing Letters*, 25(2):127–132, 1987.

[Blu67] M. Blum. A machine-independent theory of the complexity of recursive functions. *Journal of the ACM*, 14(2):322–336, April 1967.

[BM84] M. Blum and S. Micali. How to generate cryptographically strong sequences of pseudo-random bits. *SIAM Journal on Computing*, 13:850–864, 1984.

[BM88] L. Babai and S. Moran. Arthur-Merlin games: a randomized proof system, and a hierarchy of complexity classes. *Journal of Computer and System Sciences*, 36(2):254–276, 1988.

[Bor72] A. Borodin. Computational complexity and the existence of complexity gaps. *Journal of the ACM*, 19(1):158–174, January 1972.

[BP98] P. Beame and T. Pitassi. Propositional proof complexity: Past, present and future. *Bull. of the EATCS*, 65:66–89, 1998.

[BRS95] R. Beigel, N. Reingold, and D. Spielman. PP is closed under intersection. *Journal of Computer and System Sciences*, 50(2):191–202, 1995.

[Bus87] S. Buss. Polynomial size proofs of the pigeon hole principle. *Journal of Symbolic Logic*, 57:916–927, 1987.

[BV97] E. Bernstein and U. Vazirani. Quantum complexity theory. *SIAM Journal on Computing*, 26(5):1411–1473, 1997.

[Can74] G. Cantor. Ueber eine Eigenschaft des Inbegriffes aller reellen algebraischen Zahlen. *Crelle's Journal*, 77:258–262, 1874.

[CCL90] J. Cai, A. Condon, and R. Lipton. On bounded round multi-prover interactive proof systems. In *Proceedings of the 5th IEEE Structure in Complexity Theory Conference*, pages 45–54. IEEE, New York, 1990.

[CCL91] J. Cai, A. Condon, and R. Lipton. PSPACE is provable by two provers in one round. In *Proceedings of the 6th IEEE Structure in Complexity Theory Conference*, pages 110–115. IEEE, New York, 1991.

[CCL92] J. Cai, A. Condon, and R. Lipton. On games of incomplete information. *Theoretical Computer Science*, 103(1):25–38, 1992.

[CKS81] A. Chandra, D. Kozen, and L. Stockmeyer. Alternation. *Journal of the ACM*, 28:114–133, 1981.

[Cla00] Clay Mathematics Institute. Millennium prize problems. *http://www.claymath.org/prizeproblems/*, 2000.

[Cob64] A. Cobham. The intrinsic computational difficulty of functions. In *Proceedings of the 1964 International Congress for Logic, Methodology, and Philosophy of Science*, pages 24–30. North-Holland, Amsterdam, 1964.

[Coo71] S. Cook. The complexity of theorem-proving procedures. In *Proc. 3rd ACM Symp. Theory of Computing*, pages 151–158, 1971.

[Coo73] S. Cook. A hierarchy for nondeterministic time complexity. *Journal of Computer and System Sciences*, 7(4):343–353, August 1973.

[Coo83] S. Cook. An overview of computational complexity. *Communications of the ACM*, 26(6):400–408, June 1983.

[CR73] S. Cook and R. Reckhow. Time bounded random access machines. *JCSS*, 7(4):354–375, 1973.

[Deu85] D. Deutsch. Quantum theory, the Church-Turing principle and the universal quantum computer. *Proceedings of the Royal Society of London A*, 400:97, 1985.

[DJ92] D. Deutsch and R. Jousza. Rapid solution of problems by quantum computation. *Proceedings of the Royal Society of London A*, 439:553, 1992.

[Edm65a] J. Edmonds. Maximum matchings and a polyhedron with 0,1-vertices. *Journal of Research at the National Bureau of Standards (Section B)*, 69B:125–130, 1965.

[Edm65b] J. Edmonds. Paths, trees and flowers. *Canadian Journal of Mathematics*, 17:449–467, 1965.

[Fag73] R. Fagin. *Contributions to the model theory of finite structures.* Ph.D. Thesis, U.C. Berkeley, 1973.

[Fag74] R. Fagin. Generalized first-order spectra and polynomial-time recognizable sets. In *Complexity of Computation (ed. R. Karp)*, pages 27–41. SIAM-AMS Proc. 7, 1974.

[Fei91] U. Feige. On the success probability of the two provers in one round proof systems. In *Proceedings of the 6th IEEE Structure in Complexity Theory Conference*, pages 116–123. IEEE, New York, 1991.

[Fel86] P. Feldman. The optimum prover lives in PSPACE. Manuscript, 1986.

[Fey82] R. Feynman. Simulating physics with computers. *International Journal of Theoretical Physics*, 21:467, 1982.

[FFK94] S. Fenner, L. Fortnow, and S. Kurtz. Gap-definable counting classes. *Journal of Computer and System Sciences*, 48(1):116–148, 1994.

[FFS88] U. Feige, A. Fiat, and A. Shamir. Zero knowledge proofs of identity. *Journal of Cryptology*, 1(2):77–94, 1988.

[FGHP99] S. Fenner, F. Green, S. Homer, and R. Pruim. Determining acceptance possibility for a quantum computation is hard for PH. *Proceedings of the Royal Society of London*, 455:3953–3966, 1999.

[FGL+96] U. Feige, S. Goldwasser, L. Lovász, S. Safra, and M. Szegedy. Interactive proofs and the hardness of approximating cliques. *Journal of the ACM*, 43(2):268–292, March 1996.

[FGM+89] M. Fürer, O. Goldreich, Y. Mansour, M. Sipser, and S. Zachos. On completeness and soundness in interactive proof systems. In S. Micali, editor, *Randomness and Computation*, volume 5 of *Advances in Computing Research*, pages 429–442. JAI Press, Greenwich, 1989.

[FL92] U. Feige and L. Lovász. Two-prover one-round proof systems: Their power and their problems. In *Proceedings of the 24th ACM Symposium on the Theory of Computing*, pages 733–744. ACM, New York, 1992.

[For97] L. Fortnow. Counting complexity. In *In Lane Hemaspaandra and Alan Selman, editor, Complexity Theory Retrospective II*, pages 81–107. Springer, New York, 1997.

[FR99] L. Fortnow and J. Rogers. Complexity limitations on quantum computation. *Journal of Computer and System Sciences*, 59(2):240–252, 1999.

[FRS94] L. Fortnow, J. Rompel, and M. Sipser. On the power of multi-prover interactive protocols. *Theoretical Computer Science A*, 134:545–557, 1994.

[Gil77] J. Gill. Computational complexity of probabilistic complexity classes. *SIAM Journal on Computing*, 6:675–695, 1977.

[GJ79] M. Garey and D. Johnson. *Computers And Intractability: A Guide To The Theory of NP-Completeness.* W. H. Freeman, San Francisco, 1979.

[GK99] S. Goldwasser and J. Kilian. Primality testing using elliptic curves. *Journal of the ACM*, 46(4):450–472, July 1999.

[GKL93] O. Goldreich, H. Krawczyk, and M. Luby. On the existence of pseudo-random generators. *SIAM Journal on Computing*, 22(6):1163–1175, December 1993.

[GL89] O. Goldreich and L. Levin. A hard-core predicate for all one-way functions. In *Proceedings of the 21st ACM Symposium on the Theory of Computing*, pages 25–32. ACM, New York, 1989.

[GMR89] S. Goldwasser, S. Micali, and C. Rackoff. The knowledge complexity of interactive proof-systems. *SIAM Journal on Computing*, 18(1):186–208, 1989.

[GMW91] O. Goldreich, S. Micali, and A. Wigderson. Proofs that yield nothing but their validity or all languages in NP have zero-knowledge proof systems. *Journal of the ACM*, 38(3):691–729, 1991.

[Gro96] L. Grover. A fast quantum mechanical algorithm for database search. In *Proceedings of the 28th ACM Symposium on the Theory of Computing*, pages 212–219. ACM, New York, 1996.

[GS89] S. Goldwasser and M. Sipser. Private coins versus public coins in interactive proof systems. In S. Micali, editor, *Randomness and Computation*, volume 5 of *Advances in Computing Research*, pages 73–90. JAI Press, Greenwich, 1989.

[Har81] J. Hartmanis. Observations about the development of theoretical computer science. *Annals of the History of Computing*, 3(1):42–51, 1981.

[Har82] J. Hartmanis. A note on natural complete sets and godel numberings. *TCS*, 17:75–89, 1982.

[Har86] J. Hartmanis. Gödel, Von neumann and the P=?NP problem. In *Current Trends in Theoretical Computer Science*, pages 445–450. World Scientific Press, New York, 1986.

[Har94] J. Hartmanis. Turing Award Lecture: On computational complexity and the nature of computer science. *Communications of the ACM*, 37(10):37–43, October 1994.

[Hås89] J. Håstad. Almost optimal lower bounds for small depth circuits. In S. Micali, editor, *Randomness and Computation*, volume 5 of *Advances in Computing Research*, pages 143–170. JAI Press, Greenwich, 1989.

[Hås97] J. Håstad. Some optimal inapproximabiity results. In *Proceedings of the 29th ACM Symposium on the Theory of Computing*, pages 1–10. ACM, New York, 1997.

[HB75] J. Hartmanis and T. Baker. On simple godel numberings and translations. *SIAM Journal on Computing*, 4:1–11, 1975.

[HB78] J. Hartmanis and L. Berman. On polynomial time isomorphisms and some new complete sets. *JCSS*, 16:418–422, 1978.

[HILL99] J. Håstad, R. Impagliazzo, L. Levin, and M. Luby. A pseudorandom generator from any one-way function. *SIAM Journal on Computing*, 28(4):1364–1396, August 1999.

[Hoc95] D. Hochbaum. *Approximation Algorithms for NP-Hard Problems.* PSW Publishing Company, Boston, 1995.

[HS65] J. Hartmanis and R. Stearns. On the computational complexity of algorithms. *Transactions of the American Mathematical Society*, 117:285–306, 1965.

[HS66] F. Hennie and R. Stearns. Two-tape simulation of multitape Turing machines. *Journal of the ACM*, 13(4):533–546, October 1966.

[HS97] L. Hemaspaandra and A. Selman. *Complexity Theory Retrospective II.* Springer, New York, 1997.

[HS00] S. Homer and A. Selman. *Computability and Complexity Theory.* Springer, 2000.

[Iba72] O. Ibarra. A note concerning nondeterministic tape complexities. *Journal of the ACM*, 19(4):608–612, 1972.

[Imm82] N. Immerman. Relational queries computable in polynomial time. In *Proc. 14th Symposium on Theory of Computation*, pages 147–152. ACM Press, 1982.

[Imm83] N. Immerman. Languages which capture complexity classes. In *Proc. 15th Symposium on Theory of Computation*, pages 760–778. ACM Press, 1983.

[Imm88] N. Immerman. Nondeterministic space is closed under complementation. *SIAM Journal on Computing*, 17(5):935–938, 1988.

[IW97] R. Impagliazzo and A. Wigderson. P = BPP if E requires exponential circuits: Derandomizing the XOR lemma. In *Proceedings of the 29th ACM Symposium on the Theory of Computing*, pages 220–229. ACM, New York, 1997.

[Jon75] N. Jones. Space-bounded reducibility among combinatorial problems. *Journal of Computer and System Sciences*, 11:68–85, 1975.

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

[Kar72] R. Karp. Reducibility among combinatorial problems. In *Complexity of Computer Computations*, pages 85–104. Plenum Press, New York, 1972.

[Kar86] R. Karp. Combinatorics, complexity and randomness. *Communications of the ACM*, 29(2):98–109, February 1986.

[KMR87] S. Kurtz, S. Mahaney, and J. Royer. Progress on collapsing degrees. In *Proc. Structure in Complexity Theory Second Annual Conference*, pages 126–131, 1730 Massachusetts Avenue, N.W., Washington, D.C. 20036-1903, 1987. Computer Society Press of the IEEE.

[KMR89] S. Kurtz, S. Mahaney, and J. Royer. The isomorphism conjecture fails relative to a random oracle (extended abstract). In *ACM Symposium on Theory of Computing*, pages 157–166, 1989.

[KN96] E. Kushilevitz and N. Nisan. *Communication Complexity.* Cambridge University Press, Cambridge, 1996.

[KP89] J. Krajicek and P. Pudlak. Propositional proof systems, the consistency of first order theories and the complexity of computation. *Journal of Symbolic Logic*, 53(3):1063–1079, 1989.

[KW00] A. Kitaev and J. Watrous. Parallelization, amplification, and exponential time simulation of quantum interactive proof systems. In *Proceedings of the 32nd ACM Symposium on the Theory of Computing*, pages 608–617. ACM, New York, 2000.

[Lau83] C. Lautemann. BPP and the polynomial hierarchy. *Information Processing Letters*, 17(4):215–217, 1983.

[Lev73] L. Levin. Universal sorting problems. *Problems of Information Transmission*, 9:265–266, 1973. English translation of original in *Problemy Peredaci Informacii.*

[LFKN92] C. Lund, L. Fortnow, H. Karloff, and N. Nisan. Algebraic methods for interactive proof systems. *Journal of the ACM*, 39(4):859–868, 1992.

[LS91] D. Lapidot and A. Shamir. Fully parallelized multi prover protocols for NEXP-time. In *Proceedings of the 32nd IEEE Symposium on Foundations of Computer Science*, pages 13–18. IEEE, New York, 1991.

[LZ77] R. Lipton and E. Zalcstein. Word problems solvable in logspace. *Journal of the ACM*, 3:522–526, 1977.

[Mah82] S. Mahaney. Sparse complete sets for NP: solution of a conjecture of Berman and Hartmanis. *Journal of Comput. System Sci.*, 25:130–143, 1982.

[MM69] E. McCreight and A. Meyer. Classes of computable functions defined by bounds on computation. In *Proceedings of the First ACM Symposium on the Theory of Computing*, pages 79–88. ACM, New York, 1969.

[MS72] A. Meyer and L. Stockmeyer. The equivalence problem for regular expressions with squaring requires exponential space. In *Proc. of the 13th IEEE Symposium on Switching and Automata Theory*, pages 125–129, 1730 Massachusetts Avenue, N.W., Washington, D.C. 20036-1903, 1972. Computer Society Press of the IEEE.

[MY85] S. Mahaney and P. Young. Orderings of polynomial isomorphism types. *Theor. Comput. Sci.*, 39(2):207–224, August 1985.

[Myh55] J. Myhill. Creative sets. *Zeitschrift für Mathematische Logik und Grundlagen der Mathematik*, 1:97–108, 1955.

[Myh60] J. Myhill. Linear bounded automata. Technical Note 60–165, Wright-Patterson Air Force Base, Wright Air Development Division, Ohio, 1960.

[Nis92] N. Nisan. Pseudorandom generators for space-bounded computation. *Combinatorica*, 12(4):449–461, 1992.

[NSW92] N. Nisan, E. Szemerédi, and A. Wigderson. Undirected connectivity in $O(\log^{1.5} n)$ space. In *Proceedings of the 33rd IEEE Symposium on Foundations of Computer Science*, pages 24–29. IEEE, New York, 1992.

[NT95] N. Nisan and A. Ta-Shma. Symmetric *logspace* is closed under complement. *Chicago Journal of Theoretical Computer Science*, 1995(1), June 1995.

[NW94] N. Nisan and A. Wigderson. Hardness vs. randomness. *Journal of Computer and System Sciences*, 49:149–167, 1994.

[PY91] C. Papadimitriou and M. Yannakakis. Optimization, approximation, and complexity classes. *Journal of Computer and System Sciences*, 43:425–440, 1991.

[Rab63] M. Rabin. Real time computation. *Israel Journal of Mathematics*, 1:203–211, 1963.

[Raz85a] A. Razborov. Lower bounds of monotone complexity of the logical permanent function. *Mathematical Notes of the Academy of Sciences of the USSR*, 37:485–493, 1985.

[Raz85b] A. Razborov. Lower bounds on the monotone complexity of some boolean functions. *Doklady Akademii Nauk SSSR*, 281(4):798–801, 1985. In Russian. English Translation in [Raz85c].

[Raz85c] A. Razborov. Lower bounds on the monotone complexity of some boolean functions. *Soviet Mathematics–Doklady*, 31:485–493, 1985.

[Raz98] R. Raz. A parallel repetition theorem. *SIAM Journal on Computing*, 27(3):763–803, June 1998.

[RF65] S. Ruby and P. Fischer. Translational methods and computational complexity. In *Proceedings of the Sixth Annual Symposium on Switching Circuit Theory and Logical Design*, pages 173–178, New York, 1965. IEEE.

[RGR95] H. Hoover R. Greenlaw and W. Ruzzo. *Limits to Parallel Computation: P-Completeness Theory.* Oxford University Press, Oxford, 1995.

[Rob65] J.A. Robinson. A machine oriented logic based on resolution. *Journal of the ACM*, 12(1):23–41, 1965.

[Sav70] W. Savitch. Relationship between nondeterministic and deterministic tape classes. *Journal of Computer and System Sciences*, 4:177–192, 1970.

[Sav73] W. Savitch. Maze recognizing automata and nondeterministic tape complexity. *JCSS*, 7:389–403, 1973.

[Sch90] U. Schoning. The power of counting. In *In Alan Selman, editor, Complexity Theory Retrospective*, pages 204–223. Springer, New York, 1990.

[Sel88] A. Selman. *Complexity Theory Retrospective.* Springer, New York, 1988.

[SFM78] J. Seiferas, M. Fischer, and A. Meyer. Separating nondeterministic time complexity classes. *Journal of the ACM*, 25(1):146–167, 1978.

[Sha49] C.E. Shannon. Communication in the presence of noise. *IRE*, 37:10–21, 1949.

[Sha92] A. Shamir. IP = PSPACE. *Journal of the ACM*, 39(4):869–877, 1992.

[Sho97] P. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Journal on Computing*, 26(5):1484–1509, 1997.

[Sim97] D. Simon. On the power of quantum computation. *SIAM Journal on Computing*, 26(5):1474–1483, 1997.

[Sip83] M. Sipser. A complexity theoretic approach to randomness. In *Proceedings of the 15th ACM Symposium on the Theory of Computing*, pages 330–335. ACM, New York, 1983.

[Sip92] M. Sipser. The history and status of the P versus NP question. In *Proceedings of the 24th ACM Symposium on the Theory of Computing*, pages 603–618. ACM, New York, 1992.

[Smo87] R. Smolensky. Algebraic methods in the theory of lower bounds for boolean circuit complexity. In *Proc. 19th Symposium on Theory of Computation*, pages 77–82. ACM Press, 1987.

[Smu61] R. Smullyan. *Theory of Formal Systems*, volume 47 of *Annals of Mathematical Studies*. Princeton University Press, 1961.

[SS77] R. Solovay and V. Strassen. A fast Monte-Carlo test for primality. *SIAM Journal on Computing*, 6:84–85, 1977. See also erratum 7:118, 1978.

[Ste94] R. Stearns. Turing award lecture: It's time to reconsider time. *Communications of the ACM*, 37(11):95–99, November 1994.

[Sto76] L. Stockmeyer. The polynomial-time hierarchy. *Theor. Computer Science*, 3:1–22, 1976.

[SZ99] M. Saks and S. Zhou. $\mathrm{BP}_H\mathrm{SPACE(S)} \subseteq \mathrm{DPSPACE}(\mathrm{S}^{3/2})$. *Journal of Computer and System Sciences*, 58(2):376–403, April 1999.

[Sze88] R. Szelepcsényi. The method of forced enumeration for nondeterministic automata. *Acta Informatica*, 26:279–284, 1988.

[Tod91] S. Toda. PP is as hard as the polynomial-time hierarchy. *SIAM Journal on Computing*, 20(5):865–877, 1991.

[Tra64] B. Trakhtenbrot. Turing computations with logarithmic delay. *Algebra i Logika*, 3(4):33–48, 1964.

[Tra84] R. Trakhtenbrot. A survey of Russian approaches to *Perebor* (brute-force search) algorithms. *Annals of the History of Computing*, 6(4):384–400, 1984.

[Tse68] G. S. Tseitin. On the complexity of derivations in the propositional calculus. In *In Studies in Constructive Mathematics and Mathematical Logic*, volume Part II. Consultants Bureau, New-York-London, 1968.

[Tur36] A. Turing. On computable numbers, with an application to the Etscheidungs problem. *Proceedings of the London Mathematical Society*, 42:230–265, 1936.

[Urq87] A. Urquhart. Hard examples for resolution. *Journal of the ACM*, 34:209–219, 1987.

[Val79] L. Valiant. The complexity of computing the permanent. *Theoretical Computer Science*, 8:189–201, 1979.

[Var82] M. Vardi. Complexity of relational query languages. In *Proc. 14th Symposium on Theory of Computation*, pages 137–146. ACM Press, 1982.

[Wat99] J. Watrous. PSPACE has constant-round quantum interactive proof systems. In *Proceedings of the 40th IEEE Symposium on Foundations of Computer Science*, pages 112–119. IEEE, New York, 1999.

[Yam62] H. Yamada. Real-time computation and recursive functions not real-time computable. *IEEE Transactions on Computers*, 11:753–760, 1962.

[Yao90] A. Yao. Coherent functions and program checkers. In *Proceedings of the 22nd ACM Symposium on the Theory of Computing*, pages 84–94. ACM, New York, 1990.