My Favorite Ten Complexity Theorems of the Past Decade

Lance Fortnow[⋆]

Department of Computer Science
The University of Chicago
1100 East 58th Street
Chicago, Illinois 60637

Abstract. We review the past ten years in computational complexity theory by focusing on ten theorems that the author enjoyed the most. We use each of the theorems as a springboard to discuss work done in various areas of complexity theory.

1 Introduction

Just about ten years ago, in the spring of 1985, I enrolled in a graduate computational complexity course taught by Juris Hartmanis at Cornell. That course marked the beginning of study and research in computational complexity that has been a major part of my life ever since.

As a decade has passed, I find it useful to review where complexity theory has come during those years. I have found that complexity theory has flourished during that time. No twenty page paper could possibly do justice to the past ten years in complexity theory.

Instead I have taken a different approach. I have isolated ten theorems that I have enjoyed the most during the past decade following a few self-imposed guidelines (See Section 1.1). I have then used these theorems as a basis to describe many other results in complexity theory using each theorem as a springboard into a subarea of computational complexity.

Please note that each area could have a twenty page survey of its own. I cannot possibly mention all the important results in any given area. Nor am I able to bring in all the different subareas in complexity theory.

I have found the past ten years in complexity theory quite exciting. Circuit complexity has come of age during the past ten years and has provided us with a rich source of combinatorial problems. Interactive proof systems have surprised us all with their ability to use randomness to simulate alternation and their important connections to program testing and hardness of approximation algorithms. Algebraic techniques now seem to pervade all areas of complexity especially circuit complexity, interactive proof systems and counting complexity. We have also seen a lot of interest in probabilistic computation both in its

[⋆] Email: fortnow@cs.uchicago.edu. Partially supported by NSF grant CCR 92-53582.

power in interactive proof systems and the many successful attempts to reduce and eliminate randomness in various computation models.

Computational complexity theory has had steady growth since its inception in the '60s. I would guess that the number of active researchers in computational complexity theory has doubled over the past decade. Complexity theory now has a major conference, *The IEEE Structure in Complexity Theory Conference*, as well as many smaller conferences and workshops.

However, complexity theory has had its disappointments. We seem no closer to solving the famous $\mathbf{P} \neq \mathbf{NP}$ problem than we were ten years ago. I have found that really only one theorem (Theorem 4) gives us such fundamental results about complexity classes that would make it into an undergraduate course. Despite these shortcomings, this paper shows that complexity theory has provided us with great theorems throughout the past ten years.

1.1 Guidelines

In choosing the ten theorems I used the following guidelines:

- **Theorems chosen from the broad area of computational complexity theory:** There are several good theorems in other areas of theory and computer science. However, I have my expertise in computational complexity theory. I would not make a good judge in other areas.
- **Theorems chosen from the past ten years:** It would not be fair to judge theorems before I was actively involved in complexity theory.
- **Theorems chosen for importance of result, originality and difficulty of technique and relationships with other results in complexity theory.**
- **No theorems proven at The University of Chicago:** I do not want to play favorites.
- **Theorems chosen for diversity:** I wanted to get a representation of many different areas of complexity theory.
- **Theorems chosen as results instead of for people:** I did not make any attempt to choose people, instead I chose results that I enjoyed.

2 The Theorems

We present the ten favorite theorems in rough chronological order.

2.1 Branching Programs

In the last decade we have seen algebra play a much larger role in complexity theory than ever before. Usually in these results algebra plays no role in the statement of the theorem but simple algebraic properties lead to very beautiful results.

No theorem captures this algebraic beauty more than our first theorem, a surprising characterization of bounded-width branching programs due to Barrington:

Favorite Theorem 1 ([Bar89]) *Bounded-width polynomial-size branching programs recognize exactly those languages in* $\mathbf{NC}^1$.

A branching program is a rooted directed acyclic graph where each internal vertex is labelled by a variable name and for each input symbol there is a single edge leaving that vertex labelled by that symbol. The leaf vertices are labelled by "Accept" or "Reject". The machine proceeds along a root to leaf path by examining the input of the current vertex's label and then follows the edge corresponding to the value of that input. The machine accepts or rejects when it comes to the corresponding leaf vertex.

A language L has bounded-width branching programs if for some k, for all n there is a width k branching program using n inputs and at most $n^{O(1)}$ nodes that accepts exactly those strings of length n in L.

Recall that $\mathbf{NC}^1$ consists of the languages accepted by constant fan-in and logarithmic depth circuits.

One can easily show that every language that accepts bounded-width poly-size branching programs must lie in $\mathbf{NC}^1$ by divide and conquer. However, it did not seem possible to count in a bounded-width program so many believed that majority, an $\mathbf{NC}^1$ function, did not have bounded-width branching programs. In fact, Yao [Yao83] showed a super-polynomial lower bound for width-two branching programs to compute majority.

Barrington used the fact that S_5, the set of permutations on five elements, formed a nonsolvable group. More precisely there are two five cycles $\sigma_1 = (12345)$ and $\sigma_2 = (13542)$ in S_5, such that $\sigma_1\sigma_2\sigma_1^{-1}\sigma_2^{-1} = (13254)$ is a five cycle. Barrington uses this simple fact to capture the computation of a circuit by just remembering one of these cycles.

Barrington's result has some implications for the complexity class **PSPACE**. By the Chandra, Kozen and Stockmeyer [CKS81] result relating **PSPACE** with alternating polynomial-time, one can view **PSPACE** computation as a uniform $\mathbf{NC}^1$ circuit of polynomial-depth.

Cai and Furst [CF91] showed that every **PSPACE** language L can be accepted by a Turing machine that has a polynomial-size "clock", makes logarithmic space computation then erases all but a few bits of its tape between clock ticks. They prove their result by simulating an exponentially long but easily describable bounded width branching program for the $\mathbf{NC}^1$ circuit that simulates the **PSPACE** machine.

Bovet, Crescenzi and Silvestri [BCS92] developed the concept of *leaf languages*. Think of a nondeterministic polynomial-time Turing machine M on input x as generating an exponentially long string $M(x)$ read off of the end of the computation paths (leaves) of M on x. Given a machine M and a set of strings A, we define the leaf language L as the set of x such that $M(x)$ is in A.

Hertrampf, Lautemann, Schwentick, Vollmer and Wagner [HLS+93] show that any language in **PSPACE** has a leaf language defined via a regular language. Their proof uses Barrington's result by simulating the branching program with a finite automaton.

2.2 Bounded-Depth Circuits

In the mid-1980's, theorems about circuits provided great excitement among complexity theorists. The techniques used in complexity theory before then did not seem to help prove the big theorems such as $\mathbf{P} \neq \mathbf{NP}$. Many theorists believed that the new combinatorial and algebraic tools used in circuits could prove some nontrivial separations in complexity classes.

Although this promise has remained unfulfilled, we have seen several separation results in low-level circuit classes as well as many new combinatorial and algebraic techniques. Because of the vast amount of effort devoted to circuit complexity in the last decade we devote Sections 2.2 and 2.3 to circuit complexity. Circuit complexity also comes up in connection with many of the other theorems mentioned in this paper.

The early work in circuit complexity came out of an attempt to create an oracle to separate the polynomial-time hierarchy from **PSPACE**. In order to create such an oracle, an interesting combinatorial problem arose: One would need to show that a simple problem, like parity, did not have small constant depth circuits.

Furst, Saxe and Sipser [FSS84] and Ajtai [Ajt83] independently showed that parity does not have constant-depth polynomial-size circuits. Yao [Yao85] showed a strong enough bound to get the desired oracle separation. Håstad greatly simplified Yao's proof and achieved near tight bounds:

Favorite Theorem 2 ([Hås89]) *Parity does not have depth d circuits with less than $2^{(1/10)n^{1/d}}$ gates.*

Håstad's proof used random restrictions where some of the variables in a potential circuit for parity are randomly set to zero or one with some probability. Håstad's main "switching" lemma showed that a random restriction of an **AND** gate of small **OR** gates resulted in an **OR** gate of small **AND** gates. Håstad then used an inductive argument by converting any depth d circuit claiming to solve parity into a depth $d-1$ circuit for parity. This switching lemma has also found several other applications subsequently.

Razborov [Raz87] and Smolensky [Smo87] show that for primes p and q, $p \neq q$, the $\mathbf{MOD}_q$ function requires an exponential number of gates for bounded-depth circuits with **AND**, **OR** and $\mathbf{MOD}_p$ gates. Their proof uses a new idea of approximating bounded-depth circuits by low-degree polynomials over a finite field.

Linial, Mansour and Nisan [LMN93] show using Fourier Transform methods how to approximate bounded-depth circuits by low-degree polynomials over the rationals and reals. Their proof uses the Håstad switching lemma. Their result has had some applications in circuit complexity and learning theory. Tarui [Tar93] shows how to approximate bounded-depth circuits by low-degree polynomials over the integers. His proof uses the Valiant-Vazirani [VV86] lemma described in Section 2.8 and can be used to give an alternative proof of part of Theorem 8.

2.3 Monotone Circuits

Håstad's Theorem (Theorem 2) led hope that circuit complexity could now lead to perhaps a combinatorial separation of machine-based complexity classes. Since it is well known that every language in **P** has polynomial-size circuits, we could separate **P** from **NP** by exhibiting some **NP** problem that does not have polynomial-size circuits. We believe that such problems exist because Karp and Lipton [KL82] showed that if every language in **NP** has polynomial-size circuits then the polynomial-hierarchy collapses.

However, proving superpolynomial (or even superlinear) lower bounds for such an **NP** problem seems extremely difficult. Razborov looked at answering this question in a restricted model. In particular, he looked at monotone circuits, i.e., circuits with no negations or negated variables. Of course a non-monotone problem cannot have such circuits so Razborov looked at a monotone **NP**-complete problem, Clique:

Favorite Theorem 3 ([Raz85b]) *The general clique function requires exponentially large monotone circuits.*

Instead of the restriction method used by Håstad (Theorem 2), Razborov uses an approximation method. Razborov shows how to approximate each **AND** and **OR** with approximate **AND** and approximate **OR**. Each approximation cannot cause too many errors in the inputs. However, he shows the final approximated circuit must error in many inputs. Razborov then concludes that the circuits must have had lots of gates. Alon and Boppana [AB87] strengthen Razborov's bounds.

Initially, many thought that perhaps we could extend these techniques into the general case. Now it seems that Razborov's theorem says much more about the weakness of monotone models than about the hardness of **NP** problems. Razborov [Raz85a] showed that matching also does not have polynomial-size monotone circuits. However, we know that matching does have a polynomial-time algorithm [Edm65] and thus polynomial-size nonmonotone circuits. Tardos [Tar88] exhibited a monotone problem that has an exponential gap between its monotone and nonmonotone circuit complexity.

Other results on monotone complexity come out of communication complexity. Karchmer and Wigderson [KW90] show a direct connection between a communication complexity game and circuit depth. They use this characterization to show that graph connectivity requires $\Omega(\log^2 n)$ depth in polynomial-size monotone circuits. Razborov [Raz90] uses more general matrix methods to prove similar lower bounds for other problems. Raz and Wigderson [RW92] show that monotone circuits for matching require linear depth.

We also have seen several arguments that the techniques used in Theorems 2 and 3 will not help us settle the big questions like $\mathbf{P} = \mathbf{NP}$. Razborov [Raz89] himself showed that approximation techniques like those used in the proof of Theorem 3 will not work in the general case. Razborov [Raz94] later shows a fragment of arithmetic strong enough to prove Theorems 2 and 3 cannot show that **NP** does not have polynomial-size circuits. Razborov and Rudich [RR94] show that under a strong cryptographic assumption, combinatorial proofs fulfilling certain largeness and constructivity properties cannot show that **NP** does not have polynomial-size circuits.

This last decade started with the promise of great separation results using combinatorial techniques but ends finding us no closer to solving the big open questions in complexity theory.

2.4 Nondeterministic Space

We usually use many factors to determine the "quality" of a result. If we look solely at the importance of the statement of the theorem one result stands out. Immerman and Szelepcsényi independently proved the following fundamental theorem about nondeterministic space complexity:

Favorite Theorem 4 ([Imm88, Sze88]) *Nondeterministic Space is closed under complementation.*

From the very beginnings of complexity theory, theorists have thought hard about the relationship among the various deterministic and nondeterministic time and space classes. The fundamental theorems in complexity theory relate these classes, such as the basic time and space hierarchy theorems due to Hartmanis and Stearns [HS65] in 1965.

While the relationships between deterministic and nondeterministic time classes remain the single most important open area in complexity, we know much more about the related space complexity questions. In 1970, Savitch [Sav70] showed that one can simulate a $S(n)$ space nondeterministic Turing machine by a $S^2(n)$ space deterministic machine. Thus, for example, **NPSPACE** = **PSPACE**.

In the two and a half decades since, no one has improved upon this quadratic increase from nondeterministic to deterministic space. From Immerman and Szelepcsényi, we do get that only a linear blow up going from nondeterministic (existential) space to conondeterministic (universal) space.

We can think of the computation of a $s(n)$ space-bounded Turing machine M on input x as a directed graph G on $2^{O(s(|x|))}$ nodes where each node represents a configuration of $M(x)$ and each edge represents a transition. Note that $M(x)$ accepts if and only if there is a directed path from the initial configuration to an accepting configuration.

The proof of Theorem 4 uses inductive counting to show that there is no path from s to t in an n-node graph in nondeterministic $O(\log n)$ space. For a directed graph G and a node s of G define the value c_i as the number of nodes reachable from s in G by a path of length at most i. They show how using a nondeterministic log space machine they can verify the value of c_i inductively as i goes from 0 to n. They can then use c_n to determine if a node t is not reachable from node s.

Though Immerman and Szelepcsényi have similar proofs, they each went about this theorem from different angles. Immerman has built up a theory giving logical characterizations of complexity classes. Immerman looked at a characterization of nondeterministic logarithmic space (**NL**) as sets of languages describable by first-order expressions with transitive closure. Immerman showed that having first-order with the complement of transitive closure led to the same class. He then translated this proof to show that **NL** = **coNL** and then that all nondeterministic space classes are closed under complementation.

Szelepcsényi looked at the question of whether context sensitive languages are closed under complement. This was one of the few open formal languages questions in Hopcroft and Ullman [HU79]. Landweber [Lan63] showed that context sensitive languages accepted exactly those sets in nondeterministic linear space. Thus by proving that nondeterministic space is closed under complement, Immerman and Szelepcsényi also showed that context sensitive languages are closed under complementation.

Borodin, Cook, Dymond, Ruzzo and Tompa [BCD$^+$89b] extended the inductive counting arguments used in the proof of Theorem 4 to show two other results:

1. **LOGCFL**, the class of languages log-space reducible to context-free languages, is closed under complementation.
2. Undirected $s-t$ connectivity has an errorless log-space expected polynomial-time probabilistic algorithm.

2.5 Cryptographic Assumptions

In another trend during the past decade, researchers have looked at complexity issues arising from cryptography. Cryptographers used many different hardness assumptions in order to prove the security of their various protocols. Some complexity theorists looked at the relative hardness of these assumptions.

Cryptographers designed their protocols with either hardness assumptions about particular languages like factoring or discrete logarithm, or with some general assumption. Usually a general assumption took one of the following three forms:

1. One-way functions exist.
2. Pseudorandom generators exist.
3. Trap-door functions exist.

We will not define these notions formally in this paper, but keep in mind that cryptographers usually require that hardness occurs for most inputs. It was well known that the second two assumptions imply the first but ten years ago the other directions remained open.

Impagliazzo, Levin and Luby settled one relationship:

Favorite Theorem 5 ([ILL89]) *Pseudorandom generators can be constructed from any one-way function.*

This problem has a very interesting history. Blum and Micali [BM84] show how to create pseudorandom generators based on the hardness of discrete logarithm. Yao [Yao82] generalizes their algorithm to create a generator based on any one-way permutation. Levin [Lev87] shows a technical one-way property of functions that he can use to create secure pseudorandom generators. Goldreich, Krawczyk and Luby [GKL93] show how to convert a "regular" one-way function to a pseudorandom generator. Impagliazzo, Levin and Luby then showed Theorem 5. Finally, Håstad [Hås90] extended the techniques of Impagliazzo, Levin and Luby to show how to construct pseudo-random functions from any uniformly one-way function.

The proof of Theorem 5 uses a new idea of computational entropy. In other words, they use a one-way function to create a distribution that may not have large real entropy or randomness but does have large entropy in a computational sense. They then use this pseudoentropy distribution to create the pseudorandom generator. Their techniques make important use of a result of Goldreich and Levin [GL89] that shows how to get a "hard-core" bit out of any one-way function.

The relationship between one-way functions and trap-door functions appear much more difficult to resolve. Impagliazzo and Rudich [IR89] give some relativization results that shed light on this difficulty.

2.6 Isomorphism Conjecture

Back in the late 70's, Berman and Hartmanis [BH77] looked at the structure of the known **NP**-complete sets via many-one reductions. They developed some conditions under which one could show two sets were *isomorphic*, i.e., there existed a polynomial-time computable polynomial-time invertible bijection reducing one to the other. They then showed that all the **NP**-complete problems known at that time were isomorphic. They conjectured that all **NP**-complete problems are isomorphic. Since their conjecture implies $\mathbf{P} \neq \mathbf{NP}$ (or one would have finite sets isomorphic to infinite sets) they thought that maybe that difficult problem could be attacked by looking at the structure of complete sets.

The isomorphism conjecture would imply that every **NP**-complete set A has exponential density, i.e., for all n, $|A \cap \Sigma^{\leq n}| \geq 2^{n^{\Omega(1)}}$. Mahaney [Mah82] gave some evidence to this direction by showing that there are no sparse (polynomial dense) **NP**-complete sets unless $\mathbf{P} = \mathbf{NP}$.

What about complete sets via other notions of reductions. Karp and Lipton [KL82] show that if there exist sparse sets **NP**-hard via Turing reductions then the polynomial-time hierarchy collapses to the second level. However, what kind of reductions to sparse sets can we rule out by only assuming $\mathbf{P} \neq \mathbf{NP}$?

Many people tried extending the techniques of Mahaney. Ogiwara and Watanabe made large progress with a brand new trick to solve the problem for bounded truth-table reductions, i.e., nonadaptive reductions that make a constant number of queries to the set.

Favorite Theorem 6 ([OW91]) *There are no sparse sets hard for* **NP** *via polynomial-time bounded truth-table queries unless* $\mathbf{P} = \mathbf{NP}$.

The proof uses a new technique known as "left-sets". For a satisfiable formula, Ogiwara and Watanabe create a step function based on the lexicographically least witness. They then use the reduction to find this step and thus a witness.

Homer and Longpré [HL94] give a very clear proof of Theorem 6 with stronger bounds.

Complexity theorists have proven many other interesting results relating to the isomorphism conjecture in the past decade. Joseph and Young [JY85] conjecture that one could use certain one-way functions to create **NP**-complete sets nonisomorphic to **SAT**. Ko, Long and Du [KLD87] show that one-way functions exist if and only if there exist two sets reducible to each other via injective length-increasing reductions but not isomorphic to each other. Kurtz, Mahaney and Royer [KMR89] show that relative to a random oracle, those two sets could be **NP**-complete. On the other hand, Hartmanis and Hemachandra [HH91] show that there exists a relativized world where no one-way functions exist but the isomorphism conjecture fails.

Kurtz, Mahaney and Royer [KMR88] show there exists some set such that all sets many-one equivalent to it are isomorphic to it. Fenner, Fortnow and Kurtz [FFK92] give a relativized world where all the **NP**-complete sets are isomorphic, i.e., the Berman-Hartmanis conjecture holds.

2.7 Simulating Randomness

Probabilistic computation has played a major role in complexity theory during the last decade. In Section 2.10 we will see how interactive proof system models use randomness as a powerful verification tool. In this section we will look at an opposite approach–to look at how we can reduce or eliminate randomness from some computational models.

We have seen many important results in this area over the past decade. Noam Nisan has played an important role in many of these results so in this section I would like to highlight one of his more general results:

Favorite Theorem 7 ([Nis92a]) *For any $r(n)$ and $s(n)$ there exists a pseudorandom generator that converts a random seed of length $O(s(n)\log r(n))$ to $r(n)$ bits that looks random to any algorithm using $s(n)$ space.*

Nisan's generator builds on universal hashing, a technique to generate pairwise independence (or close to it) without using many random bits. Nisan then builds a random string by recursively applying a specific universal hash function.

Theorem 7 has important applications to the problem of universal traversal sequences. Aleliunas, Karp, Lipton, Lovász and Rackoff [AKL$^+$79] show that for every n there exist polynomial in n length *universal traversal sequences* that traverse every connected undirected graph on n nodes. Constructing such sequences has remained an important open question. Istrail [Ist88] requires a difficult proof just to show a constructible sequence for cycles. Theorem 7 implies that $n^{O(\log n)}$ length universal traversal sequences can be constructed in $O(\log^2 n)$ space.

Nisan [Nis92b] extends the techniques of Theorem 7 to show that every language accepted in randomized logarithmic space like undirected graph connectivity can be accepted by a deterministic Turing machine running in polynomial time and $O(\log^2 n)$ space. Also building on the generator from Theorem 7, Nisan, Szemerédi and Wigderson [NSW92] show that undirected connectivity can be computed in $O(\log^{1.5} n)$ space.

Nisan and Zuckerman [NZ93] show how to simulate any randomized $s(n)$ space bounded Turing machine that uses $poly(s(n))$ random bits in deterministic space $s(n)$. Their proof uses a procedure that extracts randomness from a defective random source using a small additional number of truly random bits.

Suppose we had a probabilistic algorithm A that gave the correct answer with probability $2/3$ using r random coins. If we wanted to get this probability up to $1-2^{-k}$ we can use the standard trick of running A $O(k)$ times and taking majority vote. However, this method will take $O(nk)$ random coins.

Can we achieve the same result using fewer random coins? Impagliazzo and Zuckerman [IZ89] give a tight answer to this question. They show how to achieve the $1-2^{-k}$ error using only $O(n+k)$ random coins. Their proof uses a new idea of taking a random walk on an expander graph.

2.8 Counting Complexity

In 1979, Valiant [Val79a] looked at the question of computing the permanent of a matrix. Unlike the determinant where we knew polynomial-time algorithms, the permanent seemed a much more difficult problem to handle.

In order to capture the power of the permanent, Valiant developed a new function class **#P**. A function is in **#P** if there exists a nondeterministic polynomial-time Turing machine M such that $f(x)$ equals the number of accepting computations of $M(x)$. Valiant showed that the permanent is **#P**-complete. In future work, Valiant [Val79b] showed that several other natural counting questions are **#P**-complete.

Clearly, **#P** functions are hard for **NP**. But we knew little about their relationship to other complexity classes. In perhaps the best complexity result of the last decade, Toda showed a surprising relationship between the polynomial-time hierarchy and **#P** functions:

Favorite Theorem 8 ([Tod91]) *Every language in the polynomial-time hierarchy can be reduced in polynomial-time to a single query of a #P function.*

In other words, one can use counting to simulate a constant number of alternations.

Toda's proof uses several new and exciting ideas making this theorem one of the prettiest structural complexity results in the last decade. Toda actually proves two separate theorems. First he shows how to randomly reduce every language in the polynomial-time hierarchy to a $\oplus$**P** question, where $L \in \oplus\mathbf{P}$ if there exists a $\#\mathbf{P}$ function f where $x \in L$ if and only if $f(x)$ is odd. Then Toda shows how to reduce languages probabilistically reducible to $\oplus\mathbf{P}$ to a single $\#\mathbf{P}$ question.

Valiant and Vazirani proved the following extremely useful lemma:

Lemma 1. *[VV86] There exists a random reduction γ mapping* **CNF** *formulas to* **CNF** *formulas such that for all formulas ϕ of n variables*

1. *If ϕ is not satisfiable then $\gamma(\phi)$ is never satisfiable.*
2. *If ϕ is satisfiable then with probability at least $\frac{1}{4n}$, $\gamma(\phi)$ has* exactly *one satisfying assignment.*

Papadimitriou and Zachos [PZ83] show that $\oplus\mathbf{P}^{\oplus\mathbf{P}} = \oplus\mathbf{P}$, i.e. a $\oplus\mathbf{P}$ machine that asks arbitrary $\oplus\mathbf{P}$ questions to an "oracle".

Toda uses Lemma 1 in a novel way combined with the Papadimitriou and Zachos result to show how to probabilistically reduce every **NP** language to a $\oplus\mathbf{P}$ language. Toda then uses a nice induction again with some new tricks to show that every language in **PH** (the polynomial-time hierarchy) probabilistically reduces to a $\oplus\mathbf{P}$ set.

Toda then shows for any polynomial q, how to modify a $\#P$ function f to a new function g such that $f(x) \bmod 2 = g(x) \bmod 2^{q(|x|)}$ for all $x \in \Sigma^*$. Using this fact, Toda can combine the randomness and the $\oplus\mathbf{P}$ question into a single $\#\mathbf{P}$ question thus completing his proof.

Toda's theorem sparked renewed interest in counting complexity. Much of the work has centered on classes defined by counting functions. We will discuss this work in Section 2.9.

Toda's result also has implications in circuit complexity. Allender [All89] shows how to use Toda's proof to show that every constant depth circuit has an equivalent depth-three quasipolynomial-size threshold circuit. The class **ACC** consists of those circuits accepted by bounded depth circuits with **AND**, **OR** and $\mathbf{MOD}_q$ gates where q is a fixed integer not necessarily prime. Yao [Yao90] and Beigel and Tarui [BT91] extend the ideas of Theorem 8 to show that every language in **ACC** is recognized by depth-two circuits with a symmetric (independent of input order) gate at the root and quasipolynomial **AND** gates of polylog fan-in at the leaves.

Fenner, Fortnow and Kurtz [FFK94] developed the notion of **GapP**. The function class **GapP** consists of functions $f(x)$ where there exists a nondeterministic polynomial-time Turing machine M where $f(x)$ is equal to the number of accepting paths of $M(x)$ minus the number of rejecting paths of $M(x)$. Equivalently, **GapP** consists of the closure of $\#\mathbf{P}$ functions under subtraction. Fenner, Fortnow and Kurtz show that many of the $\#\mathbf{P}$ closure properties also hold for **GapP** and that looking at **GapP** functions simplified many counting complexity arguments.

Toda and Ogiwara [TO92] extend part of Toda's work to show that every function in $\mathbf{GapP}^{\mathbf{PH}}$ probabilistically reduces to a **GapP** function. Their result shows that in addition to $\oplus\mathbf{P}$, **PH** probabilistically reduces to many other counting classes such as **PP** (see Section 2.9).

Toda's result leads to one of the more intriguing open questions to arise in the last decade: Does $\mathbf{P}^{\#\mathbf{P}} = \mathbf{PSPACE}$? In other words, can counting simulate polynomial alternations?

2.9 Counting Classes

As described in Section 2.8, counting complexity played a major role in complexity theory in the past decade. Instead of asking whether an **NP** question has a solution, we now ask how many. In Section 2.8 we looked at the complexity of functions defined this way. In this section we will look at complexity classes based on counting functions.

Gill [Gil77] in his seminal paper on probabilistic complexity classes defined the basic probabilistic classes that we look at today including the class **PP** - Probabilistic Polynomial-time. The class **PP** consists of those languages accepted by probabilistic polynomial-time Turing machines that accept with probability greater than a half.

This class also plays an important role in counting complexity. A language L is in **PP** if there is some nondeterministic polynomial-time Turing machine M such that x is in L if and only if the accepting paths of $M(x)$ outnumber the rejecting paths or equivalently when some **GapP** function $f(x)$ is greater than zero.

Such a class contains **NP**. By binary search, one can show that $\mathbf{P}^{\mathbf{PP}} = \mathbf{P}^{\#\mathbf{P}}$. Thus by Theorem 8, we have that $\mathbf{PH} \subseteq \mathbf{P}^{\mathbf{PP}}$ and thus if $\mathbf{PP} \subseteq \mathbf{PH}$ then the polynomial-time hierarchy collapses.

Clearly **PP** is closed under complementation but surprisingly there is no easy way to show that **PP** is closed under intersection. That question remained open until the '90s when Beigel, Reingold and Spielman showed us that **PP** does indeed have this closure property.

Favorite Theorem 9 ([BRS94]) **PP** *is closed under intersection.*

The proof of Theorem 9 really makes use of the connection between **GapP** functions and low-degree polynomials. Beigel, Reingold and Spielman make use of the fact that there exist rational functions that very closely approximate the absolute value function and use it to create a rational function of **GapP** functions that take on a positive value if and only if both **GapP** functions are positive. They then show how to test for this condition with a **PP** predicate.

Fortnow and Reingold [FR91] extend the techniques of Beigel, Reingold and Spielman to show that **PP** is closed under truth-table reductions. Beigel [Bei92] created a very useful oracle relative to which $\mathbf{P}^{\mathbf{NP}} \not\subseteq \mathbf{PP}$ and thus **PP** is not closed under Turing reductions.

We have seen many other counting classes arise in complexity theory recently. We unfortunately only have room to review a few of them.

The class $\mathbf{Mod}_k\mathbf{P}$ consists of those languages L where there exists a $\#\mathbf{P}$ function f such that $x \in L$ if and only if $f(x) \bmod k = 1$. Papadimitriou and Zachos [PZ83] look at the special case of $\oplus\mathbf{P} = \mathbf{Mod}_2\mathbf{P}$ and show that $\oplus\mathbf{P}^{\oplus\mathbf{P}} =$

$\oplus$**P**. Beigel and Gill [BG92] and Hertrampf [Her90] extend this result for $\mathbf{Mod}_k\mathbf{P}$ where k is prime. We have also already seen the importance that $\oplus$**P** plays in the proof of Theorem 8.

Fenner, Fortnow and Kurtz [FFK94] formalize a notion of complexity classes defined via **GapP** functions (see Section 2.8). They define a class **SPP** where a language L lies in **SPP** if there exists a **GapP** function f such that $f(x) = \chi_L(x)$. They show that **SPP** is contained in every reasonable Gap-definable class.

2.10 Interactive Proof Systems

No review of complexity theory in the past decade could be complete without mentioning interactive proof systems. In order to keep this section manageable we will concentrate only on the complexity theoretical aspects of interactive proof systems. I recognize the importance of many interesting zero-knowledge, program testing and approximation results that arise from the theory of interactive proof systems but will leave most of the discussion of them to other surveys.

We highlight the last of the truly great results in interactive proof systems, a paper by Arora, Lund, Motwani, Sudan and Szegedy that characterizes **NP** by a simple verification procedure:

Favorite Theorem 10 ([ALM$^+$92]) *Every language in* **NP** *has a probabilistically checkable proof system where the verifier uses only $O(\log n)$ random coins and asks only a constant number of queries to the proof.*

In order to appreciate Theorem 10, we first give a history of the complexity of interactive proof systems.

Goldwasser, Micali and Rackoff [GMR89] and Babai [Bab85, BM88] independently developed *interactive proof systems*. The model has an arbitrarily powerful prover that tries to convince an untrusting polynomial-time probabilistic verifier about whether some string is in a language. Goldwasser and Sipser [GS89] show that the power of the model does not depend on whether or not the prover can see the verifier's coins.

A *bounded-round* interactive proof system only allows a fixed number of polynomial-length messages between the prover and the verifier. Babai [Bab85] shows that every bounded-round proof system has an equivalent system consisting of a single message from the verifier followed by a single message from the prover. Boppana, Håstad and Zachos [BHZ87] show that if every **coNP** language has bounded-round interactive proof systems then the polynomial-time hierarchy collapses to Σ_2^p. Goldreich, Micali and Wigderson [GMW91] show that graph nonisomorphism has a bounded-round interactive proof system. As an immediate corollary, we get that graph isomorphism is not **NP**-complete unless the polynomial-time hierarchy collapses.

Feldman [Fel86] showed that in interactive proof systems we can assume the prover runs in polynomial-space and thus every language accepted by interactive proof systems lies in **PSPACE**. Lund, Fortnow, Karloff and Nisan [LFKN92] show that every language in **PH** has an (unbounded-round) interactive proof system. Shamir [Sha92] extends the techniques of Lund, Fortnow, Karloff and Nisan to show that, every language in **PSPACE** has an interactive proof system. The proof uses the low-degree structure of some **#P** and **PSPACE**-complete problems. These results are some of the very few known that do not relativize: Fortnow and Sipser [FS88] exhibit an oracle relative to which some **coNP** language does not have an interactive proof system.

Ben-Or, Goldwasser, Kilian and Wigderson [BGKW88] developed a *multiple-prover* interactive proof system where many separated provers try to convince the verifier that a string lies in a language. Fortnow, Rompel and Sipser [FRS94] showed that every language accepted by such proof systems lies in **NEXP**. Babai, Fortnow and Lund [BFL91] showed that every language in **NEXP** has such a proof system. Babai, Fortnow and Lund's proof uses a relativizable form of the Lund, Fortnow, Karloff and Nisan protocol combined with a multilinearity test. Feige and Lovász [FL92] show that very language in **NEXP** in fact has a two-prover one-round proof system.

Arora and Safra [AS92] define a *probabilistically checkable proof system* where the prover must write down a perhaps exponentially long proof system that the polynomial-time probabilistic verifier can check using random access to the proof. Fortnow, Rompel and Sipser [FRS94] show the equivalence between the multiple-prover interactive proof system model and an arbitrary probabilistically checkable proof. Arora, Lund, Motwani, Sudan and Szegedy, building on ideas of Arora and Safra [AS92], proved Theorem 10.

Arora, Lund, Motwani, Sudan and Szegedy's result also has implications for approximation problems. Papaditimitriou and Yannakakis [PY91] define a class **MAXSNP** of **NP** optimization problems such as finding an assignment that maximizes the number of clauses made true in a formula. A corollary of the result of Arora, Lund, Motwani, Sudan and Szegedy shows that for every **MAXSNP**-hard language L, there is some constant $\epsilon > 0$ such that one could not approximate problems in L within a $1 + \epsilon$ factor unless $\mathbf{P} = \mathbf{NP}$.

3 Conclusions

One can draw several interesting conclusions about the theorems on the list:

- No theorem stands alone. Every theorem chosen either has a long line of results leading up to it and/or has several results that use the theorem or technique in an important way.
- No one person has dominated complexity theory. Though a few strong theorists do stand out from the last decade, no single person has proven more than one of the theorems I have chosen.
- No single country dominates the list. The twenty authors of these ten results represent no fewer than nine separate countries.

Here's to hoping that computational complexity theory can achieve as many great results in the next decade as it has in this past one.

Acknowledgments

I would like to thank Satyanarayana V. Lokam, Dieter Van Melkebeek and Sophie Laplante for their comments and help on this paper.

References

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

[Ajt83] M. Ajtai. Σ_1^1-formulae on finite structures. *Annals 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.

[All89] E. Allender. A note on the power of threshold circuits. In *Proceedings of the 30th IEEE Symposium on Foundations of Computer Science*, pages 580–584. IEEE, New York, 1989.

[ALM+92] S. Arora, C. Lund, R. Motwani, M. Sudan, and M. Szegedy. Proof verification and hardness of approximation problems. In *Proceedings of the 33rd IEEE Symposium on Foundations of Computer Science*, pages 14–23. IEEE, New York, 1992.

[AS92] S. Arora and S. Safra. Probabilistic checking of proofs: A new characterization of NP. In *Proceedings of the 33rd IEEE Symposium on Foundations of Computer Science*, pages 2–13. IEEE, New York, 1992.

[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.

[Bar89] D. Barrington. Bounded-width polynomial-size branching programs recognize exactly those languages in NC^1. *Journal of Computer and System Sciences*, 38(1):150–164, 1989.

[BCD+89a] A. Borodin, S. Cook, P. Dymond, L. Ruzzo, and M. Tompa. Erratum: Two applications of inductive counting for complementation problems. *SIAM Journal on Computing*, 18(6):1283, 1989.

[BCD+89b] A. Borodin, S. Cook, P. Dymond, L. Ruzzo, and M. Tompa. Two applications of inductive counting for complementation problems. *SIAM Journal on Computing*, 18(3):559–578, 1989. See also Erratum [BCD+89a].

[BCS92] D. Bovet, P. Crescenzi, and R. Silvestri. A uniform approach to define complexity classes. *Theoretical Computer Science*, 104:263–283, 1992.

[Bei92] R. Beigel. Perceptrons, PP and the polynomial hierarchy. In *Proceedings of the 7th IEEE Structure in Complexity Theory Conference*, pages 14–19. IEEE, New York, 1992.

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

[BG92] R. Beigel and J. Gill. Counting classes: Thresholds, parity, mods, and fewness. *Theoretical Computer Science*, 103:3–23, 1992.

[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.

[BH77] L. Berman and J. Hartmanis. On isomorphism and density of NP and other complete sets. *SIAM Journal on Computing*, 1:305–322, 1977.

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

[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.

[BRS94] R. Beigel, N. Reingold, and D. Spielman. PP is closed under intersection. *Journal of Computer and System Sciences*, 1994. To appear. Paper also appeared in Proceedings of 23rd STOC conference, 1991, pages 1-9.

[BT91] R. Beigel and J. Tarui. On ACC. In *Proceedings of the 32nd IEEE Symposium on Foundations of Computer Science*, pages 783–792. IEEE, 1991.

[CF91] J. Cai and M. Furst. PSPACE survives constant-width bottlenecks. *International Journal of Foundations of Computer Science*, 2:67–76, 1991.

[CKS81] A. Chandra, D. Kozen, and L. Stockmeyer. Alternation. *Journal of the ACM*, 28(1):114–133, 1981.

[Edm65] J. Edmonds. Paths, trees and flowers. *Canadian Journal of Mathematics*, 17:449–467, 1965.

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

[FFK92] S. Fenner, L. Fortnow, and S. Kurtz. The isomorphism conjecture holds relative to an oracle. In *Proceedings of the 33rd IEEE Symposium on Foundations of Computer Science*, pages 30–39. IEEE, New York, 1992.

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

[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.

[FR91] L. Fortnow and N. Reingold. PP is closed under truth-table reductions. In *Proceedings of the 6th IEEE Structure in Complexity Theory Conference*, pages 13–15. IEEE, New York, 1991.

[FRS94] L. Fortnow, J. Rompel, and M. Sipser. On the power of multi-prover interactive protocols. *Theoretical Computer Science A*, 1994. To appear.

[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.

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

[GKL93] O. Goldreich, H. Krawczyk, and M. Luby. On the existence of pseudorandom 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.

[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.

[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.

[Hås90] J. Håstad. Pseudo-random generators under uniform assumptions. In *Proceedings of the 22nd ACM Symposium on the Theory of Computing*, pages 395–404. ACM, New York, 1990.

[Her90] U. Hertrampf. Relations among MOD classes (note). *Theoretical Computer Science*, 74:325–328, 1990.

[HH91] J. Hartmanis and L. Hemachandra. One-way functions and the nonisomorphism of NP-complete sets. *Theoretical Computer Science*, 81(1):155–163, 1991.

[HL94] S. Homer and L. Longpré. On reductions of NP sets to sparse sets. *Journal of Computer and System Sciences*, 48(2):324–336, 1994.

[HLS$^+$93] U. Hertrampf, C. Lautemann, T. Schwentick, H. Vollmer, and K. Wagner. On the power of polynomial time bit-reductions. In *Proceedings of the 8th IEEE Structure in Complexity Theory Conference*, pages 200–207. IEEE, New York, 1993.

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

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

[ILL89] R. Impagliazzo, L. Levin, and M. Luby. Pseudo-random number generation from one-way functions. In *Proceedings of the 21st ACM Symposium on the Theory of Computing*, pages 12–24. ACM, New York, 1989.

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

[IR89] R. Impagliazzo and S. Rudich. Limits on the provable consequences of one-way permutations. In *Proceedings of the 21st ACM Symposium on the Theory of Computing*, pages 44–61. ACM, New York, 1989.

[Ist88] S. Istrail. Polynomial universal traversing sequences for cycles are constructible. In *Proceedings of the 20th ACM Symposium on the Theory of Computing*, pages 491–503. ACM, New York, 1988.

[IZ89] R. Impagliazzo and D. Zuckerman. How to recycle random bits. In *Proceedings of the 30th IEEE Symposium on Foundations of Computer Science*, pages 248–253. IEEE, New York, 1989.

[JY85] D. Joseph and P. Young. Some remakrs on witness functions for poynomial reducibilities in NP. *Theoretical Computer Science*, 39:225–237, 1985.

[KL82] R. Karp and R. Lipton. Turing machines that take advice. *L'Enseignement Mathematique*, 28:191–209, 1982.

[KLD87] K. Ko, T. Long, and D. Du. A note on one-way functions and polynomial-time isomorphisms. *Theoretical Computer Science*, 47:263–276, 1987.

[KMR88] S. Kurtz, S. Mahaney, and J. Royer. Collapsing degrees. *Journal of Computer and System Sciences*, 37(2):247–268, 1988.

[KMR89] S. Kurtz, S. Mahaney, and J. Royer. The isomorphism conjecture fails relative to a random oracle. In *Proceedings of the 21st ACM Symposium on the Theory of Computing*, pages 157–166. ACM, New York, 1989.

[KW90] M. Karchmer and A. Wigderson. Monotone circuits for connectivity require super-logarithmic depth. *SIAM Journal on Discrete Mathematics*, 3:255–265, 1990.

[Lan63] P. Landweber. Three theorems on phase structure grammars of type 1. *Information and Control*, 6(2):131–136, 1963.

[Lev87] L. Levin. One-way functions and pseudo-random generators. *Combinatorica*, 7:357–363, 1987.

[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.

[LMN93] N. Linial, Y. Mansour, and N. Nisan. Constant depth circuits, fourier transform, and learnability. *Journal of the ACM*, 40(3):607–620, 1993.

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

[Nis92a] N. Nisan. Pseudorandom generators for space-bounded computation. *Combinatorica*, 12(4):449–461, 1992.

[Nis92b] N. Nisan. RL is contained in SC. In *Proceedings of the 24th ACM Symposium on the Theory of Computing*, pages 619–623. ACM, New York, 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.

[NZ93] N. Nisan and D. Zuckerman. More deterministic simulation in logspace. In *Proceedings of the 25th ACM Symposium on the Theory of Computing*, pages 235–244. ACM, New York, 1993.

[OW91] M. Ogiwara and O. Watanabe. On polynomial-time bounded truth-table reducibility of NP sets to sparse sets. *SIAM Journal on Computing*, 20(3):471–483, 1991.

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

[PZ83] C. Papadimitriou and S. Zachos. Two remarks on the power of counting. In *Proceedings of the 6th GI Conference on Theoretical Computer Science*, volume 145 of *Lecture Notes in Computer Science*, pages 269–276. Springer, Berlin, 1983.

[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. *Dokl. Akad. 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 Math. dokl.*, 31:485–493, 1985.

[Raz87] A. Razborov. Lower bounds on the size of bounded depth circuits over a complete basis with logical addition. *Mathematical Notes of the Academy of Sciences of the USSR*, 41(4):333–338, 1987.

[Raz89] A. Razborov. On the method of approximations. In *Proceedings of the 21st ACM Symposium on the Theory of Computing*, pages 167–176. ACM, New York, 1989.

[Raz90] A. Razborov. Applications of matrix methods to the theory of lower bounds in computational complexity. *Combinatorica*, 10(1):81–93, 1990.

[Raz94] A. Razborov. Bounded Arithmetic and lower bounds in Boolean complexity. *Feasible Mathematics II*, 1994. To appear.

[RR94] A. Razborov and S. Rudich. Natural proofs. In *Proceedings of the 26th ACM Symposium on the Theory of Computing*, pages 204–213. ACM, New York, 1994.

[RW92] R. Raz and A. Wigderson. Monotone circuits for matching require linear depth. *Journal of the ACM*, 39(3):736–744, 1992.

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

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

[Smo87] R. Smolensky. Algebraic methods in the theory of lower bounds for Boolean circuit complexity. In *Proceedings of the 19th ACM Symposium on the Theory of Computing*, pages 77–82. ACM, New York, 1987.

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

[Tar88] É. Tardos. The gap between monotone and nonmonotone circuit complexity is exponential. *Combinatorica*, 8:141–142, 1988.

[Tar93] J. Tarui. Probabilistic polynomials, AC^0 functions and the polynomial-time hierarchy. *Theoretical Computer Science A*, 113:167–183, 1993.

[TO92] S. Toda and M. Ogiwara. Counting classes are at least as hard as the polynomial-time hierarchy. *SIAM Journal on Computing*, 21(2):316–328, 1992.

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

[Val79a] L. Valiant. The complexity of computing the permanent. *Theoretical Computer Science*, 8:189–201, 1979.

[Val79b] L. Valiant. The complexity of reliability and enumeration problems. *SIAM Journal on Computing*, 8:410–421, 1979.

[VV86] L. Valiant and V. Vazirani. NP is as easy as detecting unique solutions. *Theoretical Computer Science*, 47:85–93, 1986.

[Yao82] A. Yao. Theory and applications of trapdoor functions. In *Proceedings of the 23rd IEEE Symposium on Foundations of Computer Science*, pages 80–91. IEEE, New York, 1982.

[Yao83] A. Yao. Lower bounds by probabilistic arguments. In *Proceedings of the 24th IEEE Symposium on Foundations of Computer Science*, pages 420–428. IEEE, New York, 1983.

[Yao85] A. Yao. Separating the polynomial-time hierarchy by oracles. In *Proceedings of the 26th IEEE Symposium on Foundations of Computer Science*, pages 1–10. IEEE, New York, 1985.

[Yao90] A. Yao. On ACC and threshold circuits. In *Proceedings of the 31st IEEE Symposium on Foundations of Computer Science*, pages 619–631. IEEE, New York, 1990.