# Comparing Notions of Full Derandomization

Lance Fortnow*
NEC Research Institute

## Abstract

Most of the hypotheses of full derandomization fall into two sets of equivalent statements: Those equivalent to the existence of efficient pseudorandom generators and those equivalent to approximating the accepting probability of a circuit. We give the first relativized world where these sets of equivalent statements are not equivalent to each other.

## 1 Introduction

Impagliazzo and Wigderson [IW97] show that if there exists a language **E** that requires $2^{\Omega(n)}$ size circuits then **P** = **BPP**. Andreev, Clementi and Rolim [ACR98] show that if efficient hitting set generators exist then **P** = **BPP**. A careful look reveals that not only do these papers have the same conclusion but in fact their hypotheses imply each other.

Both of their assumptions produce a pseudorandom generator that succeeds against all small circuits. It might be easier to allow the pseudorandom generator access to the circuit it tries to fool. The existence of generators with access to the circuit is equivalent to many other derandomization hypotheses such as efficiently approximating the accepting probability of a circuit, Promise-**RP** is easy, Promise-**BPP** is easy, efficiently finding accepting inputs of circuits that accept many inputs, and the equivalence of the classes **AP** and **APP** defined by Kabanets, Rackoff and Cook [KRC00].

We give some evidence that allowing access to the circuit does help in derandomization. We produce the first relativized world where efficient pseudorandom generators do not exist but we still can approximate the accepting probability of a circuit. In fact our oracle makes the Impagliazzo-Wigderson assumption fail in a strong way—relative to it the class **E** has linear-size circuits.

Relativization plays an important role in derandomization results. Relativization helps explain why despite a large collection of recent derandomization results we still lack any nontrivial unconditional derandomization of **BPP** or **ZPP**. Nearly all of the derandomization techniques, including all of the results mentioned in this paper, relativize and there exist relativized worlds where **BPP** = **NEXP** [Hel86] and **ZPP** = **EXP** [Hel84, Kur85]. We will need truly different and revolutionary new techniques to get unconditional derandomization results or to show efficiently approximating the accepting probability of a circuit implies efficient pseudorandom generators.

In Section 2 we give an overview of the various notions of full derandomization and how they relate. In Section 3 we give a proof of our main result of a relativized world that distinguishes the two important equivalent classes of full derandomization hypotheses. In Section 4 we describe the role of relativization in derandomization results.

## 2 Full Derandomization

In this section we discuss the various notions of full derandomization and how they are known to relate. This section is meant as a survey of these notions. If no citations are given, the results should be considered folklore. Proofs, when given, are for completeness. Note that all of the results mentioned in this section hold relative to any oracle.

Figure 1 shows the hypotheses that we consider in this paper. To make a long story short, each of these hypotheses imply all of the ones below it and there are relativized worlds that prevent all of the reverse implications. The statements listed in each box are equivalent. Current belief has Hypothesis I false and the rest true.

Let us consider the hypotheses in detail. Hypoth-

*Address: 4 Independence Way, Princeton, NJ 08540 USA. Email: `fortnow@research.nj.nec.com`. `http://www.neci.nj.nec.com/homepages/fortnow`.

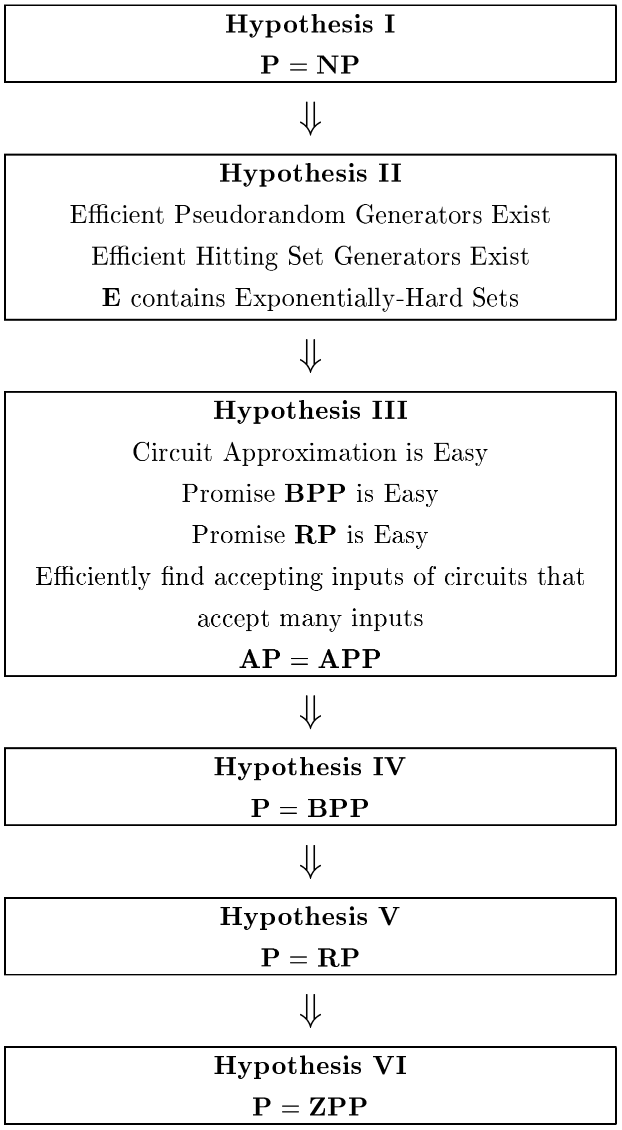
Hypothesis I
P = NP
Hypothesis II
Efficient Pseudorandom Generators Exist
Efficient Hitting Set Generators Exist
E contains Exponentially-Hard Sets
Hypothesis III
Circuit Approximation is Easy
Promise BPP is Easy
Promise RP is Easy
Efficiently find accepting inputs of circuits that accept many inputs
AP = APP
Hypothesis IV
P = BPP
Hypothesis V
P = RP
Hypothesis VI
P = ZPP

Figure 1: Derandomization Hypotheses

esis I, **P** = **NP**, has been studied in great detail (see the book of Garey and Johnson [GJ79] for example). **P** = **NP** is not a derandomization hypothesis per se but we include it for comparisons to the other hypotheses.

To understand Hypothesis II let us give formal definitions of efficient pseudorandom generators and efficient hitting set generators.

**Definition 2.1** An *efficient pseudorandom generator* is a function $G : \Sigma^{k \log n} \to \Sigma^n$ (for some fixed $k$) computable in time polynomial in $n$ such that for all circuits $C$ of size $n$

$$|\Pr_{r \in \Sigma^n}(C(r) = 1) - \Pr_{y \in \Sigma^{k \log n}}(C(G(y)))| \leq \frac{1}{n}.$$

**Definition 2.2** A *hitting set generator* is a function $H$ mapping $1^n$ to a polynomially large set of strings of length $n$ such that if $C$ is a circuit with $n$ input bits and size at most $n$ and

$$\Pr_{y \in \{0,1\}^n}(C(y) = 1) \geq \frac{1}{n},$$

then $C$ accepts some string in $H(1^n)$. An *efficient hitting set generator* is a hitting set generator $H$ that runs in time polynomial in $n$.

The existence of hitting set generators follows from picking the range of $H$ at random and showing that with high probability it fulfills the properties of Definition 2.2. By the definitions, one can see that a minimum hitting set generator is computable in the polynomial-time hierarchy so if **P** = **NP** then efficient hitting set generators exist. No relativizable proof of the converse can hold: Relative to a random oracle, hitting set generators exist but **P** $\neq$ **NP** [BG81].

**Theorem 2.3** *The following are equivalent:*

1. *Efficient pseudorandom generators exist.*
2. *Efficient hitting set generators exist.*
3. *There is a $L$ in* **E** = **DTIME**$(2^{O(n)})$ *such that $L$ requires circuits of size $2^{\epsilon n}$ for some $\epsilon > 0$ and all but finitely many $n$.*

**Proof Sketch:**

$1 \Rightarrow 2$ The range of a pseudorandom generator is also a hitting set generator.

$2 \Rightarrow 3$ [ISW99]: Suppose we have a efficient hitting set generator $H$ on $k(n) = c \log n$ inputs. Consider prefixes of the range of $H$ of size $k(n) + 1$. This is in **E** by trying all inputs but if it has a circuit of size $2^{(k(n)+1)/2c} \ll n$ we can create a circuit of size $n$ that accepts many inputs but misses the range of $H$.

$3 \Rightarrow 1$ This was proven by Impagliazzo and Wigderson [IW97] in their breakthrough paper.

Now let us consider Hypothesis III.

**Definition 2.4** We say *Circuit approximation is easy* if there exists a function $f(C, \epsilon)$ computable in time polynomial in $|C|$ and $1/\epsilon$ such that

$$|\Pr(C(x) = 1) - f(C, \epsilon)| \leq \epsilon.$$

**Definition 2.5** We say *Promise-***BPP** *is easy* if for every probabilistic polynomial-time machine $M$ there exists a language $L$ in **P** such that for all $x$,

- If $M(x)$ accepts with probability at least $2/3$ then $x$ is in $L$, and
- If $M(x)$ accepts with probability at most $1/3$ then $x$ is not in $L$.

We put no restrictions on $L$ when the probability of acceptance of $M$ is between $1/3$ and $2/3$.

We can similarly define when Promise-**RP** is easy.

The classes **AP** and **APP** are defined by Kabanets, Rackoff and Cook [KRC00].

**Definition 2.6**

- The class **APP** consists of those real-valued functions $g : \{0,1\}^* \to [0,1]$ such that there exists a probabilistic Turing machine $M$ running in time polynomial in $|x|$ and $1/\epsilon$ such that

$$\Pr(|M(x,\epsilon) - g(x)| \leq \epsilon) \geq 3/4.$$

- The class **AP** consists of those real-valued functions $g : \{0,1\}^* \to [0,1]$ such that there exists a deterministic Turing machine $M$ running in time polynomial in $|x|$ and $1/\epsilon$ such that

$$|M(x,\epsilon) - g(x)| \leq \epsilon.$$

We can now formally state the equivalences of Hypothesis III.

**Theorem 2.7** *The following are equivalent:*

1. *Circuit approximation is easy.*
2. *Promise* **BPP** *is easy.*
3. *Promise* **RP** *is easy.*
4. *There exists a polynomial-time computable function $f$ such that for all circuits $C$ that accepts at least half of its inputs, $C$ accepts $f(C)$.*
5. **APP** $=$ **AP**.

**Proof Sketch of Theorem 2.7:**

$1 \Rightarrow 4$ Suppose we have a circuit $C$ that accepts half its inputs. Consider $C_0$ with the first bit set to zero and $C_1$ with the first bit set to one. Approximate the accepting probability of $C_0$ and $C_1$ to within $1/n^2$. Pick the one that approximates to a larger value and repeat until one finds an accepting input.

$4 \Rightarrow 3$ We can deterministically convert an **RP** machine $M$ on an input $x$ to a circuit polynomial in $|x|$ whose inputs are the random bits used by $M(x)$.

$3 \Rightarrow 2$ Buhrman and Fortnow [BF99] show how Lautemann's proof that **BPP** is in $\Sigma_2^p$ [Lau83] actually gives promise-**BPP** in $\mathbf{RP}^{\text{Promise-}\mathbf{RP}}$.

$2 \Rightarrow 1$ Consider the probabilistic machine that on input $M(C, 1^j, 1^k)$ picks $j$ random inputs to $C$ and accepts if $C$ accepts at least $k$ of them. By applying standard Chernoff arguments, one can show that when the probability that $C$ accepts is greater than $(k+O(\sqrt{j}))/j$ then $M$ will accept with high probability and when the probability that $C$ accepts is at most $(k-O(\sqrt{j}))/j$ then $M$ will reject with high probability. Picking $j = O(1/\epsilon^2)$, we output $k/j$ for $k$ the least value such that our deterministic algorithm for promise-**BPP** algorithm for $M(C, 1^j, 1^k)$ guarantees that it does not reject with high probability.

$4 \Leftrightarrow 5$ Straightforward.

One can use an efficient pseudorandom generator in straightforward way to do circuit approximation. This shows that Hypothesis II implies Hypothesis III. Andreev, Clementi and Rolim [ACR98] (see also [ACRT99]) directly show that the existence of efficient hitting set generators imply that Promise-**BPP** is easy.

Kabanets and Cai [KC00] show that if one can compute in polynomial time from the truth-table of a function, the size of its minimum circuit then Hypotheses II and III are equivalent.

The main result of our paper, discussed in Section 3, shows that no relativizable proof can show that Hypotheses II and III are equivalent.

The implications of Hypothesis III to IV to V to VI are all immediate from definitions. The relativized world where IV holds and III fails follows from many published oracles. In particular consider a generic oracle built on top of TQBF: Impagliazzo and Naor [IN88] show that **P** $=$ **BPP** relative to this oracle but one easily gets that Hypothesis III fails.

Muchnik and Vereshchagin [MV96] give relativized worlds where IV fails and V holds and where V fails and VI holds.

Baker, Gill and Solovay [BGS75] give an oracle where all the hypotheses hold. Heller [Hel84] and Kurtz [Kur85] give a relativized world where **ZPP** $=$ **EXP** and all of the hypotheses fail in a strong way.

## 3 Main Result

In this section we give a relativized world where Hypothesis II fails but Hypothesis III holds. We need only show **E** has size $2^{o(n)}$ circuits for infinitely many

$n$ to get II to fail but in fact we will get something considerably stronger.

**Theorem 3.1** *There exists a oracle $A$ such that $\mathbf{E}^A$ has linear size circuits and we can efficiently (relative to A) find accepting inputs of relativized circuits that accept many inputs.*

**Corollary 3.2** *There exists a relativized world where circuit approximation is easy but efficient pseudorandom generators do not exist.*

One can think of the difference between circuit approximation and efficient pseudorandom generators as a one of order of quantifiers. A pseudorandom generator must fool all circuits of a certain size. A circuit approximator needs only approximate a specific circuit given as input.

Our proof will encode an **EXP**-complete language at a very hard to find place that can be pointed to with advice. This will have the effect of making a pseudorandom generator fail on hard to find circuits. We will also encode in these same hard to find places enough information so that an algorithm given a circuit that accepts many inputs, can use the circuit to find this information and use it to find an accepting input.

**Proof of Theorem 3.1:**

We will construct our oracle $A$ in stages. In stage $m$ of the construction we will only add strings to the oracle of the form $\langle m, y_m, \ldots\rangle$ for some fixed $y_m$ of length $5m$ though each stage will add infinitely many strings to the oracle.

Let $K^A$ be a relativizable $\mathbf{DTIME}^A(2^n)$ machine that accepts a linear-time complete set for $\mathbf{E}^A$. We will guarantee in the construction that for all $x$,

$$x \in L(K^A) \Leftrightarrow \langle |x|, y_{|x|}, x, 1\rangle \in A$$

The $y_m$ will serve as the advice for the linear-size circuit accepting $L(K^A)$.

We also guarantee that for every circuit $C^A$ that accepts half of its inputs, there will be a polynomial-time in $A$ procedure that given $C$ finds an accepted input.

We define the *key* of an oracle query $\langle m, y, \ldots\rangle$ to be $y$, the second element of the tuple.

We give the construction of $A$ in Figure 2.

By construction $K^A(x)$ is unaffected by any strings added in stage $m = |x|$ or later. So by using $y_m$ as advice we can determine whether $K^A(x)$ accepts by querying $\langle m, y_m, x, 1\rangle$.

Let $C^A$ be a circuit that accepts at least half its inputs. We will give a polynomial-time in $A$ algorithm for finding an accepted input. Our algorithm will have $y_1$ hardwired.

Our algorithm will first either find an accepted input or find a $y_j$ for $j > 1$. We repeat this procedure getting larger $j$'s each time until we find an accepted input. Our algorithm must halt with some $j \leq |C|$ since $C^A(x)$ cannot query any string with a key of $y_m$ for $m > |C|$.

Let $A_m$ represent the state of the oracle $A$ after stage $m$. Note that $C$ cannot query any string $\langle m, y_m, C', \ldots\rangle$ for $|C'| \geq |C|$. The circuit $C^{A_m}$ queries exactly the same strings as when we processed $C$ in stage $m$.

Let $k = 1$. We assume the algorithm knows $y_k$. We repeat the following until we have found an accepted input.

Look at $\langle k, y_k, C, \ldots, 2\rangle$ and see whether we have encoded an input $x$ or a set of keys $S$.

If we have encoded an input $x$ then simulate $C^A(x)$. If it accepts we are done. If not then we have $C^{A_k}(x)$ accepts but $C^A(x)$ rejects. So $C^{A_k}(x)$ must have queried some string $\langle j, y_j, \ldots\rangle \in A - A_k$ with $j > k$. The algorithm now knows $y_j$ and we then repeat the procedure for $k = j$.

If we have encoded a set $S$, search for a string $y_j$ in $S$ such that $j > k$ and $\langle j, y_j, 0\rangle$ is in $A$. Let $k = j$ and repeat the procedure.

If we have encoded a set $S$ of strings then $C^{A_k}$ rejects on all inputs but $C^A$ accepts on at least half of the inputs. On each of the inputs $x$ such that $C^A(x)$ accepts, $C^{A_k}(x)$ must have queried some string with a key of $y_j$ with $k < j \leq |C|$. For one of these $j$, $y_j$ must be a key on strings queried on at least a $\frac{1}{2|C|}$ fraction of inputs and thus $y_j \in S$. $\square$

## 4 Relativization and Derandomization

Derandomization results in the past years have made considerable us of combinatorics and algebra which might lead one to believe that many of the proofs do not relativize. However most results on derandomization do relativize including *every* result mentioned in Section 2. In fact Klivans and van Melkebeek [KvM99] show that the Impagliazzo-Wigderson [IW97] result relativizes in a very strong way.

**Theorem 4.1 (Klivans-van Melkebeek)** *Let $\mathcal{A}$ be a class of oracles and $B$ an oracle. If there is a boolean function $f$ in $\mathbf{E}^{\mathcal{A}}$ such that no circuit family of size $2^{o(n)}$ with oracle gates for $B$ can compute $f$ then there exists an efficient pseudorandom generator computable with an oracle for $\mathcal{A}$ secure against circuits with oracle gates for $B$.*

Initially set $A = \emptyset$.

Stage $m$:

Pick a $y_m$ of length $5m$ such that for all $x$ of length at most $m$, $K^A(x)$ does not query a string with a key of $y_m$. By a simple counting argument such a $y_m$ must exist. Put $\langle m, y_m, 0\rangle$ in $A$.

For each $x$ of length $m$ put $\langle m, y_m, x, 1\rangle$ in $A$ if $K^A(x)$ accepts.

For every circuit $C$ in increasing order of circuit size do the following:

If $C^A$ accepts some string then pick such a string $x$ and encode $x$ at $\langle m, y_m, C, \ldots, 2\rangle$, i.e., put $\langle m, y_m, C, i, 2\rangle$ in $A$ if the $i$th bit of $x$ is one.

If $C^A$ does not accept any string then consider the set $S$ of keys $v$ such that on at least a $\frac{1}{2|C|}$ fraction of the inputs $x$, $C^A(x)$ queries some string with a key of $v$. Note that $|S| \leq 2|C|^2$. Encode $S$ at $\langle m, y_m, C, \ldots, 2\rangle$.

This ends stage $m$.

Figure 2: Construction of $A$

Only very few results in the area of derandomization do not relativize. Babai, Fortnow, Nisan and Wigderson [BFNW93] give a nonrelativizing proof of the following.

**Theorem 4.2 (BFNW)** *If* **BPP** *is not infinitely often in subexponential time then* **EXP** = **MA**.

However one should not view Theorem 4.2 as a nonrelativizing derandomization result but as a combination of a relativizing derandomization result and a nonrelativizing result based on interactive proofs. Indeed Babai, Fortnow, Nisan and Wigderson prove Theorem 4.2 by giving a *relativizing* proof of Theorem 4.3.

**Theorem 4.3 (BFNW)** *If* **BPP** *is not infinitely often in subexponential time then* **EXP** *has polynomial-size circuits.*

Theorem 4.2 follows from Theorem 4.3 and the following result based on an observation of Nisan [LFKN92] and the proof of Babai, Fortnow and Lund [BFL91] showing that **NEXP** has multiprover interactive proof systems.

**Theorem 4.4 (Nisan,Babai-Fortnow-Lund)** *If* **EXP** *has polynomial-size circuits then* **EXP** = **MA**.

Results on interactive proofs have so far given the best examples of nonrelativizing results (see the survey of Fortnow [For94]). It should come as no surprise then that applications of these results to derandomization do not relativize either.

Impagliazzo, Kabanets and Wigderson [IKW01] use derandomization techniques to prove the following result.

**Theorem 4.5 (IKW)** *If* **NEXP** *has polynomial-size circuits then* **NEXP** = **MA**.

Their proof does not relativize even with the weaker conclusion **NEXP** = **EXP**. However if one looks carefully at their proof one sees that the only nonrelativizing tool they use is once again Theorem 4.4. In fact the following version of Theorem 4.5 does hold relative to all oracles.

**Theorem 4.6 (IKW)** *If* **NEXP** *has polynomial-size circuits and* **EXP** = **AM** *then* **NEXP** = **EXP**.

Perhaps the most interesting potentially nonrelativizing technique in derandomization is due to the "other" Impagliazzo-Wigderson result [IW98].

**Theorem 4.7 (Impagliazzo-Wigderson)** *If* **BPP** $\neq$ **EXP** *then* **BPP** *infinitely often has a subexponential heuristic simulation.*

Their proof uses a random-self-reducible property of the permanent function which in itself does not relativize. This is the same property that led to a nonrelativizing interactive proof system for the permanent [LFKN92]. It remains open whether there exists a relativizing proof of Theorem 4.7.

Impagliazzo, Kabanets and Wigderson [IKW01] use Theorem 4.7 to prove the following interesting consequence.

**Theorem 4.8 (IKW)** *If every tally set in* **EXP** *is in* **BPP** *then* **EXP** = **BPP**.

We give a relativizing proof for this result.

**Proof of Theorem 4.8:** We consider two cases based on whether **EXP** has polynomial-size circuits.

If **EXP** does not have polynomial-size circuits then by Theorem 4.3, **BPP** is infinitely often in subexponential time but by simple diagonalization there exist tally sets in **EXP** that do not have this property.

If **EXP** has polynomial-size circuits, let $K$ be an **EXP**-complete set and suppose $K$ has circuits of size $n^j$. Let $L$ consist of $1^{\langle n,i\rangle}$ such that the $i$th bit of the lexicographically first circuit of size at most $n^j$ computing $K$ on strings of length $n$ is one. Note that $L$ is a tally set computable in **EXP** so by assumption $L$ is in **BPP**. To compute $K$ in **BPP**, on input $x$ we first find the circuit $C$ for $K$ on strings of length $|x|$ by computing $L$. We then just output $C(x)$. □

## 5 Further Research

How does an hypothesis like Promise-**ZPP** is easy fit in with the other hypotheses? If Promise-(**NP** ∩ **coNP**) is easy then **P** = **NP** by doing a self-reduction [ESY84]. One might hope that a similar technique could show that Promise-**ZPP** is easy would imply Promise-**RP** is easy.

## Acknowledgments

I would like to thank Dieter van Melkebeek who suggested the problem, helped describe the results in Sections 2 and 4 and has given many useful comments on the current exposition. Vinodchandran Variyam has also given useful comments on the exposition. I also thank Harry Buhrman, Steve Cook and Luca Trevisan for helpful discussions.

## References

[ACR98] A. Andreev, A. Clementi, and J. Rolim. A new general derandomization method. *Journal of the ACM*, 45(1):179–213, January 1998.

[ACRT99] A. Andreev, A. Clementi, J. Rolim, and L. Trevisan. Weak random sources, hitting sets, and BPP simulations. *SIAM Journal on Computing*, 28(6):2103–2116, December 1999.

[BF99] H. Buhrman and L. Fortnow. One-sided versus two-sided randomness. In *Proceedings of the 16th Symposium on Theoretical Aspects of Computer Science*, volume 1563 of *Lecture Notes in Computer Science*, pages 100–109. Springer, Berlin, 1999.

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

[BG81] C. Bennett and J. Gill. Relative to a random oracle, $P^A \neq NP^A \neq co-NP^A$ with probability one. *SIAM Journal on Computing*, 10:96–113, 1981.

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

[ESY84] S. Even, A. Selman, and Y. Yacobi. The complexity of promise problems with applications to public-key cryptography. *Information and Control*, 61(2):159–173, May 1984.

[For94] L. Fortnow. The role of relativization in complexity theory. *Bulletin of the European Association for Theoretical Computer Science*, 52:229–244, February 1994.

[GJ79] M. Garey and D. Johnson. *Computers and Intractability. A Guide to the theory of NP-completeness.* W. H. Freeman and Company, New York, 1979.

[Hel84] H. Heller. Relativized polynomial hierarchies extending two levels. *Mathematical Systems Theory*, 17(2):71–84, May 1984.

[Hel86] H. Heller. On relativized exponential and probabilistic complexity classes. *Information and Computation*, 71:231–243, 1986.

[IKW01] R. Impagliazzo, V. Kabanets, and A. Wigderson. In search of an easy witness: Exponential versus probabilistic time, 2001. These proceedings.

[IN88] R. Impagliazzo and M. Naor. Decision trees and downward closures. In *Proceedings of the 3rd IEEE Structure in Complexity Theory Conference*, pages 29–38. IEEE, New York, 1988.

[ISW99] R. Impagliazzo, R. Shaltiel, and A. Wigderson. Near-optimal conversion of hardness into pseudo-randomness. In *Proceedings of the 40th IEEE Symposium on Foundations of Computer Science*, pages 181–190. IEEE, New York, 1999.

[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.

[IW98] R. Impagliazzo and A. Wigderson. Randomness vs. time: De-randomization under a uniform assumption. In *Proceedings of the 39th IEEE Symposium on Foundations of Computer Science*, pages 734–741. IEEE, New York, 1998.

[KC00] V. Kabanets and J. Cai. Circuit minimization problem. In *Proceedings of the 32nd ACM Symposium on the Theory of Computing*, pages 73–79. ACM, New York, 2000.

[KRC00] V. Kabanets, C. Rackoff, and S. Cook. Efficiently approximable real-valued functions. Technical Report 00-034, Electronic Colloquium on Computational Complexity, 2000.

[Kur85] S. Kurtz. Sparse sets in NP−P: Relativizations. *SIAM Journal on Computing*, 14(1):113–119, February 1985.

[KvM99] A. Klivans and D. van Melkebeek. Graph nonisomorhism has subexponential size proofs unless the polynomial-time hierarchy collapses. In *Proceedings of the 31st ACM Symposium on the Theory of Computing*, pages 659–667. ACM, New York, 1999.

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

[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.

[MV96] A. Muchnik and N. Vereshchagin. A general method to construct oracles realizing given relationships between complexity classes. *Theoretical Computer Science*, 157(2):227–258, 5 May 1996.