A Post's Program For Complexity Theory

Harry Buhrman[*] Leen Torenvliet[†]

Abstract

In 1944, E. Post proposed a program that would lead to the identification of separate degrees of recursively enumerable sets. Post proposed to identify structural properties that sets of different degrees would not share. Thus proving such a property for sets in one degree would imply that these sets are not in the other. We propose a similar program for the separation of complexity classes and identify three properties that are potential separators: auto-reducibility, robustness, and mitoticity. Some partial results that do separate complexity classes have already been established. Also, answering the question whether complete sets in certain classes do or do not have these properties *either way* gives an answer to separation problems of central interest.

1 Introduction

The major quest for the complexity theory community is finding methods that may separate complexity classes. For the standard sequential hierarchy of complexity classes, P, NP, PSPACE, EXP, NEXP, EXPSPACE,... we know that e.g., P $\neq$ EXP by the ancient hierarchy theorems. However in the game of separating complexity classes, the hierarchy theorems have a big drawback. Their proofs relativize, as do many other techniques currently known in complexity theory. As there are oracle worlds in which e.g., P = PSPACE and others where P $\neq$ PSPACE, such proofs can never be used to determine the relations we are currently interested in.

Many approaches have been tried in recent and more distant history to prove complexity classes different. We name a few.

Lower bounds. The most direct approach to proving that complexity classes A and B are not the same is of course to take a problem in one class and prove a lower bound on it's computational complexity that shows that it is not in the other. However, most complexity classes of interest have rather large flexibility. Most time bounds that define time complexity classes are closed under taking polynomials and hence a lower bound would imply quite a gap between A and B. As an example, better than linear lower bounds have yet to be proven for satisfiability.

Diagonalization. Another way of proving that A and B are different is by *constructing* a problem in B that is not in A. Therefore we take an enumeration of all

[*]Centrum voor Wiskunde en Informatica; Kruislaan 413; 1098 SJ Amsterdam; buhrman@cwi.nl

[†]University of Amsterdam; ILLC; Plantage Muidergracht 24; 1018 TV Amsterdam; leen@science.uva.nl

"A-machines" and construct a language that is in B, but differs from every "A-machine's language" in at least one point. To be able to diagonalize we have to have a single machine of "B-complexity" that is able to simulate all A languages, in other words B has to have a universal language for A. This often means that A and B are rather far apart. Also, most known diagonalization proofs relativize. Most notably, the well-known time and space hierarchy theorems [11] have proofs of this type and relativize als already mentioned above. There exist broad consensus nowadays that techniques that relativize are of rather limited use.

Stronger Hypotheses. Sometimes it is easier to prove a more general statement, from which the statement of interest then follows To give a rather trivial example, it can be quite hard to prove that a given graph is 4-colorable, yet its 4-colorability would follow easily from its planarity, which is easy to prove. This is the idea behind the formulation of hypotheses even stronger than the separation of complexity classes we are interested in, and work on proving them. As an example, the Berman-Hartmanis conjecture that all NP-complete problems are isomorphic immediately implies P $\neq$ NP, as do the following more recent hypotheses

1. The measure hypothesis: NP does not have p-measure 0.

2. The pseudo-NP hypothesis: there is an NP- language L such that any DTIME(2^{n^e}) language L' can be distinguished from L by an NP refuter.

3. The NP-machine hypothesis: there is an NP machine accepting $\{0\}^*$ for which no 2^{n^e}-time machine can find infinitely many accepting computations.

Recently, Hitchkock and Pavan proved some interesting relations between these hypotheses [10]. Thus far however, attempts to prove that e.g., NP does not have p-measure 0 have not been succesful.

The approach we propose is of a different type. To show that classes A and B are different we could also identify some property such that some sets in B have the property, whereas all sets in A do not have the property. It then follows that $A \neq B$, without the direct proof of a difference in computational complexity. The difference between A and B then proven is the *structural* difference involving the property. This approach is not new. It was first proposed in 1944, by E. Post, who then sought to prove that there were more than just two Turing degrees in r.e. (namely 0 and $0'$). That problem was resolved by Friedberg and Muchnick [8, 12], who constructed incomplete sets by diagonalization. This was however not the type of solution that Post envisioned. His program was aimed at constructing nonrecursive r.e. sets with very "thin" complements, resulting in a set that could not be Turing complete. Such a set was finally constructed by Degtev [7]. In complexity theory a similar approach can be identified e.g., in the work on sparse sets. Mahaney proved early-on that sparse sets cannot be NP-complete unless P = NP as an addition to the aforementioned Berman-Hartmanis conjecture and much work has been done since on sparse sets in complexity (See [9]) In fact sparseness is a structural property with which unconditionally the difference between complexity classes can be shown. It is known that EXP does not have sparse many-one complete sets [3] whereas P does. Therefore P $\neq$ EXP. The interesting question is therefore whether NP has sparse m-complete sets. If it does, then NP $\neq$ EXP and if it doesn't then P $\neq$ NP.

In this survey we concentrate on three different structural properties for which similar results have been achieved. These properties also give a direct separation between complexity classes. Most notably, these properties do *not* relativize. Results that hold for these properties in the real world are not true in some oracle worlds.

1. Auto-reducibility. A set A is auto-reducible if there exists a polynomial time oracle machine M such that $(\forall x)A(x) = M^{A-\{x\}}(x) = M^{A\cup\{x\}}(x)$. That is, membership in A can be determined by M by asking queries of A *other* than x.

2. Robustness. A set A is robust against some other set S under reduction type r if $A - S \equiv_r A$.

3. Mitoticity. A set A is *weakly mitotic* under reduction type r if $A \equiv_r A_1 \equiv_r A_2$ where $A = A_1 \cup A_2$ and $A_1 \cap A_2 = \emptyset$. A is moreover *mitotic* if there exists a set B in P, such that $A_1 = A \cap B$ and $A_2 = A \cap \overline{B}$.

Some obivious relations hold between these properties, e.g., if A is strongly mitotic, then A is also auto-reducible, but the converse is not necessarily true (see [?]), and if A is not auto-reducible, then A is also not robust. For the investigation of the abovementioned properties we focus on complete sets. Why? For the complexity classes in the complexity hierarchy, the inclusion relation holds. If $A \subseteq B$ and $A \neq B$, then at least the complete sets in B are not in A. Therefore, complete sets in B might have properties that complete sets in A do not, or vice versa.

2 Auto-Reducibility

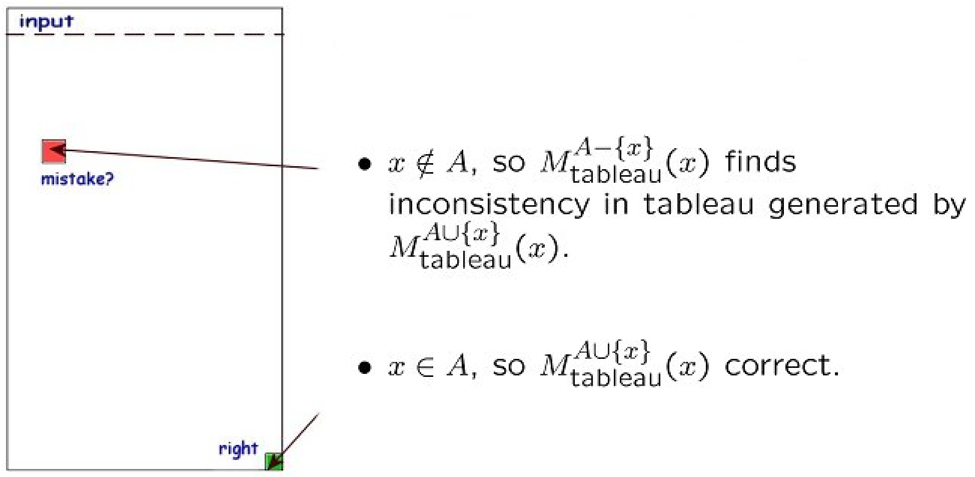
There will not be many proofs in this survey. Nonetheless, we begin this section with the proof of the theorem that all exponential time complete sets are auto-reducible, because it is a beautiful example of a non-relativizing proof and this proof idea underlies many of the other results concerning auto-reducibility and robustness.

Picture an exponential time computation of a Turing machine as in Figure 1, computing membership of a string x in an exponential time complete set A by a Turing machine M running in time $2^{p(|x|)}$. Since this is a tableau of an exponential-time computation, the contents of *every cell* in this tableau can be computed by an exponential-time machine, and since there are only exponentially many cells in the tableau *all* cells can be computed by an exponential-time machine. Since A is complete for exponential time, there exists a polynomial time oracle machine, call it $M_{tableau}$ that can perform either of the following tasks on input x.

1. Use oracle B to compute the outcome of the computation $M(x)$. (Without loss of generality, assume that this outcome can be read from the cell in position $(i, j) = (2^{p(|x|)}, 2^{p(|x|)})$.)

2. Using oracle $B\Delta\{x\}$, find the coordinates of the first inconsistency (i, j) in the computation done in the previous item, or report failure.

Now $M_{tableau}$ simply has to perform these tasks for $B = A - \{x\}$ and/or $B = A \cup \{x\}$. One of these is the correct oracle. Therefore, for one of these it must both compute



- $x \notin A$, so $M^{A-\{x\}}_{\text{tableau}}(x)$ finds inconsistency in tableau generated by $M^{A\cup\{x\}}_{\text{tableau}}(x)$.
- $x \in A$, so $M^{A\cup\{x\}}_{\text{tableau}}(x)$ correct.

Figure 1: an exponential time computation

the right outcome, and report failure in trying to find an inconsistency. Note here that, though $M_{tableau}$ *finds* the inconsistent cells, if they exist, using oracle $B\Delta\{x\}$, checking that they are indeed inconsistent only involves oracle B.

On the other hand, not every complexity class has the property that all Turing complete sets are auto-reducible. We can in fact eventually both code the universal set K and simultaneously diagonalize against polynomial-time Turing reductions. This involves a surprising property of polynomial-time Turing reductions. Namely they can either be forced to reject an input string consistently or will accept a given input string no matter how one plays with the oracle. We dubbed this the "type" of the reduction in [6], type 0 or type 1. Though this property seems mysterious, it is a rather easy consequence of basic logic. Let us first look at the setup of the diagonalization. We take a Turing complete set, which will be complete through a very simple reduction. It will consist of strings $\langle 0, x\rangle$ and $\langle 1, x\rangle$, for strings $x \in \Sigma^*$ and strings $0^{b(n)}$ for some suitable fast increasing function $b(n)$. Without loss of generality, $\langle i, x\rangle$ is not in $\{0\}^*$. In certain regions, we wish to code K, the complete set either on $\langle 0, x\rangle$ or on $\langle 1, x\rangle$ and say whether we did so by putting $0^{b(n)}$ either out of (if coding is on $\langle 0, x\rangle$) or in (if coding is on $\langle 1, x\rangle$) the set we are constructing. This can be done thanks to the aforementioned property: Given n, let A be the set of all strings of length n through 2^n, and some polynomial time oracle machine M. Either $(\forall B \subset A)(\exists C \subset A)$ such that $M^{\langle 0,B\rangle\cup\langle 1,C\rangle}(0^n) = 1$, or $\exists B \subset A$ such that $(\forall C \subset A)[M^{\langle 0,B\rangle\cup\langle 1,C\rangle}(0^n) = 0]$. This is evidently true, since the or part of this claim is just the logical negation of the either part. However this means that any oracle machine can either be forced to accept (either case) while we can encode K on the set $\langle 0, B\rangle$ or it can be set up to reject (or case) while we can encode K on the set $\langle 1, C\rangle$. To complete the reduction, we put 0^n in the set only in the second case and note that this does not change the situation since the auto-reduction cannot query 0^n. It is *only* the vast amounts of strings that have to be considered finding the type of the reduction that makes that the diagonalization must be done in double exponential space. Basically we need to compute the behavior of the Turing reduction on all subsets of A that can be queried on input 0^n before we can decide whether it is of type 0 or type 1. Since a Turing reduction can be viewed as an exponentially bigger truth table, which can be inspected line-by-line to determine the type in double exponential space, this behavior can be computed in double exponential space.

Some remarks can be made.

- Looking closer at the type computation, its complexity can be brought down to a nonconstant number of alternations of exponetnial time quantifiers, since actually only a superpolynomial bound on the length of strings in the type computation is needed.

- Limiting the reduction to truth table reductions, the space used in computing the type can be brought down to single exponential. Therefore, there exist non-truth-table auto-reducible sets in exponential space.

- The type of the reduction can be oracle information. Therefore there exists an oracle A such that EXP^A has non-auto-reducible Turing complete sets.

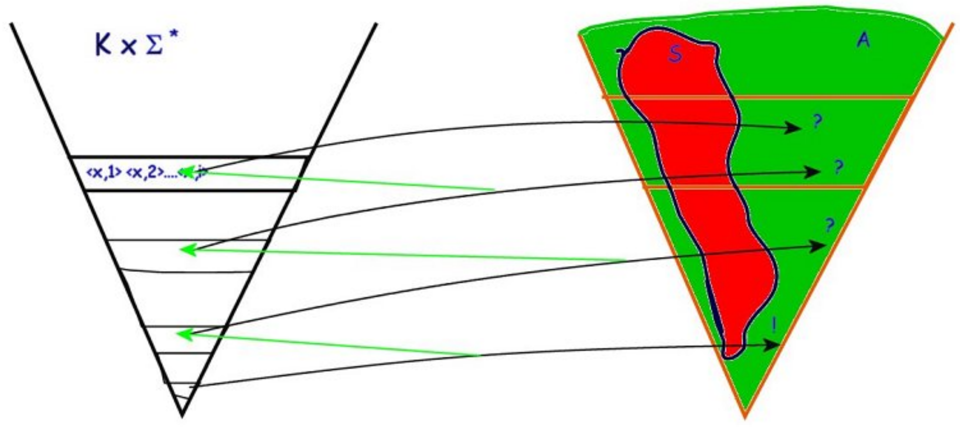
These theorems lead to the following set of consequences of answers to various auto-reducibility questions.

question	yes	no
Are all $\leq_T$-complete sets in EXPSPACE $\leq_T$ auto-reducible?	NL $\neq$ NP	PH$\neq$ PSPACE
Are all $\leq_T$-complete sets in EEXP $\leq_T$ auto-reducible?	NL $\neq$ NP P$\neq$ PSPACE	PH $\neq$ EXP
Are all $\leq_{tt}$-complete sets in PSPACE $\leq_{tt}$-auto-reducible?	NL$\neq$ NP	PH $\neq$ PSPACE
Are all $\leq_{tt}$-complete sets in EXP $\leq_{tt}$-auto-reducible?	NL$\neq$ NP P$\neq$PSPACE	PH $\neq$ EXP

3 Robustness

3.1 Non-adaptive reductions

Robustness of a (complete) set is the extension of the auto-reducibility question to a set of forbidden queries. It is clear that, if a sufficiently dense set is taken as a forbidden set of queries, then completeness is lost. The question that we ask in the robustness setting is: how dense can the set of forbidden queries be and what can it's computational complexity be. Let us begin with an easy example. If EXP has polynomial-size circuits, i.e., a sparse complete set, then EXP also has a tally complete set T. A proof for this can be found, e.g., in [4], but the theorem is much older. Since P $\neq$ EXP, the empty set cannot be complete in EXP. Therefore, if we declare the set $\{0\}^*$ as forbidden queries, the set T will no longer be complete. In other words $T - \{0\}^*$ is not complete for EXP, even if T is. If EXP does *not* have polynomial size circuits then the situation is different. In fact one can easily see that for any A that is many-one complete in EXP (hence not sparse) and any sparse polynomial time computable set S, the set $A - S$ is still many-one complete.



This changes when we let the density of S grow. For any fixed $\epsilon > 0$, the set $\{\langle x, 0^{|x|^{\lceil \frac{1}{\epsilon} \rceil}} \rangle \mid x \in K\}$ is clearly complete for EXP. On the other hand, the set $S = \{\langle x, 0^{|x|^{\lceil \frac{1}{\epsilon} \rceil}} \rangle \mid x \in \Sigma^*\}$ is polynomial-time computable, and $K - S = \emptyset$ and hence not complete in EXP. However, for every many-one complete set A in EXP and *subexponentially* dense and subexponential-time computable set S it is shown in [5] that.

1. $A - S$ remains complete under randomized many-one reductions, and

2. $A - S$ remains Turing complete.

The theorem is proved by observing that since sets in EXP are complete under 1-1 reductions and dense, there must be many strings x such that a many-one reduction of $K \times \Sigma^*$ to A produces a string in $\Sigma^* - S$, i.e., a string that can be queried. Such a string is produced with high probability by a randomized reduction, because there are many. On the other hand, an iterative proces querying $A - S$ can produce such a string. Let f'_K be a 1-1 reduction from $K \times \Sigma^*$ to A. Since S is of subexponential density we only have to compute $f_{K'}(x, y)$ for a subexponential number of strings y to come up with a string that is outside S. Moreover, taking the first $|x|/2$ bits of x as input, say this is z, a computation of complexity roughly $2^{|x|/2}$ can come up with a string y such that $f_{K'}(zv, y)$ is not in S *for all* v with $|zv| = |x|$ (hence $f_{K'}(x, y)$ is also not in S). But then, since A is complete, the outcome of this process can be translated to a query to A which has about half the size of $|x|$. Thus, to be able to answer this query, a new string has to be found that is also outside S, but at a smaller length. At the bottom of this process, see Figure 3.1, simply all strings can be examined. One of these will give a string outside S for the next level higher up, which will give a string for the next level higher up etc., until we find a string that fits x.

The proof can be generalized to 2-tt reductions by, but by the non-autoreducibility result of the previous section, it can *not* be generalized to 3-tt reductions. Moreover, by the same non-autoreducibility result, the proof does not relativize.

3.2 Adaptive Reductions

Since Turing complete sets for EXP may be sparse, the robustness question for sets of polynomial or higher density is difficult to answer without tackling long-standing open problems. We can however address this question for sets of logarithmic density. The basic idea is the same as the proof of the auto-reducibility of exponential-time complete sets. We observe that the tableau that is used there can also be recomputed by a reduction that is not allowed to query a fixed set of small, logarithmic density. Instead of just assuming the answer to the input, such a reduction could assume the answers to a set of queries whose cardinality is limited by the logarithm of the length of the input as well. The number of possible settings of the answers to these queries is bounded by a polynomial in the length of the input. Thus a polynomial number of "competing" algorithms can be set up, that use the oracle $A - S$ to try to answer the question "$x \in A$?"

Some of these algorithms claim $x \in A$, whereas others claim $x \notin A$. As all possible settings to queries that fall into S are considered *at least one* of the algorithms must compute the right answer. Moreover, the algorithms that claim the wrong answer, *must* compute inconsistent contents in at least one place in the tableau. Finnaly, such inconsistent contents can be recovered by an algorithm that has the right[1] oracle setting. The number of different settings and runtimes are bounded by a polynomial. Hence all of these computations can be performed by a single machine using oracle $A - S$.

As with the auto-reducibility results this result does not relativize and can be generalized to include the delta classes in the exponential-time hierarchy. This has the following consequences.

question	yes	no
Are $\leq_m$-complete sets in EXP robust against subexp-dense P sets?	open	EXP $\subseteq 2^{o(n)}$-circuits
Are $\leq_T$-complete sets in EXP robust against sparse P sets?	EXP $\not\subseteq$ P/poly	open
Are $\leq T$-complete sets in EEXP robust against log-dense P sets?	P $\neq$ PSPACE	PH $\neq$ EXP

4 Mitoticity

The third and strongest form of redundant storage of information in a complete set is that of mitoticity. In the robustness problem we ask what happens to the complete set when we take out a set S. In the mitoticity problem we are not only interested in the density of the set taken out, but also in its structure. What if we take out of a complete set A *another* complete set B. Can A *still* remain complete? In other words, can we *split* a complete set A into two complete sets A_1 and A_2. As we mentioned in the introduction, there are two different types of this mitoticity property. The most general is the one in which we just ask: ist there a separation? We call this notion weak-mitoticity. A stronger notion, introduced under the name mitoticity by Ambos-Spies [2] is the notion where there exists a set B in P where $A_1 = A \cap B$ and $A_2 = A \cap \overline{B}$. Of course, mitoticity

[1]An algorithm trying to find inconsistencies in the tableaux computed by competing algorithms may ask a different set of queries, yet as the total number of queries is bounded by $\log|x|$ the total number of settings is still bounded by a polynomial.

implies weak-mitoticity. Mitoticity questions are dependent on the type r of the reduction and are therefore often specified as r-mitoticity, respectively weak-r-mitoticity.

A proof that all many-one complete sets in EXP are weakly-p-m-mitotic can be found in [4]. The following proof that many-one complete sets in EXP are also p-m-mitotici is due to Kurtz [13].

Consider a many-one complete set A in EXP. It is known that A is complete under length increasing reductions. Let f be a length increasing reduction from $K \times \{0\}^*$ to A, and let p be a polynomial such that $(\forall x)[|x| < |f(x)| < p(|x|)$. Now set $g(1) = 1$ and $g(n) = p(g(n-1) + 1$ and let $B = \{y | g(2n) \leq |y| < g(2n+1)\}$. Clearly, $B \in$ P. Also, $A \cap B$ and $A \cap \overline{B}$ are many-one complete for EXP. To see this we provide a reduction from K to $A \cap B$ (A reduction from K to $A \cap \overline{B}$ can be constructed similarly. For $x \in \Sigma^*$ we compute $f(\langle x, \epsilon \rangle), \ldots, f(\langle x, 0^{p(|x|+1)} \rangle)$. At least one of these strings must be in B, hence in $A \cap B$ if and only if $x \in K$. We use the first such string in this sequence as the query string for our reduction from K to $A \cap B$.

The above proof can be extended to show that the complete sets under 2-tt reductions are also 2-tt mitotic. The idea of the proof is, given that thee are only 16 different 2-tt reductions, a 2-tt complete set can, by case analysis, be shown to be complete under 2-tt reductions that depend only on queries *longer* than the input. Then Kurtz' proof can be adapted to also show the theorem for 2-tt reductions.

From the relation between mitoticity and auto-reducibility, it follows again that this does not go through for 3-tt reductions. The question whether all Turing complete sets in EXP are also Turing mitotic still remains open. A nice picture of consequences to mitoticity questions as in the previous section cannot (yet) be made.

Recently Glaßer et al. [?] proved some new results on mitoticity and autoreducibility. They showed, e.g., that the PSPACE complete sets are weakly Turing mitotic, which makes the remaining open question for the Turing mitoticity for EXP more urgent, but it also implies that proving non-mitoticity for EXP could be quite hard. A yes/no separation result like the results in the previous sections could be a non-mitoticity result for EXPSPACE, which can be put next on this agenda.

5 Conclusions

Auto-reducibility, Robustness, and Mitoticity seem good candidates for structural properties that may show a separation of complexity classes that are dear to us. Of these we think that Robustness is the most promising, since it has the most dimensions along which a separation can be achieved. Namely, type of the reduction, complexity of the complete set, density of the forbidden set, and complexity of the forbidden set. Mitoticity on the other hand has the most still open problems of the three and therefore we cannot say much about its potential. We will continue working on this.

References

[1] K. Ambos-Spies. p-mitotic sets. In E. Börger, G. Hasenjäger, and D. Roding, editors, *Logic and Machines, Lecture Notes in Computer Science 177*, pages 1–23. Springer-Verlag, 1984.

[2] L. Berman. On the structure of complete sets: Almost everywhere complexity and infinitely often speedup. *Proc. 17th IEEE Symposium on Foundations of Computing*, pages 76–80, 1976.

[3] H. Buhrman, A. Hoene, and L. Torenvliet. Splittings, robustness and structure of complete sets. *SIAM Journal on Computing*, 27(3):637–653, 1998.

[4] H. Buhrman and L. Torenvliet. Separating complexity classes using structural properties. In *Proceedings 19th IEE Conference on Computational Complexity*, pages 130–138. IEEE Computer Society Press, 2004.

[5] H. Buhrman, D. van Melkebeek, L. Fortnow, and L. Torenvliet. Using autoreducibility to separate complexity classes. *Siam Journal on Computing*, 29(5):1497–1520, 2000.

[6] A.N. Degtev. *tt* and m-degrees. *Alg. Log.*, 12:143–161, 1973.

[7] C. Glaßer, M. Ogihara, A. Pavan, A. Selman, and L. Zhang. Autoreducibility, mitoticity, and immunity. Technical Report 2004-22, Department of Computer Science and Engineering, State University of New York at Buffalo, Buffalo, NY, USA, December 2004.

[8] R.M. Friedberg. Two recursively enumerable sets of incomparable degrees of unsolvability. In *Proc. Nat. Acad. Sci.*, volume 43, pages 236–238, 1957.

[9] L. Hemachandra, M. Ogiwara, and O. Watanabe. How hard are sparse sets? In *Proc. Structure in Complexity Theory 7th Annual Conference*, pages 222–238, Boston, Mass., 1992. IEEE Computer Society Press.

[10] J.M. Hitchkock and A. Pavan. Hardness hypotheses, derandomization, and circuit complexity. In *24th Conference on Foundations of Software Technology and Theoretical Computer Science*, pages 336–347. Springer Verlag, December 2004.

[11] J. Hartmanis and R. Stearns. On the computational complexity of algorithms. *Trans. Amer. Math. Soc.*, 117:285–306, 1965.

[12] A.A. Muchnik. On the unsolvability of the problem of reducibility in the theory of algorithms. *Dokl. Acad. Nauk SSSR*, 108:194–197, 1956.

[13] S.Kurtz. Private communication.