## 7 Appendix - Further discussion of provably recursive functions

In this appendix we discuss some difficulties related to Lemma 5 (in section 3.1). In particular, we explain why can't we use this lemma to conclude that $PA_1 \vdash P \neq NP$ iff $PA \vdash P \neq NP$. To demonstrate these difficulties, consider the following algorithm (which we denote by $A$), that is a variation of the algorithm in example 1:

*On input $x$, if $x$ is a code of a proof in $PA$ of a contradiction - reject $x$.*
*Otherwise - enter an infinite loop.*

Clearly, since $PA$ is consistent, then $A$ halts on every input. Moreover, we can bound its running time by, say, $n^3$. (the time that takes to verify that $x$ is not a code of a contradiction proof). Let us now define the function $EXEC_A$:

*$EXEC_A(x) =$ the first (and only) $y$ s.t. $|y| \leq |x|^3$,*
*and $y$ is (a coding of) an accepting execution of $A$ on $x$.*

Since the claim: "*$\forall x\ \exists y$ of length $\leq |x|^3$ s.t. $y$ is (a coding of) an accepting execution of $A$ on $x$*"
is a $\Pi_1$ formula that is true in $\mathcal{N}$, then $PA_1 \vdash$ "*$EXEC_A$ is a complete function*". A non-careful usage of lemma 4, would lead to the conclusion that $PA \vdash$ "*$EXEC_A$ is a complete function*", and hence
$PA \vdash$ "*$PA$ is consistent*", which is a contradiction to Godel's second theorem [Go31].

The catch here lies in the definition of the term "provably recursive function". Let us recall that $f$ is provably recursive in $PA$ iff there exists an algorithm $A_f$ that computes $f$, s.t. $PA \vdash$ *$A_f$ halt on every input.* The definition says nothing about the ability of $PA$ to prove that $A_f$ computes $f$. In the above example, one can easily design an algorithm that computes $EXEC_A$, s.t. $PA$ can prove its totality. But then, $PA$ will not be able to prove that this algorithm computes $EXEC_A$.

Back to our business, assume that $PA_1 \vdash P \neq NP$, then - by theorem 4 - $R^P_{SAT}$ is dominated by $F_\alpha$ for some $\alpha \leq \epsilon_0$. Consider the following algorithm (which we denote by $B$):

*On input $x$, if $R^P_{SAT}(|x|) \leq F_\alpha(|x|)$ - return $R^P_{SAT}(|x|)$*
*otherwise - return 0*

1. Clearly, $B$ halts on every input. We can even bound the running time of $B$ by $F_\alpha(n) \times n^{\log F_\alpha(n)} \times 2^{n+1}$. Since $PA \vdash$ "*$F_\alpha$ is complete*" then $PA \vdash$ "*$B$ halts on every input*".

2. By our assumption, $\forall x, R^P_{SAT}(|x|) \leq F_\alpha(|x|)$, So $B$ really computes $R^P_{SAT}(|x|)$.

From 1 and 2 we conclude that, by definition, $R^P_{SAT}$ is provably recursive in $PA$. Still this does not mean that $PA \vdash$ "*$R^P_{SAT}$ is complete*", since we did not show that $PA \vdash$ "*$B$ computes $R^P_{SAT}$*. Consequently, we do not know how to prove, that in such a case, $PA \vdash P \neq NP$.

[Sm83] Smorynski C., "'Big' News from Archimedes to Friedman", Notice of American Society, Vol 30. (1983), 251-256.

[Sm77] Smorynski C., "The Incompleteness theorem", Handbook of mathematical logic, J.Barwise ed., North-Holland, New-York (1977), 821-865.

[Wa70] Wainer S.S. "A Classification of Ordinal Recursive Functions." Arch. Math Logic, 13 (1970), 136-153.

[Go31] Godel K., “Uber formal unentscheidbare Satze der Principia Mathematica und verwandter Systeme I”, ibid., Vol. 38, (1931), 173-198.

[GRS90] Graham R.L., Rothschild B.L., and Spencer J.H, ”Ramsey theory”, A Wiley-Interscience Publication (1990)

[HH76] Hartmanis J., Hopcroft J.E., “Independence results in computer science”, SIGACT News 8,4 (1976), 13-24.

[Ha85] Hartmanis J., ”Independence results about Context Free Languages and lower bounds”, Information processing Letters 20 (1985) 241-248.

[JY81] Joseph D., Young P., “Independence results in computer science?”, J.Compute.System.Sci. 23 (1981), 311-338.

[JY85] Joseph D., Young P., “A survey of some recent results on computational complexity in weak theories of arithmetic”, Fundamentica Informatica 8, (1985), 104-121.

[Kr52] Kreisel G., “On the concepts of completeness and interpretation of formal systems”, FM, Vol. 39, (1952), 103-127.

[KM81] Krishnamurthy B., and Moll R.N., ”Examples of Hard Tautologies in Propositional Calculus”, Proceedings of 13’th STOC (1981) 28-37.

[KOR87] Kurtz S. A., O’donnell M. J., and Royer J. S., “How to Prove Representation-Independent Independence Results”, Information Processing Letters 24 (1987) 5-10.

[Ku80] Kunen K., ”Set theory”, North-Holland (1980).

[Le82] Leivant D., ”Unprovability of Theorems of Complexity Theory in Weak Number Theories”, Theoretical Computer Science 18 (1982) 259-268.

[Li78] Lipton R.J., “Model theoretic aspects of computational complexity”, FOCUS 19 (1978), 193-200.

[LN88] Loebl M., and Nesetril J., ”Linearity and Unprovability of Set Union Problem Strategies” Proceedings of 20’th STOC 1988 360-366.

[Ly75] Lynch N., ”On reducibility to complex or sparse sets”, Journal of Association for Computing machinery, Vol. 22, No. 3, (1975), 341-345.

[PH77] Paris J., and Harrington L., “A mathematical incompleteness in Peano arithmetic”, Handbook of mathematical logic, J.Barwise ed., North-Holland, New-York (1977), 1133-1142.

[Sm80] Smorynski C., “Some rapidly growing functions”, Mathematical intelligencer 2 (1980), 149-154.

2. *$n \overset{*}{\rightarrow} (l)^r_k$ denotes the claim: For every function $F$ from the $r$-subsets of $\{1, 2, \ldots, n\}$ to a set of size $k$, there exists a large subset $S \subseteq \{1, 2, \ldots, n\}$ of cardinality at least $l$, such that $F$ is constant on the $r$ subsets of $S$.*

**Example:** Let us define the language

$$L_{PH} = \{< 1^k, 0^n >: n \overset{*}{\rightarrow} (k+1)^k_k\}$$

1. The following is an immediate consequence of Theorem 3.2 of Paris and Harrington [PH77]:
$$PA \not\vdash \text{“}\{k : \exists n(< 1^k, 0^n > \in L_{PH})\}\ is\ infinite\text{“}$$
In particular, $PA \not\vdash$ “*$L_{PH}$ is not regular*”.

2. A standard pumping argument shows that $L_{PH}$ is not a Context Free Language.

3. It can be shown that $L_{PH} \in Co - NP$

## 6 Acknowledgments

We wish to thank Michal Dvir, her M.Sc. thesis (under the first author‘s supervision), has acquainted us with the ideas and techniques of proof theory that underline this research. We also owe a lot to Menachem Magidor for a very insightful discussion at an early stage of this work. We also thank Janos Makowsky for his inspiring interest in the applications of Model-Theory to Computer Science.

## References

[Be92] Ben-David S., ”Can Finite Samples Detect Singularities of Real-Valued Functions?”. To appear in Proceedings of STOC‘92.

[BD91] Ben-David S., and Dvir M., ”Non-Standard Models for Independent Arithmetical Statements”, Technical Report, Technion, (1991)

[BI87] Blum M., and Impagliazzo R., “Generic Oracles and Oracle Classes”, 28‘th Symposium on Foundations of Computer Science, (1987),118-126.

[Bo74] Book R.V., ”Tally languages and complexity classes”, Information and Control, 26, (1974), 186-193.

[Co63] Cohen P.J., “The independence of the continuum hypothesis”, Proceeding of the National Academy of Science, USA 50, 1143-1148 (1963)

[FLO83] Fortune S., Leivant D., O’donnel M., “The Expressiveness of Simple and Second-Order Type Structures”, Journal of the Association for Computing Machinery, Vol. 30, No. 1 (1983), 151-185.

**Theorem 13***:*

1. *For every $f$ there exists a language $D_f$, computable in $f$, such that both $D_f$ and its complement $\bar{D}_f$ are $f$-meager. (Furthermore, for every recursive $f$ there exists a recursive $f'$ that dominates $f$ such that $D_{f'}$ is computable in lineartime and in logspace).*

2. *If $D$ is $f$-meager then so is $L \cap D$ for every $L$.*

3. *For every meager language $L$, the statement "$L$ is not finite" is not provable in $PA_1$.*

4. *For every language $L$ there exists a partition $L = L_0 \cup L_1$ such that neither $L_0$ nor $L_1$ can be proved to be infinite within the framework of Peano Arithmetic. (Furthermore, each of $L_0, L_1$ is computable from $L$ in lineartime and logspace).*

**Proof:** The theorem easily follows from the proof of Theorem 4.4 of [KOR87].

The following corollary is implicit in [KOR87]. In their Theorem 4.4 they could have also stated that the language $L_0$ they define, the language that the theory cannot prove to be infinite, is actually outside the complexity class $S$.

**Corollary 14***:* *There exist arbitrarily complex languages such that $PA_1$ cannot prove that they are not finite languages.*

Let us conclude this section by demonstrating an inherent 'easiness' property of meager languages, namely, that using such a language as an oracle cannot significantly speed up any computation.

**Lemma 11***:* *Let $G$ be a meager language. If for a standard language $L$ and an SCF $f$, $PA_1 \vdash$"$L \notin DTIME(f)$" then, $PA_1$ cannot prove that an Oracle Turing Machine can compute $L$ in time $f$ using $G$ as an oracle.*

Note that by our main results, if the negation of a statement $L \in DTIME(f)$ is not provable in $PA_1$, then the statement itself is true (semantically) in $\mathcal{N}$ (up to adding the inverse of any Wainer function to $f$).

Let us also remark that, as any generic oracle (in the sense of Blum-Impagliazzo [BI87]) is necessarily a meager language, the last lemma offers an easy proof to to their Theorem 1.5.

## 5.1 A concrete example of non-provability

Let us demonstrate the emergence of non-provability of a lower bound, by translating the Paris Harrington version of Ramsey Theorem into a decision problem of a binary language. For more on this theorem see [GRS90].

**Definition 11***:*

1. *A set $S \subseteq N$ is* large *if its cardinality is bigger than its first element.*

## 5 Meager Languages

To help focusing on the source of non-provability of complexity statements, we introduce the notion of *Meager Languages*. Our notion of meager languages is quite similar to [KOR87]'s notion of an emaciated subset of $N$. Somewhat like Sparse sets [Ly75] and Tally sets [Bo74], meager languages combine some inherent 'easiness' properties with the possibility of being arbitrarily complex. We shall show that on one hand, there are arbitrarily complex languages that are meager, while on the other hand, (all) meager languages are minimal elements in the lattice of Turing reducibility (i.e. they are useless as oracles).

Our interest in meager languages is mainly due to the phenomena they exhibit with respect to formal provability: If $L$ is a meager language, then the statement "$L$ is infinite" is not provable in $PA$ (as well as in $PA_1$).

**Definition 9***:* *Let $f : \mathcal{N} \to \mathcal{N}$ be a function such that $f(n) > n$ for all $n$. We call a language $L$* f-meager *if, for infinitely many $n$'s, any string in $L$ is either shorter than $n$ or longer than $f(n)$.*

intuitively speaking, meager languages are languages that infinitely often skip big chunks of lengths, avoiding all strings of such skipped lengths.

**Definition 10***:*

1. *A language $L$ is* meager *if it is $f$-meager for every $f$ in the Wainer hierarchy - $\{F_\alpha : \alpha < \epsilon_0\}$.*
2. *For a class $C$ we say that a language $L$ is* C-meager *if there exists some $L' \in C$ such that the symmetric difference $L \bigtriangleup L' = \{x : L(x) \neq L'(x)\}$ is meager.*

**Claim 1***:*

1. *Define a function $M_L(n) \overset{def}{=}$ '*the n'th i for which $L \bigcap \Sigma^i \neq \Phi$*'. A recursive $L$ is meager iff $M_L$ is not provably recursive.*
2. *If a class $C$ is closed under complementation then so is the class of all $C$-meager languages.*

It appears that there is a strong connection between the notion of $C$-meagerness and provability:

**Lemma 10***:* *For every function $g$, let us denote DTIME($n^{g(n)}$) by $C_g$. $PA_1 \not\vdash P \neq NP$ if and only if $SAT$ is $C_g$-meager for every S.C.F $g$.*

**Proof:** By theorem 4, $PA_1 \not\vdash P \neq NP$ iff $R^P_{SAT}$ is not dominated by Wainer hierarchy. By Lemma 2, if $h$ is an S.C.F. and $R$ isn't dominated by the hierarchy then for every $f$ in the hierarchy, there exists infinitely many $n$'s s.t. $\forall n \leq m \leq f(n)$, $R^{-1}(n) \leq h(n)$.

Let $A$ be the algorithm of lemma 7 (for computing $SAT$), and let $A$' be a similar algorithm, with running time bounded by $n^{g(n)}$, then $L(A')\triangle SAT$ is meager iff $R^P_{SAT}$ is not dominated by the Wainer hierarchy. □

Following [KOR87], we apply the above characterization of non-provability to prove the existence of wide class on languages whose best provable lower bounds lie far below their true worst case complexity.

## 4.2 Existence of One Way Functions

In this section, we consider the implications of our main theorem to the existence of one-way-functions. We use a slight variation of the common definitions of one-way-functions:

**Definition 8:** *Let $f$ be a function that is computable in polynomial time.*

1. *Let $g$ be any S.C.F. We say the $f$ is a $g$-one-way-function in the (non-)uniform formulation if $\forall$ deterministic (non-uniform) algorithm $A$ in DTIME($n^{g(n)}$) ( $NU(n^{g(n)})$ ) $\exists$ $N$ such that $\forall n > N$ $\exists x \in \{0,1\}^n$ for which $f(A(x)) \neq x$*

2. *We say the $f$ is a (non-)uniform one-way-function if there exists a S.C.F. $g$ s.t. $f$ is a (non-)uniform $g$-one-way-function.*

Note that our definitions are weaker then the usual ones, in that we only demand that deterministic (non-uniform) algorithm will fail reversing $f(x)$ on at least one input of every length. On the other hand, it is stronger then the usual one in that we demand not only that every polynomial algorithm can't reverse $f$, but also the existence of a super-polynomial function $h$ such that every algorithm in DTIME($h$) ($NU(h)$) fail to do so.

As an immediate corollary, from the above discussion, we get:

**Corollary 11:** *If $PA_1 \nvdash P \neq NP$ then uniform one-way-functions do not exist.*

**Proof:** Let $f$ be a function that is computable in polynomial time, and let $g$ be a S.C.F. We denote $R^P_{SAT}$ by $R$. By theorem 3, SAT is solvable by a deterministic algorithm with complexity $n^{(1+\log(R^{-1}(n))} \times R^{-1}(n)$. Since $f$ is computable in polynomial time, then the problem of inverting $f$ is clearly in $NP$. (A nondeterministic TM can, on input $y$, simply guess $x$ and then compute $f(x)$ and see if it equals $y$). Thus, the problem of inverting $f$ is reducible to SAT. This means that $\exists$ deterministic algorithm $M$ that inverts $f$ and works in time complexity of $T(n) = n^{k[1+\log(R^{-1}(n^k)]} \times R^{-1}(n^k)$ for some $k$.

Since $g$ is a standard complexity function, then so is the function $h(n) = 2^{\frac{1}{2k}g(\log n)}$. By theorem 4, if $PA_1 \nvdash P \neq NP$ then A is not dominated by Wainer hierarchy, and by claim 1 there exists infinitely many $n$'s s.t.

$$R^{-1}(n^k) < h(n^k) = 2^{\frac{1}{2k}g(\log n^k)}$$

and thus $2k\log(R^{-1}(n^k)) < g(\log n^k) < g(n)$

Since for big enough $n$'s hold $k[1+\log(R^{-1}(n^k))] + \frac{\log(R^{-1}(n^k))}{\log n} < 2k\log(R^{-1}(n^k))$

Then there are infinity many $n$'s s.t. $k[1+\log(R^{-1}(n^k))] + \frac{\log(R^{-1}(n^k))}{\log n} < g(n)$

hence $T(n) = n^{k[1+\log(R^{-1}(n^k)]} \times R^{-1}(n^k) < n^{g(n)}$

This means that an algorithm that works like $M$, but halt after at most $n^{g(n)}$ steps, will succeed in inverting $f$ for infinity many lengths, and thus, $f$ is not a $g$-one-way-function.

$\square$

We can prove a similar corollary for the non-uniform case as well:

**Corollary 12:** *If $PA_1 \nvdash SAT \notin NU(P)$ then non-uniform one-way-functions do not exist.*

Note that the intervals in the corollary are the same for all languages in $NP$.

**Proof:** Let $L$ be any language in $NP$. Since L has a polynomial reduction to $SAT$, then there is a polynomial algorithm translating inputs to $L$ of length n, to inputs to $SAT$ of length at most $Q(n)$ for some polynomial Q. By the above discussion, under the assumptions of the corollary, $SAT$ has infinitely many easy intervals. Furthermore, applying Lemma 2 again, it can be seen that there exist (infinitely many) such intervals of the form $[n, A(A(n))]$. It follows that for big enough $n$'s (for which $Q(n) < A(n)$ ) the intervals $[n, A(n)]$ are 'easy' for L. $\square$

Krishnamurthy and Moll show in [KM81] an example of infinitely many tautologies that are probably hard instances for $\overline{SAT}$. These tautologies appear too frequently to avoid our 'easy intervals'. Therefore, assuming their tautologies are hard (say, $\Omega(n^{\alpha(n)})$), there is no proof for the independence of $P \neq NP$.

## 4.1 Provability in the Non-Uniform model

We now turn our attention to provability of questions in the Non-Uniform Circuit complexity model. First we consider a family of non-uniform complexity classes, for which we can show a surprisingly strong result:

**Theorem 8:** *Let L be any standard language and let $f$ be any standard complexity function.*

1. *If $L \notin$ Non-Uniform-size(f) ($NU(f)$) then $PA \vdash L \notin NU(f)$.*

2. *If $L \in NU(f)$ then $PA_1 \vdash L \in NU(f)$.*

**Proof:**

1. if $L \notin NU(f)$, there exists an integer $n$ s.t. for every boolean circuit $C_n$ that has at most $f(n)$ gates, there is an input $x$ of length $n$, s.t. $C_n(x) \neq L(x)$. The last claim can be stated as an existential formula. It is known that every existential formula that is true in $\mathcal{N}$ can be proved by $PA$, and hence $L \notin NU(f)$ is provable in $PA$.

2. If $L \in NU(f)$ then for every $n$ there is a circuit $C_n$ of size $\leq f(n)$ s.t. for every input $x \in \{0,1\}^n$ holds $C_n(x) = L(x)$. This is a $\Pi_1$ statement that is true in $\mathcal{N}$, and hence - it is provable in $PA_1$. $\square$

Things are not that easy when we deal with NU complexity classes that are not defined by a single function, such as Non-Uniform-P ( NU(P) ). In this case, we use the non-uniform approximation rate to state a theorem, and a corollary similar to theorem 4, and corollary 6.

**Theorem 9:** *$PA_1 \vdash$ "$SAT \notin NU(P)$" if and only if there is a function in Wainer hierarchy that dominates $NR^P_{SAT}$.*

**Corollary 10:** *If it is provable (in any method known today) that $SAT \notin NU(P)$ is independent of $PA$, then $SAT \in NU(n^{f^{-1}(n)})$, where $f$ is not dominated by Wainer hierarchy.*

The proof is similar to the proof of theorem 4 and corollary 6.

**Proof:**

If: Suppose there is a function $g$ in the hierarchy that dominates $R^P_{SAT}$. By the definition of the approximation-rate, $\forall i\ \exists x$ *of length* $\leq R^P_{SAT}(i)$ *s.t.* $M_i(x) \neq SAT(x)$. Since for all sufficiently large $i$, $R^P_{SAT}(i) < g(i)$, and by lemma 1, such $g$ is provably recursive, then the claim:

$$\forall i \exists x\ s.t.\ l(x) < g(i)\ and\ M_i(x) \neq SAT(x)$$

is a $\Pi_1$ formula, being true in $\mathcal{N}$ it is therefore provable in $PA_1$.

Only-if: If $R^P_{SAT}$ is not total then, by Lemma 7, $P = NP$. In such a case, the soundness of $PA_1$ prevents it from proving the inequality. So assume $R^P_{SAT}$ is a total function. If it isn't dominated by any function in Wainer hierarchy, then by Lemma 1, $PA_1 \not\vdash$ "$R^P_{SAT}$ *is complete*". By Lemma 7, this implies that $PA_1 \not\vdash$ "$P \neq NP$". $\square$

Since we didn't use any special properties of SAT or $P$ in the last proof, we can apply the same proof for every language and every complexity class.

**Theorem 5***: Let $C$ be any standard complexity class and let $L$ be any standard language: $PA_1 \vdash$ "$L \notin C$" if and only if $R^C_L$ is dominated by Wainer hierarchy.*

**Corollary 6***: If it is provable (in any method known today) that $P \neq NP$ is independent of $PA$, then the search problem of $SAT$ (i.e. on input $x \in SAT$, find an assignment that satisfies $x$) is in DTIME($n^{f^{-1}(n)}$), where $f$ is not dominated by Wainer hierarchy.*

**Proof:** By corollary 3 (Section 3.1), if it is provable that $P \neq NP$ is independent of $PA$ then it is also independent of $PA_1$. By theorem 4, this means that the approximation rate of $SAT$ to $P$, (which we denote by $f$), is not dominated by any function in Wainer hierarchy. Since we know that $x \in SAT$, we can use the algorithm of lemma 7 without computing $f^{-1}(n)$ in advance. By the argument of lemma 7, the search problem is solvable in deterministic time $T_A(n) = f^{-1}(n) \times n^{[1+\log(f^{-1}(n))]} \leq n^{f^{-1}(n)}$. $\square$

## Interpretation

To appreciate the significance of the last corollary, note that $f^{-1}$ is constant on any interval of the form $f(n), f(n)+1, f(n)+2, \ldots, f(n+1)$. By Lemma 2 this implies that, for an $f$ as implied by the corollary, $n^{f^{-1}(n)}$ is infinitely often bounded (from above) throughout very long intervals by, say, $n^{\alpha(n)}$ (where $\alpha(n)$ is the inverse of the Ackermann function $A(n)$). We call such an interval - an 'easy interval' for $SAT$.

Could it be the case that $SAT$ is easy on (infinitely many) long stretches of $n$'s and yet it has infinitely many hard instances?

An indication to the implausibility of such a situation is demonstrated by the following:

**Corollary 7***: If it is provable (in any method known today) that $P \neq NP$ is independent of $PA$, then there exists infinitely many intervals of the form [n, $A(n)$], such that for every language $L \in NP$ all these intervals (except, perhaps, finite number of them) are 'easy intervals' for L.*

**Lemma 7:** *For every language $L$ and every class $C$, $R_L^C$ is a total function iff $L \notin C$.*

In particular - we'll consider the approximation rate of $P$ by $SAT$. Let us pick a standard enumeration $\{\hat{M}_i : i \in N\}$ satisfying:

1. The function $DESC(i) = \hat{M}_i$ can be computed in linear time.

2. For every $i$, $M_i$ runs at most $n^{\log i}$ steps on every input.

**Lemma 8:** *Let us denote the approximation rate of SAT by $P$ ($R_{SAT}^P$) by $R$. If $R^{-1}$ is bounded by some easily computed function $g$, then SAT is in DTIME( $n^{[1+\log g(n)]} \times g(n)$ ).*

**Proof:** Consider the following algorithm:

```
for $i := 1$ to $g(|x|)$ do begin
    Compute $\hat{M}_i$ ; Use a simulation of $M_i$ as a $SAT$-oracle to find a satisfying assignment for $x$; If you find one - accept $x$.
end
reject $x$.
```

Since $R$ is the approximation rate of SAT to $P$, we don't have to check more then $R^{-1}(n) \leq g(n)$ TMs until we find one that gives right answer on every input of length $\leq n$. For $i \leq g(n)$ $M_i$ runs for at most $n^{\log g(n)}$ steps, and we call each $M_i$ at most $n$ times, so the whole process ends within $n^{[1+\log g(n)]} \times g(n)$ many steps. □

The following is a natural extension of the notion of approximation rate to the context of Non-Uniform circuit complexity.

**Definition 7:** *Let $C = \bigcup_k NU(f_k)$ (where $NU(f_k)$ denotes the class Non-Uniform-size($f_k$)), and let $L$ be a language. The non-uniform approximation rate of $L$ by $C$ is the function:*

$$NR_L^C(i) = max_{j \leq i}\{min\{n : \forall\, circuit\ C\ of\ size \leq f_j(n),\ \exists\, x \in \{0,1\}^n\ s.t.\ L(x) \neq C(x)\}\}$$

**Lemma 9:** *If $C = \bigcup_k NU(f_k)$ and $L$ is a language for which $NR_L^C = g(n)$ then, $L$ is in $NU(f_{1+g^{-1}(n)}(n))$.*

For an example -let Non-Uniform($P$) $= \bigcup_k NU(n^k)$ and let us denote by $g$ the function $NR_{SAT}^P$, then $SAT$ is in $NU(n^{1+g^{-1}(n)})$.

## 4 The Main Results

**Theorem 4:** *$PA_1 \vdash P \neq NP$ if and only if there exists $F_\alpha$ for $\alpha < \epsilon_0$ in the Wainer hierarchy, that dominates the approximation rate of SAT to $P$ ($R_{SAT}^P$).*

Well, all this is very nice, but why should we care about provability in a non-recursive theory? Of course, whenever we show that some statement is not provable in $PA_1$ we also get its non-provability in $PA$. The distinctive virtue of $PA_1$ is that, for a significant class of properties, the above implication can be also reversed. This is manifested by the following lemmas. Both lemmas are part of the folklore of proof theory. [FLO83] present a proof of the next lemma and attribute it to G. Kreisel.

**Lemma 5***:* [6] *A function is provably recursive in $PA_1$ if and only if it is provably recursive in $PA$.*

**Lemma 6***:* *For a relational structure $M$ let $\Pi_1(M)$ denote $\{\psi : \psi \text{ is a } \Pi_1 \text{ formula that holds true in } M\}$. If the non-provability of a statement $\phi$ relative to a theory $\mathcal{T}$ can be demonstrated by starting with a model $M$ of $\mathcal{T}$ in which $\phi$ holds and constructing a submodel in which $\neg\phi$ holds, then $\phi$ is also unprovable in $\mathcal{T} \bigcup \Pi_1(M)$.*

Any known technique for proving independence from sufficiently strong theories of statements that are neither self-referential nor inherently proof-theoretic, meets the assumptions of Lemma 6.

**Corollary 3***:* *Any independence result from $PA$ or from $ZFC$ obtained through any of the approaches currently known, can be extended to an independence result relative to that theory augmented with all true $\Pi_1$ statements.*

## 3.2 Approximating a Language by a Complexity Class

In order to characterize non-provable complexity statements, we define a measure of distance between a language $L$, and a complexity class $C$ – the approximation rate of $L$ by $C$. We will show that "$L \notin C$" is not provable iff the approximation rate of $L$ by $C$ is an extremely fast growing function. We will also show a close relationship between the approximation rate and worst-case complexity.

The idea of approximation rate of a complex language by a class of easier languages is analogous to the idea of Diophantine approximations to Real numbers by Rationals. Pushing this analogy further, languages that are outside $C$ but not provably so - resemble Liouville numbers. We exploit this idea further in [Be92].

**Definition 6***:* *Let $L$ be a language, let $C$ be a complexity class, and let $M_1, M_2, \ldots$ be some canonical enumeration of $C$. The* approximation rate *of $L$ by $C$, is the function:*

$$R_L^C(i) = \ max_{j \leq i}\{min\{|x| : L(x) \neq M_j(x)\}\}$$

*(Where $|x|$ denotes the length of the string $x$).*

Note that the definition of $R_L^C$ depends upon the canonical enumeration we chose. Just the same, all the properties of $R_L^C$ that are relevant to our discussion are invariant with respect to this choice.

[6]Due to a subtle weakness of the definition of 'provably recursive function', this lemma is not as significant as it may sound. See the Appendix for an elaborated discussion.

$PA_1$ enjoys a property that guarantees that non-provability phenomena is inherently due to the languages in question and not an artifact of their presentation. This property is introduced and discussed in [KOR87], they call it *Representation Completeness*. Let us demonstrate (and implicitly define) the representation completeness of $PA_1$ by the following Lemma 3 and Corollary 1. First we first need some notation.

**Definition 5***:*

- *A* Standard Language *is a language $L$ for which the membership relation "$x \in L$" is definable by a $\Pi_0$ formula - a formula having only bounded (by a function of $x$) quantifications.*[5]

- *A class of languages, $C$, is a* Standard Complexity Class *if there exists an easily computable (e.g. linear time) enumeration $\{\hat{M}_i : i \in N\}$ where each $\hat{M}_i$ is a standard code of a Turing machine endowed with a provably recursive (relative to PA) upper bound on its running time, such that $C = \{L(M_i) : i \in N\}$. We call such an enumeration a* canonical enumeration *for $C$.*

Note that 'in practice', any language or complexity class one ever comes across complies with the above definition.

**Lemma 3***:* *Let $L$ be a standard language. If $M$ is a TM whose running time is bounded by some provably recursive $f(n)$, and if $L(M) = L$ then $PA_1 \vdash$ "$L(M) = L$".*

**Proof:** Using standard coding techniques, one can express the claim: "*There is a sequence of successive configurations of $M$, $< C_1, \ldots, C_k >$, with $C_1$ being the starting configuration of $M$ on input $x$ and $C_k$ being an accepting final configuration of $M$*" as a formula with free variables $k$ and $x$. Thus the claim: "*For every $x$ - there is such a sequence of length $< f(n)$ iff $x \in L$*" can be expressed as a $\Pi_1$ formula, and hence, being a true such formula, is an axiom of $PA_1$. $\square$

**Corollary 1***:* *If $M_1$ and $M_2$ are TMs with running time bounded by some provably recursive $f(n)$, then $L(M_1) = L(M_2)$ if and only if $PA_1 \vdash$ "$L(M_1) = L(M_2)$".*

**Corollary 2***:* *If $P = NP$ then $PA_1 \vdash$ "$P = NP$".*

This is true because if $P = NP$ then there is a TM $M$ with running time bounded by $n^k$ (for some constant $k$) that recognizes $SAT$. Let $M'$ be a TM similar to $M$, except that $M'$ has a 'clock' that stops it after $n^k$ steps (if it didn't stop before). Clearly it is provable in $PA_1$ that $M'$ is polynomial. Since $SAT$ is a standard language then, by lemma 2, $PA_1 \vdash$ "$L(M') = SAT$", and hence the claim. More generally -

**Lemma 4***:* *Whenever $C$ is a standard complexity class and $L$ is a standard language, $L \in C$ if and only if $PA_1 \vdash$ "$L \in C$".*

[5]As far as the results of this work go, we could have extended the definition to allow languages definable by $\Pi_1$ formulas as well. But, not being able to come up with a natural example of a language that is not 'standard', we feel that $\Pi_0$ is the more natural choice.

# 3 Setting - Up the Ground

## 3.1 The Theory $PA_1$

The natural, and commonly accepted, formal system for reflecting finitistic mathematical reasoning, is Peano Arithmetic (or, equivalently, Set Theory without the axiom of infinity). We therefore chose to state our final results in terms of provability with respect to $PA$.[3]

Still, as a working tool, we need a stronger theory, $PA_1$, that we define below.

**Definition 4***:*

1. *We say that a formula is a $\Pi_1$ formula if it is of the form $\forall x \phi$ where $\phi$ is a formula in the (first order ) language of arithmetic and all its quantifiers are bounded. (In the context of higher order logics, such formulas are referred to as $\Pi^0_1$). By 'bounded quantifiers' we mean quantification of the form $\forall x < g(y)$, where $g$ is any* provably recursive *function.*[4]

2. *$PA_1$ is the proof system having $PA \bigcup \{\phi : \phi$ is a $\Pi_1$formula that is true in $\mathcal{N}\}$ as its set of axioms. (Here $\mathcal{N}$ stands for the standard model of arithmetic).*

It is not hard to see that $PA_1$ is not recursive (this can be seen, for an example, by noting that the consistency of $PA_1$ itself can be formalized as a $\Pi_1$ statement). Consequently $PA_1$ is not a conceivable candidate for a workable framework for mathematical reasoning.

To motivate our use of a non-recursive proof system, consider the following example of a non-provable statement:

**Example 1***: Let $\mathcal{TH}$ be any sound and recursive proof system for arithmetic. Define an algorithm $A_{\mathcal{TH}}$ by:*

> *On input $x$, if $x$ is a code of a proof in $\mathcal{TH}$ of a contradiction - reject $x$.*
> *Otherwise - accept $x$.*

*Let $A_{\mathcal{TH}}$ run over binary strings and use $L(A)$ to denote the language accepted by $A$. As $\mathcal{TH}$ is consistent $L(A_{\mathcal{TH}}) = \{0,1\}^*$. On the other hand, as Godel's theorem implies that such $\mathcal{TH}$ cannot prove its own consistency, $\mathcal{TH} \not\vdash$ "$L(A_{\mathcal{TH}}) = \{0,1\}^*$".*

This example can be easily modified to produce independence of whichever property that comes to mind: The running time of an algorithm, the totality of a function etc. (And, in a sense, is not too far from the independence results of [HH76].

Clearly, the non-provability exhibited here stems from our very specific choice of algorithm. The same language, $\{0,1\}^*$, can be defined by an algorithm that raises no provability problems.

[3]All of our results can be easily translated to similar theorems for any recursive formal system that extends $PA$ (in particular, to $ZFC$ set theory).

[4]Here we differ from the common definition of $\Pi_1$ formulas [Sm77] that allows only primitive recursive bounds. It is not hard to realize that the relevant proof theoretic results (mainly Lemma 5 below) can be strengthened to apply for our definition. This is shown explicitly in [BD91].

- *If a total recursive function $f : N \rightarrow N$ is provably recursive in $PA$ then, for some $\alpha < \epsilon_0$, it is dominated by $F_\alpha$.*

Furthermore, Fortune Leivant and O'donnell [FLO83] prove that the set of functions that are provably recursive in $PA$ equals that set of functions that can be computed in (deterministic) time, $t(n)$, dominated by some $F_\alpha$ in $\{F_\alpha : \alpha < \epsilon_0\}$.

**Definition 3***:*

- *We call $f : N \rightarrow N$ a* Standard Complexity Function *(SCF), if it is provable in PA that f is total (i.e. defined for every $i \in N$), monotonic, and unbounded.*
- *For any total monotonic and unbounded $f$, we define $f^{-1}$ - the inverse of $f$ - by:*

$$f^{-1}(n) \overset{def}{=} max\{i : f(i) \leq n\}.$$

Examples of standard complexity functions are $f(n) = n^k$, $f(n) = log(n)$, and $f(n) = 2^n$, as well the functions $\{F_\alpha : \alpha < \epsilon_0\}$ in the Wainer hierarchy and their inverses.

**Lemma 2***:*

*If $f$ is a standard complexity function and $g$ is a monotone function that is not dominated by Wainer hierarchy, then there are infinitely many $n$'s satisfying:*

$$\forall m\,[(n < m < A(n) \rightarrow (g^{-1}(m) < f^{-1}(m), f(m) < g(m))]$$

*(We have chosen $A(n)$ just for concreteness, the claim remains valid for any bound on $m$ which is an $SCF$ function of $n$)*

**Proof:**

Let us denote by $h$ the function $h(n) = \max\{\ f(n), f^{-1}(n)\ \}$. Since both $f$ and $f^{-1}$ are $SCF$s, then so is $h$. Let $F_\alpha$ be a function in the hierarchy that dominates both $h$ and the Ackermann Function. Since $g$ is not dominated by Wainer hierarchy, there are infinitely many $n$'s satisfying : $F_{\alpha+1}(n) < g(n)$. Note that, for every $n > 2$, $F_\alpha(F_\alpha(n)) < F_{\alpha+1}(n)$. It follows that for every $m$ that satisfies $n < m < A(n)$:

$$h(m) < h(A(n)) < F_\alpha(F_\alpha(n)) < F_{\alpha+1}(n) < g(n) < g(m)$$

and therefore,

$$g^{-1}(m) < h^{-1}(m) \leq f(m) \leq h(m) < g(m).$$

□

Note, that we didn't use any special property of $A(n)$ in our proof. The same proof can be applied for any other $SCF$ function.

and only if the approximation rate of $L$ by $C$ is an extremely fast growing function (i.e. not provably recursive).

We conclude the paper by a brief discussion of our notion of a language being extremely-fast-approximable by a class of 'easy' languages (say, $P$). We show that, on one hand, such languages exist in arbitrarily high levels of the worst-case-complexity hierarchy, but on the other hand, these languages are 'easy' for all practical purposes (e.g. cryptography).

## 2 Proof Theoretic Background

In this section we review some basic facts from the proof theory of Arithmetic. We refer the reader to [Sm80, Sm83] for an elaborated and truly enjoyable discussion of this topic.

The basic idea goes back to Kreisel [Kr52]. For every recursive formal theory which is sound for arithmetic there exist total recursive functions such that the theory cannot prove their totality. Such functions can be characterized by their rate of growth.

Wainer [Wa70] supplies a useful measuring rod for the rate of growth of recursive functions (from natural numbers to natural numbers) - the *Wainer hierarchy.* The Wainer hierarchy is an extension, to infinite countable ordinal indices, of the more familiar Ackermann hierarchy of functions:

$$\begin{array}{ll} F_1(n) = & 2n \\ F_{k+1}(n) = & F_k^{(n)}(n) \\ \text{and, for a limit ordinal } \delta, & \\ F_\delta(n) = & F_{\delta_n}^{(n)}(n) \end{array}$$

where $F_k^{(n)}$ denotes the $n$'th iterate of $F_k$ and, for each limit ordinal $\delta$, $\{\delta_n : n \in \mathbb{N}\}$ is some fixed recursive sequnce of ordinals increasing to $\delta$. The famous Ackermann Function is $F_\omega$ - quite low in this hierarchy.

**Definition 1***:*

1. *We say that a function $f$* dominates *a function $g$ if for all large enough $n$'s, $g(n) < f(n)$.*
2. *$\epsilon_0$ is the first ordinal $\alpha$ satisfying $\omega^\alpha = \alpha$. (The exponentiation here is ordinal exponentiation and $\epsilon_0$ turns out to be a countable ordinal - the limit of the sequence $\omega, \omega^\omega, \omega^{\omega^\omega}, \ldots$).*

Note that for every $\alpha < \beta$, $F_\beta$ dominates $F_\alpha$. It is also worthwhile to recall that Ackermann Function, $F_\omega$, dominates all primitive recursive functions.

**Definition 2***: A function $f$ is* provably recursive *in a formal theory $\mathcal{T}$ if there is an algorithm $A$ that computes $f$ for which $\mathcal{T}$ proves (using some fixed recursive coding of algorithms) that $A$ halts on every input.*

**Lemma 1***: [Wa70]*

- *For every ordinal $\alpha < \epsilon_0$, $F_\alpha$, the $\alpha$'th function in the Wainer hierarchy, is provably recursive in $PA$.*

of a language $L$ in a class $C$ cannot be decided in our strong system, then $L$ is very close to members of $C$ (in terms of its worst case complexity function). Focusing on a concrete and well investigated theory, namely $PA$, allows us to strengthen the easiness condition of [KOR87].

We show that the currently known tools for proving independence, are not refined enough to distinguish between our system and the standard Peano Arithmetic. We conclude that, as long as no new proof theoretic method will be employed, proving independence from $PA$ of, say, $P \neq NP$, would imply the existence of an almost polynomial decision algorithm for $SAT$.

We offer a new tool for measuring the distance between a language and a complexity class. We show that it yields precise characterization of provability and that it inhibits some practically jarring aspects of the worst case measure. The new technique enables us to generalize our results to the membership problem of any language in any complexity class.

## 1.3 Outline of our proof

For the sake of clarity let us concentrate on the case of provability relative to Peano Arithmetic. Our results can be modified to handle provability relative to other proof systems ( in particular the ZFC axioms of set theory).

We augment the Peano Arithmetic axioms with the $\Pi_1$ theory of the natural numbers - the set of all $\Pi_1$ formulas that are true in $\mathcal{N}$ (the standard model of arithmetic). [1] Let us denote the resulting theory by $PA_1$ and use $PA$ to denote (non-augmented) Peano Arithmetic. $PA_1$ is not a recursive theory and therefore too strong for being considered as a conceivable formalization of mathematical reasoning. Still, it turns out to be a useful tool for analyzing provability relative to Peano Arithmetic: If a statement $\phi$ is not provable in $PA_1$ then, of course, it is not provable in $PA$. On the other hand, if $\phi$ is a natural mathematical statement for which "$\phi$ is independent of $PA$" is provable in any of the known methods for establishing such results[2], then $\phi$ is independent of $PA_1$ as well!

The next ingredient in our approach comes from the proof theory of Arithmetic. The basic idea behind the Paris-Harrington independence proof may be viewed as a 'fast growing functions' principle. This idea goes back to Kreisel who in [Kr52] proved that, as far as provability in $PA$ goes, recursive functions having very high rate of growth are not provably total. It turns out that the totality of such functions is also independent of $PA_1$. Furthermore, we shall show that, any complexity lower-bound statement that is independent of $PA_1$ can be represented as a statement of that type (i.e. that some meaningful recursive function is total).

The last component we need is a new way of measuring how close is a given language to a complexity class. Given a class $C$ of languages and a language $L$ we define the *approximation rate* of $L$ by $C$. The central step in this work is theorem 4 stating that $PA_1 \nvdash$ "$L \notin C$" if

[1] Our definition of $PA_1$ differs slightly from the common definition. See section 3.1 for details.

[2] We use the term 'a natural mathematical statement' to rule out the conspicuous counter example to our claim: By Godels' theorem, the consistency of $PA$ *is* a true $\Pi_1$ statement that is not provable in $PA$. So far, Godel's type of argument, as well as reductions to this non-provability theorem, have been successful only when applied to self-referential or inherently proof-theoretic statements $\phi$.

## 1.1 Previous Work

Previous work on this issue can be roughly classified into the following categories:

1. Results that are based on self-referential algorithms (or diagonalization). The work of Hartmanis and Hopcroft [HH76] is a central example in this category. Results of this type tend to be easily generalizable to any recursive proof system. On the other hand, as such results can be easily extended to oracle complexity models, they are of inherently limited range. Rather than referring to the complexity of a language they refer to the complexity of a specific algorithm or to the complexity of languages in relativised models of computation.

2. Another line of attack , is to show independence of natural problems relative to a weak (and consequently of limited interest) proof systems. The work of Lipton [Li78] Joseph and Young [JY81, JY85], and Leivant [Le82] can be viewed as taking this approach.

3. More recently, several researchers have succeeded in applying the basic idea behind the Paris-Harrington result to computer theoretic questions. As shall be elaborated below, once a (recursive) function grows extremely fast, its totality cannot be proved in finite set theory. Fortune Leivant and O'donnell [FLO83], and Loebl and Nesetril [LN88], have found problems in the areas of Semantics of Programming Languages and of Data Structures (respectively) that give rise to such functions and, consequently, to non-provable (as far as finitistic methods go) true statements in these fields. Hartmanis [Ha85] and Kurtz O'donnell and Royer [KOR87] similarly prove that, for every recursive proof system, the existence of non-trivial languages that are not provably infinite.

4. Kurtz O'donnell and Royer chose to attack the problem of provability by *strengthening* a natural proof system (rather than taking the common approach of replacing it by a weaker one). They discuss the extension of a proof system by adding to it a non-recursive set of true axioms - the set of true $\Pi_1$ statements. They present a sufficient condition for the the non-provability, relative to such an extended system, of the infiniteness of a language. A novel contribution of [KOR87] is that they consider, for the first time, the reverse direction as well. They prove that if $P \neq NP$ is not provable in such an extended system, then the (deterministic) complexity of $SAT$ satisfies some easiness condition.

## 1.2 Our Results

We wish to understand the issue of independence of natural complexity questions (such as $P \overset{?}{=} NP$) with respect to proof systems that reflect the reasoning process of everyday mathematical investigation. We follow the approach introduced in [KOR87] and consider provability relative to the theory $PA$ augmented by all true $\Pi_1$ statements. We prove a necessary and sufficient condition for the independence of computational complexity questions relative to this strong proof system.

Our sufficient condition is a generalization of the ideas underlying the third of the above-mentioned approaches. Our necessary condition indicates the inherent limitations of independence proofs in the realm of computational complexity. We prove that if the membership

## 1 Introduction

Godel's incompleteness theorem [Go31] has opened a new era in the history of mathematical thought. In the face of an open conjecture we can try to prove it or try to refute it, but Godel has brought to our awareness a third possibility - it may sometimes be the case that, in the framework of what we consider as acceptable mathematical reasoning, the conjecture is neither provable nor is it refutable. In such cases we say that the conjecture is *independent* of the underlying mathematical theory.

Godel's incompleteness theorem states that every formal theory that is recursive, sound and sufficiently expressive, cannot be complete. In other words, once we fix our proof technique, there is always a statement that both it and its negation are not provable in that formal theory.

It took more than thirty years to come up with the first example of a genuine mathematical question that is independent of mathematics (i.e. Set Theory). In [Co63] Cohen proves the independence of the continuum hypothesis. Cohen introduces a special technique for proving independence from the axioms of set theory - the 'forcing' technique. So far, Cohen's forcing is the only available framework for proving such results.

The forcing method has, however, some inherent limitations: There is a class of formulas, called *absolute formulas* [Ku80], for which the forcing method cannot produce independence proofs. In particular, forcing cannot prove the independence (from set theory) of any open question in Number Theory.

It may be justified, when dealing with number theoretic statements, to investigate independence from number theoretic theories. Peano Arithmetic proof system ($PA$) may be considered a reasonable candidate for reflecting the finitistic working tools of a mathematician, yet it is strictly weaker than full Set Theory. This approach was successfully applied by Paris and Harrington [PH77] who, in 1977, proved the independence from Peano Arithmetic of a combinatorial statement concerning finite sets of natural numbers.

The results of Paris and Harrington have raised some hopes that similar independence results could be obtained for interesting statements in Computer Science. Questions concerning computational complexity in the Turing Machine model can usually be coded as questions about natural numbers. It is only natural to wonder whether our apparent helplessness in the face of questions like $P \overset{?}{=} NP$ is not the result of an inherent independence of this statement (rather than an indication to an insufficient understanding of the issue).

The aim of this work is to investigate this rather exotic possibility. The bottom line of our analysis can be roughly stated as follows: In order to prove a lower bound independence result, with the tools currently available to logicians, one would basically have to resolve the original question. In particular, the existence of a proof, by any currently known method, of the independence from $PA$ of the statement "$P \neq NP$" is *equivalent* to the existence of almost-polynomial upper bounds on the deterministic complexity of $NP$. Such an independence result would imply, for an example, the nonexistence of (polynomial) One-Way-Functions. (An analogous result holds for the independence of $P = NP$).

On the other hand we prove the existence of arbitrarily complex languages $L$ for which $PA \nvdash$ "*$L$ is not a regular language*". We also present an explicitly defined language exhibiting such behavior.

# On the Independence of $P$ versus $NP$

(Revised Version)

Shai Ben-David [*] Shai Halevi

Dept. of Computer Science
Technion, Haifa 32000, Israel

**Abstract**

We investigate the possibility that the current failure to resolve basic complexity theoretic questions stems from the independence of these questions with respect to the formal theories underlying our mathematical reasoning.

We show that, any question in the field of computational complexity that is independent of a certain extension of the axioms of Peano Arithmetic, using currntly available techniques, is 'practically insignificant'. This implies that if $P \neq NP$ can be shown to be independent of Peano Arithmetic, using any currently known mathematical paradigm, then $NP$ has extremely-close-to-polynomial deterministic time upper bounds. In particular, in such a case, there is a $DTIME(n^{log^*(n)})$ algorithm that computes $SAT$ correctly on infinitely many huge intervals of input lengths.

We provide a complete characterization of the worst case behavior of languages whose location in the complexity hierarchy is independent (with respect to sufficiently strong proof systems, including Peano Arithmetic). Such languages are, on one hand easily computable for long stretches of inputs, and, on the other hand, they are complex infinitely often. (We also construct an explicit example of such a language).

Our results hold for both the Turing Machine and the Non-Uniform Circuit complexity models.

[*]e-mail: shai@cs.technion.ac.il