On Bounded Nondeterminism and Alternation

Mathias Hauptmann*

May 4, 2016

Abstract

We continue our work on the combination of variants of McCreight and Meyer's Union Theorem with separation results along the line of Paul, Pippenger, Szemeredi, Trotter and S.Gupta. Under assumption NP=PSPACE, we prove a union theorem for the class PSPACE=AP with respect to a particular subfamily $(\tilde{S}_{(i)})$ of alternating machines. This yields a union function F which is nondeterministically computable in time $F(n)^C$ for some constant C. We show that for each problem $L \in$AP, there exists a polynomial time bounded machine $\tilde{S}_{(i)}$ such that $L = L(\tilde{S}_{(i)})$. Moreover, we prove a variant of Gupta's result who showed that DTIME$(t) \subsetneq \Sigma_2(t)$ for time-constructible functions $t(n)$. Our variant of this result holds with respect to the subfamily $(\tilde{S}_{(i)})$ of alternating machines. We show that these two results contradict each other, whence our starting assumption cannot hold.
Keywords: Alternating Turing Machines, Nondeterministic versus Alternating Time Complexity, Union Theorem, Bounded Nondeterminism.

1 Introduction

Let us first recall the two starting points for our previous paper [H16]. The *Union Theorem* of McCreight and Meyer [McCM69] states that whenever (f_i) is a family of functions $f_i \colon \mathbb{N} \to \mathbb{N}$ such that $\lambda i, n.f_i(n)$ is recursive, then there exists a single recursive function F such that DTIME$(F) = \bigcup_i$ DTIME(f_i). Indeed this does not only hold for deterministic time complexity classes but for any Blum complexity measure.

The second starting point is the result of Paul, Pippenger, Szemeredi and Trotter [PPST83] who proved that DLIN $\neq$ NLIN. This proof is based on their remarkable result that deterministic computations can be simulated by Σ_4-computations with a small speedup. Namely, for every time-constructible function $t(n)$, DTIME$(t \log^*(t)) \subseteq \Sigma_4(t)$. Gupta [G96] improved on this result and showed that for each such t, DTIME$(t \log^*(t)) \subseteq \Sigma_2(t)$. Moreover he proved that for every time-constructible t, DTIME$(t) \subsetneq \Sigma_2(t)$. See also the discussion at the end of [H16].

Our approach in [H16] was now the following. We assumed $P = \Sigma_2^p$. The idea was to construct then a union function F such that $P =$ DTIME$(F) = \Sigma_2(F) = \Sigma_2^p$ such as to obtain a contradiction to the result of Gupta. However, a direct construction of such a union function F along the lines of McCreight and Meyer yields a function F which is recursive but not time-constructible. The second idea was then that it might be possible to construct F such that $F(n)$ is computable in time polynomial in $F(n)$, namely by making use of the assumption $P = \Sigma_2^p$. If the function F is, say, deterministically computable in DTIME(F^C), then one might try to use *Padding* in order to show that DTIME$(F) = \Sigma_2(F)$ also implies DTIME$(F^C) = \Sigma_2(F^C)$, which then yields the desired contradiction to Gupta's result. However, for such a *padding construction* to work, it is necessary that the function F satisfies an inequality of the kind $F(n)^C \leq F(n^h)$. In

*Dept. of Computer Science, University of Bonn. e-mail: hauptman@cs.uni-bonn.de

[H16] we called this a *Padding Inequality*. Now again, if we construct a union function directly along the lines of McCreight and Meyer, then the resulting function F would *not satisfy* such a *Padding Inequality*. The third idea is then that a padding inequality might be reachable provided the underlying machines satisfy some additional property. The construction of a union function for polynomial time complexity classes is essentially a diagonalization against all the machines whose running time is *not polynomially bounded*. Now suppose that all the machines under consideration have the following additional property: Whenever a machine violates a given polynomial time bound at some input length (and say both the polynomial and the input length are sufficiently large), then the machine has already violated a similar but somewhat smaller polynomial time bound at a number of smaller input lengths. It turns out in [H16] that such a property can be formalized and the construction of the union function can be adjusted that it will satisfy a Padding Inequality. But this means now that we have switched from a standard enumeration of, say, all alternating machines to a subfamily, namely of machines which actually satisfy this additional property. Then it is of course necessary to show that this does not change the complexity classes under consideration.

We give now an outline how to apply the method from [H16] to the classes NP and AP = PSPACE. We will assume NP = AP. Then we start from a standard enumeration (S_i) of alternating machines and construct a subfamily of machines, containing for each original machine S_i and every integer d a machine $\tilde{S}_{i,d}$. These machines will satisfy the additional property, which is the same as in [H16] and which we call *Property* $[\star]$. We will show that going from a general family (S_i) of alternating machines to a subfamily $(\tilde{S}_{i,d})$ will not change the complexity classes under consideration. Then we construct a union function F such that $\mathrm{NP} = \mathrm{NTIME\tilde{E}}(F) = \mathrm{ATIME\tilde{E}}(F) = \mathrm{AP}$. Here by $\mathrm{NTIME\tilde{E}}(F)$ and $\mathrm{ATIME\tilde{E}}(F)$ we denote the classes with respect to our restricted family of machines. We show that this function F satisfies a *Padding Inequality*. This yields that we also have $\mathrm{NTIME\tilde{E}}(F^C) = \mathrm{ATIME\tilde{E}}(F^C)$. The function F^C is not time-constructible, but it can be computed nondeterministically and co-nondeterministically in time $F(n)^C$. Thus we will show an analogon of Gupta's result, namely that $\mathrm{NTIME\tilde{E}}(t) \subsetneq \mathrm{ATIME\tilde{E}}(t)$ for every function t which is nondeterministically and co-nondeterministically computable in time $t(n)^{1-\epsilon}$ for some $\epsilon > 0$. For such a result to work, we also have to require that the nondeterministic machines in our subclass have *bounded nondeterminism*. It suffices to restrict the number of nondeterministic steps to the squareroot of the running time of the machine, and again this will not change the complexity class NP.

2 Implications from the Assumption NP=AP

We let (S_i) be a standard enumeration of alternating machines. We let Check be the following decision problem:

$$\mathrm{Check} = \left\{(i, n, a, b) |\, \mathrm{time}_{S_i}(n) > an^b\right\}$$

Since AP=PSPACE is closed under complement, NP=AP implies NP=coNP. Via some standard padding argument, this yields that the decision problem Check can be solved by some NTIME∩coNTIME-algorithm which has running time $c_0 \cdot (i \cdot an^b)^{c_0}$ on inputs (i, n, a, b), for some constant c_0. This means that this algorithm is a guessing algorithm which has three possible outputs $0, 1,$"?". For each input x, either there exists no computation path of the algorithm on input x with output 0, or there exists no path with output 1, and furthermore, for each input x there exists at least one path with output either 0 or 1. Since we are not aware of any existing short notation for classes NTIME(t)∩coNTIME(t) from the literature, we will denote them as ZNTIME(t) =NTIME(t)∩coNTIME(t), where ZNTIME stands for zero-error nondeterministic

time. Moreover, abusing notation, we write

$$\text{Check} \in ZNTIME(c_0 \cdot (i \cdot an^b)^{c_0}).$$

In this case, $ZNTIME(c_0 \cdot (i \cdot an^b)^{c_0})$ does not denote a complexity class but just the fact that the problem Check can be solved by such a zero-error nondeterministic algorithm within the time bound $c_0 \cdot (i \cdot an^b)^{c_0})$ for inputs of the form (i, n, a, b).

3 A New Programming System for AP

We start from a standard programming system (also called Gödel numbering) (S_i) of alternating machines. Without loss of generality we assume that this numbering has the following additional property: Whenever S_i is a nondeterministic machine, meaning that it does not have any universal states, then the number of guesses made by machine S_i is already bounded by the squareroot of its running time. As we have described in [H16], in order to ensure that the union function F satisfies the *Padding Inequality* $F(n)^c \leq F(n^h)$, we want to work with a new family of alternating machines such that if $t(n)$ is a running time function of some machine from this family, then $t(n)$ satisfies the following condition which we called *Property* $[\star]$.

Definition 3.1. *[H16] We say function $t(n)$ satisfies Property $[\star]$ with parameters c, d, p if for every $n \geq 2^{c^2}$ and all pairs of integers a, b with $c \leq a \leq b \leq \frac{\log(n)}{c}$, if $t(n) > an^b$, then there exist $\lceil \frac{\log\log(n)}{c} \rceil$ pairwise distinct integers $m_1, \ldots, m_{\lceil \frac{\log\log n}{c} \rceil}$ in the interval*

$$I_{n,d} = \left(n^{1/h}, n^{1/h} \cdot \left(1 + \frac{\log(n)}{n^{1/(h \cdot d)}} \right)^d \right)$$

such that m_l is not an h-power and $t(m_l) > (a-p)m_l^{b-p}$, $l = 1, \ldots, \lceil \frac{\log\log n}{c} \rceil$.

Starting from the family (S_i) of alternating machine, we will construct a new family $(\tilde{S}_{i,d})$ of alternating machines which contains for each machine S_i and every integer d a machine $\tilde{S}_{i,d}$. This new family of machines will have the following properties:

- For every machine index i and every integer d, the machine $\tilde{S}_{i,d}$ satisfies Property $[\star]$ with some parameters $c_{i,d}, d, p_i$.
- For every i and d, the running time of the machine $\tilde{S}_{i,d}$ will be bounded by twice the running time of machine S_i.
- If S_i is a nondeterministic machine, then for each integer d, the machine $\tilde{S}_{i,d}$ is also a nondeterministic machine for which the number $\text{guess}_{\tilde{S}_{i,d}}(n)$ of guesses made by $\tilde{S}_{i,d}$ satisfies $\text{guess}_{\tilde{S}_{i,d}}(n) \leq c_{i,d} \cdot \sqrt{\text{time}_{\tilde{S}_{i,d}}(n)}$.
- If the running time function $\text{time}_{S_i}(n)$ already satisfies Property $[\star]$ with parameters $\frac{c_{i,d}}{2}, d, \frac{p_i}{2}$, then $L(\tilde{S}_{i,d}) = L(S_i)$.

We shall work with the associated nondeterministic and alternating time complexity classes $\text{NTIM}\tilde{\text{E}}_{\surd^-}(t)$ and $\text{ATIM}\tilde{\text{E}}(t)$, which are defined as follows:

- L is in $\text{ATIM}\tilde{\text{E}}(t)$ if there exists some alternating machine $\tilde{S}_{i,d}$ such that $L = L(\tilde{S}_{i,d})$ and $\text{time}_{\tilde{S}_{i,d}}(n) = O(t(n))$.
- L is in $\text{NTIM}\tilde{\text{E}}_{\surd^-}(t)$ if there exists some nondeterministic machine $\tilde{S}_{i,d}$ such that $L = L(\tilde{S}_{i,d})$ and $\text{time}_{\tilde{S}_{i,d}}(n) = O(t(n))$. Especially, since $\tilde{S}_{i,d}$ is a nondeterministic machine, we have $\text{guess}_{i,d}(n) \leq c_{i,d} \cdot \sqrt{\text{time}_{i,d}(n)} = O(\sqrt{t(n)})$.

3.1 Construction of Machines $\tilde{S}_{i,d}$

Before we describe the construction of the new family $(\tilde{S}_{i,d})$ of alternating machines, let us give a characterization of Property $[\star]$ which we have already used in [H16]. This will be useful in the construction of the machines $\tilde{S}_{i,d}$. Intuitively, the machine $\tilde{S}_{i,d}$ will just perform a step-by-step simulation of the machine S_i, but at the same time it will check if it is allowed to continue this simulation, such as to satisfy Property $[\star]$. In the following we just give a name to the property which machine $\tilde{S}_{i,d}$ has to check.

Here we give first the general formulation for functions $t\colon \mathbb{N} \to \mathbb{N}$.

Definition 3.2.
The predicate $P \subseteq (\mathbb{N} \to \mathbb{N}) \times \mathbb{N}^6$ is defined as follows. $P(t,c,d,p,n,a,b)$ holds if

- $n < 2^{c^2}$ *or* $a < c$ *or* $b > \frac{\log(n)}{c}$ *or*
- $n \geq 2^{c^2}$ *and there exist pairwise distinct integers* $m_1, \ldots, m_{\lceil \frac{\log\log(n)}{c} \rceil} \in I_{n,d}$ *which are not h-powers and such that* $t(m_l) > (a-p)m_l^{b-p}$ *and* $P(t,c,d,m_l,a-p,b-p)$, $l = 1, \ldots, \lceil \frac{\log\log(n)}{c} \rceil$.

We observe that $t\colon \mathbb{N} \to \mathbb{N}$ satisfies Property $[\star]$ with parameters c,d,p iff for all $n \geq 2^{c^2}$ and $c \leq a \leq b \leq \frac{\log(n)}{c}$, $t(n) > an^b$ implies $P(t,c,d,p,n,a,b)$

Now we can describe the construction of the machines $\tilde{S}_{i,d}$. In the case of P versus Σ_2^p in [H16], the machines $\tilde{S}_{i,d}$ just performed a simulation of the machines S_i and in parallel checked (deterministically) if the property $P(\,)$ was satisfied such as to continue the simulation. Here the situation is slightly different. Since we work under the assumption NP=AP, this especially means that we do not have a deterministic algorithm at hand to check the predicate $P(\,)$. Instead, we are given now an NTIME$\cap$coNTIME-algorithm for the predicate $P(\,)$, and the machine $\tilde{S}_{i,d}$ is supposed to simulate machine S_i, which may now be any alternating machine. This means that checking the predicate $P(\,)$ will cause the machine $\tilde{S}_{i,d}$ to perform some additional alternations.

We construct the machine $\tilde{S}_{i,d}$ as follows: Given an input x of length n, consider two consecutive pairs of integers $(a',b'),(a,b)$ with $c_{i,d} \leq a' \leq b' \leq \frac{\log(n)}{c_{i,d}}$ and $c_{i,d} \leq a \leq b \leq \frac{\log(n)}{c_{i,d}}$ (consecutive with respect to the order $b' < b$ or $(b' = b$ and $a' < a)$). Within the time interval $\left(a'n^{b'}, an^b\right]$ the machine $\tilde{S}_{i,d}$ uses the first half of the time in this interval to check if the predicate $P(\,)$ holds, i.e. if it is allowed to continue the computation for more than an^b steps. The remaining half of the time within this interval is used to simulate computation steps of the machine S_i on input x.

Thus, as long as the machine $\tilde{S}_{i,d}$ is allowed to continue its computation, it will make twice as many computation steps as S_i on the same input x. We are now ready to give the definition of the predicate P for machines $\tilde{S}_{i,d}$.

Definition 3.3. *(Predicate P for machines $\tilde{S}_{i,d}$)*
$P(i,d,n,a,b)$ holds if ($n \geq 2^{c_{i,d}^2}$ and $c_{i,d} \leq a \leq b \leq \frac{\log n}{c_{i,d}}$) implies that there exist pairwise distinct non-h-power integers $m_1, \ldots, m_{\lceil \frac{\log\log(n)}{c_{i,d}} \rceil} \in I_{n,d}$ such that for $l = 1, \ldots, \lceil \frac{\log\log(n)}{c_{i,d}} \rceil$, $time_{S_i}(m_l) > \frac{1}{2}(a-p_i)m_l^{b-p_i}$ and for all pairs of integers $(\alpha,\beta) \leq_{lex} (a-p_i, b-p_i)$, $P(i,d,m_l,\alpha,\beta)$ holds.

On input x of length n, if $n \leq 2^{c_{i,d}^2}$, then $\tilde{S}_{i,d}$ just simulates S_i on input x and makes in total twice as many computation steps as S_i. On the other hand, if $n > 2^{c_{i,d}^2}$, then $\tilde{S}_{i,d}$ does the following: Let $L = L_{i,d}(n)$ denote the number of pairs of integers (a,b) with $c_{i,d} \leq a \leq b \leq \frac{\log(n)}{c_{i,d}}$,

and let $(a_l, b_l), l = 1, \ldots, L$ be these pairs in the lexicographic order described above. The associated intervals are $T_0 = \left(0, a_1 n^{b_1}\right]$, $T_l = \left(a_l n^{b_l}, a_{l+1} n^{b_{l+1}}\right], l < L$ and $T_L = \left(a_L n^{b_L}, \infty\right)$. Now $\tilde{S}_{i,d}$ uses the first half of the computation steps within each interval $T_l, l < L$ to simulate computation steps of S_i on input x. The second half of the time within interval T_l is used to check if the predicate $P(i, d, n, a_{l+1}, b_{l+1})$ holds. If within some interval T_l the computation of S_i on input x terminates, then $\tilde{S}_{i,d}$ terminates as well, with the same output (accept/reject). If within some interval T_l, the computation of S_i does not yet terminate but the predicate $P(i, d, n, a_{l+1}, b_{l+1})$ does not hold, $\tilde{S}_{i,d}$ also completes this interval and then terminates and rejects. Otherwise, it continues within the next inerval T_{l+1}. If the computation reaches the interval T_L, then it just continues to simulate the computation of S_i and does not check the predicate $P(\,)$ anymore. The computation of $\tilde{S}_{i,d}$ is organized in such a way that while being in an interval $T_l, l < L$, it always makes precisely twice as many computation steps as S_i. This will be particularly important when we show in Lemma 3.2 that for those machines S_i which already satisfy Property [$\star$], the machine $\tilde{S}_{i,d}$ will compute the same as machine S_i, i.e. $L(\tilde{S}_{i,d}) = L(S_i)$. We give a pseudo-code description of the machine $\tilde{S}_{i,d}$.

Machine $\tilde{S}_{i,d}$
Input: x of length n

If $n < 2^{c^2_{i,d}}$, simulate the computation of machine S_i on input x
and make in total twice as many computation steps as S_i.

If $n \geq 2^{c^2_{i,d}}$
For $l = 0, \ldots L-1$ (where $L = L_{i,d}(n)$)
Use the first half of the interval T_l to continue
the simulation of computation of S_i on input x
If the computation of S_i terminates after t steps in T_l,
then $\tilde{S}_{i,d}$ performs t additional dummy steps
and then also terminates.
Otherwise, use the second half of the interval T_l to check
if $P(i, d, n, a_{l+1}, b_{l+1})$ holds and fill
the rest of T_l with dummy steps.
If $P(i, d, n, a_{l+1}, b_{l+1})$ does not hold, stop and reject.
/$\star$ Now we are in the interval $T_L = \left(a_L n^{b_L}, \infty\right)$ $\star$/
Continue the simulation of computation of S_i on input x.

We will now show that for each $l < L$, one half of the interval T_l suffices to check if the predicate $P(i, d, n, a_{l+1}, b_{l+1})$ holds.

Lemma 3.1. *Predicate $P(i, d, n, a, b)$ can be checked in $NTIME(n^{b-p_i})$, where $p_i = \lceil \frac{i}{2} \rceil$.*

Proof. In order to check if $P(i, d, n, a, b)$ holds, it suffices to solve the instances (i, m, a', b') of the problem Check for $a' \leq b' \leq b - p_i$ and $m \in R_{i,d}(n)$, where the set $R_{i,d}(n)$ - which we call the set of relevant input lengths for i, d, n - is defined as follows: $R_{i,d}(n) = \bigcup_l R^l_{i,d}(n)$ with

$$\begin{aligned} R^1_{i,d}(n) &= I_{n,d} = \left(n^{1/h},\, n^{1/h}\left(1 + \tfrac{\log(n)}{n^{1/(dh)}}\right)^d\right) \\ R^{l+1}_{i,d}(n) &= \bigcup_{m \in R^l_{i,d}(n), m \geq 2^{c^2_{i,d}}} I_{m,d} \end{aligned}$$

We give a bound on the cardinality of the set $R_{i,d}(n)$ as follows. We have $R^l_{i,d}(n) \subseteq [L_l, R_l]$,

where $L_l = n^{1/h^l}$ and

$$\begin{aligned} R_l &= 2^{1+\frac{1}{h}+\ldots+\frac{1}{h^{l-1}}} \cdot n^{1/h^l} = 2^{\frac{1-1/h^l}{1-1/h}} \cdot n^{1/h^l} \\ &= 2^{(1-\frac{1}{h^l})\cdot\frac{h}{h-1}} \cdot n^{1/h^l} \leq 2^{\frac{h}{h-1}} \cdot n^{1/h^l} \end{aligned}$$

Moreover, $R_l < 2^{c_{i,d}^2}$ implies that the level set $R_{i,d}^{l+1}(n)$ is empty. Now the condition $R_l < 2^{c_{i,d}^2}$ follows - by taking logarithms and combining it with the previous inequality - from

$$\begin{aligned} & \frac{1}{h^l} \cdot \log(n) + \frac{h}{h-1} < c_{i,d}^2 \\ \Leftrightarrow \quad & h^l > \frac{1}{c_{i,d}^2 - \frac{h}{h-1}} \cdot \log(n) \\ \Leftrightarrow \quad & l > \frac{1}{\log(h)} \left(\log\log(n) - \gamma_{i,d}\right) \end{aligned}$$

where $\gamma_{i,d}$ is a constant that only depends on i, d and the global constant h. Thus we obtain

$$|R_{i,d}(n)| \leq n^{1/h} \cdot \log\log(n)$$

Thus $P(i,d,n,a,b)$ can be decided nondeterministically by solving at most $n^{1/h} \cdot \log\log(n) \leq n^{2/h}$ instances of the problem Check of the form (i, m, a', b') with $a' \leq a - p_i, b' \leq b - p_i, m \leq 2 \cdot n^{1/h}$, which can be done nondeterministically in time $n^{2/h} \cdot c \cdot (i(a-p_i)2n^{(1/h)\cdot(b-p_i)})^c \leq n^{b-p_i}$. □

Remark. This lemma also yields $P(i,d,n,a,b) \in ZNTIME(n^{b-p_i})$. Moreover, if S_i is a nondeterministic machine, then for every d, the machine $\tilde{S}_{i,d}$ is nondeterministic as well.

Lemma 3.2. *(Properties of the machines $\tilde{S}_{i,d}$)*
Machine $\tilde{S}_{i,d}$ satisfies Property $[\star]$ with parameters $c_{i,d}, d, p_i$. Furthermore, $time_{\tilde{S}_{i,d}}(n) \leq 2 \cdot time_{S_i}(n)$. If the machine S_i already satisfies Property $[\star]$ with parameters $\frac{c_{i,d}}{2}, d, \frac{p_i}{2}$, then $L(\tilde{S}_{i,d}) = L(S_i)$.

Corollary 1. *Under assumption $NP = AP$ we have $NP = N\tilde{P}_{\surd} = A\tilde{P} = AP$.*

Proof. (of Lemma 3.2) In order to show that $\tilde{S}_{i,d}$ has Property $[\star]$ with parameters $c_{i,d}, d, p_i$, we first prove the following

> *Auxiliary Claim*: For every n, for all pairs (a,b) of integers with $a \leq b$, ($\text{time}_{S_i}(n) > \frac{1}{2}an^b$ and $\forall(\alpha,\beta) \leq_{lex} (a,b)\ P(i,d,n,\alpha,\beta)$) implies that $\text{time}_{\tilde{S}_{i,d}}(n) > an^b$.

Proof of the Auxiliary Claim. For $n < 2^{c_{i,d}^2}$, we have $\text{time}_{\tilde{S}_{i,d}}(n) = 2\text{time}_{S_i}(n)$, and thus the claim holds. On the other hand, for $n \geq 2^{c_{i,d}^2}$, it follows directly from the construction of the machine $\tilde{S}_{i,d}$ that if $\text{time}_{S_i}(n) > \frac{1}{2}an^b$ and the property $P(i,d,n,\alpha,\beta)$ holds for all $(\alpha,\beta) \leq_{lex} (a,b)$, then this implies $\text{time}_{\tilde{S}_{i,d}} > an^b$.

Now we use the Auxiliary Claim in order to show that $\tilde{S}_{i,d}$ satisfies Property $[\star]$ with parameters $c_{i,d}, d, p_i$. Thus suppose that $\text{time}_{\tilde{S}_{i,d}}(n) > an^b$ and $c_{i,d} \leq a \leq b \leq \frac{\log(n)}{c_{i,d}}$. Since $\text{time}_{S_{i,d}}(n) \leq 2 \cdot \text{time}_{S_i}(n)$, this implies that $\text{time}_{S_i}(n) > \frac{1}{2}an^b$, and moreover, $P(i,d,n,a,b)$ holds. This means that there exist pairwise distinct non-h-power integers $m_1, \ldots, m_{\lceil \frac{\log\log n}{c_{i,d}} \rceil} \in I_{n,d}$ such that $\text{time}_{S_i}(m_l) > \frac{1}{2}(a-p_i)m_l^{b-p_i}$ and $\forall(\alpha,\beta) \leq_{lex} (a-p_i, b-p_i)\ P(i,d,m_l,\alpha,\beta)$ holds for $l = 1, \ldots, \lceil \frac{\log\log n}{c_{i,d}} \rceil$. Now the Auxiliary Claim yields that $\text{time}_{\tilde{S}_{i,d}}(m_l) > (a-p_i)m_l^{b-p_i}$ for $l = 1, \ldots, \frac{\log\log n}{c_{i,d}}$, which means that $\tilde{S}_{i,d}$ satisfies Property $[\star]$ with parameters $c_{i,d}, d, p_i$.

Now suppose that S_i satisfies Property $[\star]$ with parameters $\frac{c_{i,d}}{2}, d, \frac{p_i}{2}$. We want to show that $L(\tilde{S}_{i,d}) = L(S_i)$, i.e. that for every input x, $\tilde{S}_{i,d}(x)$ completely simulates $S_i(x)$. This holds in case when $|x| < 2^{c_{i,d}^2}$. Now suppose $n = |x| \geq 2^{c_{i,d}^2}$. If $\tilde{S}_{i,d}(x) \neq S_i(x)$, then there exists some $l \in \{0, \ldots, L_{i,d}(n)\}$ such that $\text{time}_{S_i}(x) > \frac{1}{2} a_{l+1} n^{b_{l+1}}$, $P(i, d, n, a_{l+1}, b_{l+1})$ does not hold and $P(i, d, n, a_{l'}, b_{l'})$ holds for all $0 \leq l' < l$. Since $n \geq 2^{c_{i,d}^2} > 2^{(\frac{1}{2} c_{i,d})^2}$ and $\frac{c_{i,d}}{2} \leq \frac{a_{l+1}}{2} \leq b_{l+1} \leq \frac{\log(n)}{c_{i,d}/2}$, we obtain that there exist pairwise distinct integers $m_1, \ldots, m_{\frac{\log \log n}{c_{i,d}/2}}$ in $I_{n,d}$ which are not h-powers and such that $\text{time}_{S_i}(m_l) > (\frac{a_{l+1}}{2} - \frac{p_i}{2}) m_l^{b_i - p_i/2}$. If we assume n to be minimal with this property, then we have $\text{time}_{\tilde{S}_{i,d}}(m_l) > (a_{l+1} - p_i) m_l^{b_i - p_i/2} \geq (a_{l+1} - p_i) m_l^{b_i - p_i}$. Now the construction of $\tilde{S}_{i,d}$ implies that $P(i, d, m_l, \alpha, \beta)$ holds for all $(\alpha, \beta) \leq_{lex} (a - p_i, b - p_i)$, and, using the Auxiliary Claim, this yields that $\tilde{S}_{i,d}(x)$ will continue and go to the $(l+1)$st iteration of the for loop, a contradiction. Altogether we have shown that $L(\tilde{S}_{i,d}) = L(S_i)$. □

4 The Union Function $F(n)$

Now we will briefly describe the construction of the union function. The construction is the same as in [H16]. The function F is constructed in stages. In stage n, the function value $F(n)$ is defined. The whole construction is a diagonalization against all those machines $\tilde{S}_{i,d}$ which violate a pre-defined polynomial upper bound for the running time of the machine, namely a variant of the original construction from [McCM69]. The construction is based on the notion of *guesses*. A *guess* is just a pair $(\tilde{S}_{i,d}, b)$, where $\tilde{S}_{i,d}$ is a machine from our family of alternating machines and b is a positive integer. The guess *holds in stage* n if $\text{time}_{\tilde{S}_{i,d}}(n) \leq b \cdot n^b$, otherwise the guess is *violated in stage* n. Our union function F is supposed to have the following property.

$$\text{N}\tilde{\text{P}}_{\surd^-} = \text{NTIM}\tilde{\text{E}}_{\surd^-}(F) \qquad = \text{ATIM}\tilde{\text{E}}(F) \qquad = \text{AP} \tag{1}$$

Moreover, the function $F(n)$ will be computable in $ZNTIME(F(n)^C)$ for some constant C. Using *Padding*, we want to show then that $\text{NTIM}\tilde{\text{E}}_{\surd^-}(F) = \text{ATIM}\tilde{\text{E}}(F)$ also implies $\text{NTIM}\tilde{\text{E}}_{\surd^-}(F^{C^2}) = \text{ATIM}\tilde{\text{E}}(F^{C^2})$. As we shall see below, for such a padding construction to work, we will also have to satisfy the following additional requirement:

$$\text{N}\tilde{\text{P}}_{\surd^-} = \text{NTIM}\tilde{\text{E}}_{\surd^-}(F(n+1)) = \text{ATIM}\tilde{\text{E}}(F(n+1)) = \text{AP} \tag{2}$$

While (1) is the s property of a standard union function in this context, property (2) will be used when we assign to each given problem $L \in \text{ATIM}\tilde{\text{E}}(F^{C^2})$ an associated polynomially padded version L'. While it turns out to be easy to guarantee that $L' \in \text{ATIME}(F)$, showing that we also have $L' \in \text{ATIM}\tilde{\text{E}}(F)$ will require making use of property (2).

We arrange the machines $\tilde{S}_{i,d}$ in a list and let $\tilde{S}_{(j)}$ denote the j^{th} machine in this order. This is done in such a way that from j we can efficiently compute i, d such that $\tilde{S}_{i,d} = \tilde{S}_{(j)}$ and furthermore, $c_{i,d} \leq j$. During the construction of the union function F, we maintain a list $\mathcal{L}$ of guesses. In stage n of the construction, the function value $F(n)$ will be determined. In the stage n, guesses for the first $\log^*(n)$ machines from the list will be taken into account. The overall strategy is to select a lexicographically smallest violated guess, to diagonalize against the associated machine and then to replace this guess by one with a higher value b_j. In order to satisfy both (1) and (2), we will maintain for each machine $\tilde{S}_{(j)}$ two guesses. In stage n, one is used for diagonalizing against violations at input length n and one for input length $n-1$. We will denote these guesses as $(\tilde{S}_{(j)}, b_{j,1}, 1)$ and $(\tilde{S}_{(j)}, b_{j,2}, 2)$. In stage n, the first of the two guesses, $(\tilde{S}_{(j)}, b_{j,1}, 1)$, will be checked for violation at length n (i.e. if $\text{time}_{\tilde{S}_{(j)}}(n) \leq b_{j,1} n^{b_{j,1}}$ is

violated), and the second guess $(\tilde{S}_{(j)}, b_{j,2}, 2)$ will be checked for violation at length $n-1$, i.e. it will be checked if $\text{time}_{\tilde{S}_{(j)}}(n-1) \leq b_{j,2} n^{b_{j,2}}$ holds.

We let $\mathcal{L}_n$ denote this list at the beginning of stage n of the construction. Thus the invariant will be that $\mathcal{L}_n$ will contain guesses $(\tilde{S}_{(j)}, b_{j,1}, 1)$ and $(\tilde{S}_{(j)}, b_{j,2}, 2)$ for $j = 1, \ldots, \log^*(n)$ with $b_j \leq \log^*(n)$. The following notations turn out to be useful, cf. [H16]. We let $\log^*(n) = \min\{t | 2^{2^{\cdots^2}}]_t \geq n\}$, the minimum height of a tower of 2's which is $\geq n$. For $t \in \mathbb{N}$ we let I_t be the set of integers $n \in \mathbb{N}$ for which $\log^*(n) = t$. This means that $I_t = [\delta_{t-1}+1, \delta_t)$ with $\delta_0 = 0, \delta_1 = 2, \delta_{t+1} = 2^{\delta_t}, t \geq 1$.

In the construction of the union function, we consider two cases. If n is an h-power, i.e. $n = n'^h$ for some integer n', then from the list $\mathcal{L}_n$ the lexicographically smallest violated guess $(\tilde{S}_{(j)}, b_{j,l}, l)$ is selected (first ordered by the value b_j, then by l and then by j). If a guess $(\tilde{S}_{(j)}, b_{j,1}, 1)$ is selected, then we define $F(n) := n^{b_{j,1}}$, replace the guess $(\tilde{S}_{(j)}, b_{j,1}, 1)$ by $(\tilde{S}_{(j)}, \log^* n, 1)$ and continue. If a guess $(\tilde{S}_{(j)}, b_{j,2}, 2)$ is selected, then we define $F(n) := (n-1)^{b_{j,2}}$, replace the guess $(\tilde{S}_{(j)}, b_{j,2}, 2)$ by $(\tilde{S}_{(j)}, \log^* n, 2)$ and continue.

If n is not an h-power, we proceed slightly differently. We consider the following set of guesses: all the guesses from $\mathcal{L}_n$ and additionally guesses of the form $(\tilde{S}_{(j)}, b_{j,l} - p_{(i)}, l)$, where $(\tilde{S}_{(j)}, b_{j,l}, l)$ is a guess in $\mathcal{L}_n$ and within the interval I_t with $\log^* n = t$ the guess $(\tilde{S}_{(j)}, b_{j,l} - p_{(i)}, l)$ has not yet been selected. In order to keep track of this, we let $\sigma = (\sigma_{1,1}, \ldots, \sigma_{t,2})$ with $\sigma_{j,l} = 0$ iff the machine $\tilde{S}_{(j)}$ has not yet been selected within a guess of the form $(\tilde{S}_{(j)}, b_{j,l} - p_{(j)}, l)$ within the interval I_t, and $\sigma_{j,l} = 1$ otherwise.

Now in such a stage n we select the smallest violated guess from this set with respect to the following order: first ordered by the value $b_{j,l}$, then by the second entry of the guess ($b_{j,l}$ or $b_{j,l} - p_{(j)}$), then by $l \in \{1,2\}$ and then by j. If a guess $(\tilde{S}_{(j)}, b_{j,1} - p_{(j)}, 1)$ is selected, we set $F(n) := n^{b_j - p_{(j)}}$ and $\sigma_{j,1} := 1$. If a guess $(\tilde{S}_{(j)}, b_{j,2} - p_{(j)}, 2)$ is selected, we set $F(n) := (n-1)^{b_j - p_{(j)}}$ and $\sigma_{j,2} := 1$. If a guess $(\tilde{S}_{(j)}, b_{j,1}, 1)$ is selected, we set $F(n) := n^{b_{j,1}}$. If a guess $(\tilde{S}_{(j)}, b_{j,2}, 2)$ is selected, we set $F(n) := (n-1)^{b_{j,2}}$. In both cases, the guess $(\tilde{S}_{(j)}, b_{j,l}, l)$ being selected is replaced by $(\tilde{S}_{(j)}, \log^* n, l)$ in the list $\mathcal{L}$.

Finally, if $n = \delta_t$ is the last stage within the interval I_t, then two new guesses $(\tilde{S}_{(t+1)}, t+1, l), l = 1, 2$ are added to the list.

Initially, in stage 1, we set $\mathcal{L} = \{(\tilde{S}_{(1)}, 1, 1), (\tilde{S}_{(1)}, 1, 2)\}$, $\sigma_{1,1} := 1$ and $F(1) := 1$. Moreover, at the beginning of every stage $n = \delta_{t-1}+1$, i.e. the first stage within the interval I_t, we initialize the vector $\sigma = (\sigma_{1,1}, \ldots, \sigma_{t,2})$ as $(0, \ldots, 0)$ and then proceed with stage n as described above.

For more information about the construction as well as for proofs of the following results (which are identical to those in the case of P versus Σ_2^p) we refer to [H16].

Lemma 4.1. *We have* $NP = N\tilde{P}_{\surd\!-} = NTIM\tilde{E}_{\surd\!-}(F) = ATIM\tilde{E}(F) = A\tilde{P} = AP$ *and* $NTIM\tilde{E}_{\surd\!-}(F) = NTIM\tilde{E}_{\surd\!-}(F(n+1)) = ATIM\tilde{E}(F(n+1)) = A\tilde{P} = AP$.

Proof. Since in the construction of the union function, guesses are added and replaced with increasing b-value, F majorizes every polynomial. On the other hand, if the running time of a machine $\tilde{S}_{i,d}$ violates every polynomial bound infinitely often, then in the construction of the union function, machine $\tilde{S}_{i,d}$ will be selected infinitely often within a guess $(\tilde{S}_{i,d}, b, 1)$ and infinitely often within a guess $(\tilde{S}_{i,d}, b, 2)$, both with increasing values of b. Thus in that case, neither $\text{time}_{\tilde{S}_{i,d}}(n) = O(F(n))$ nor $\text{time}_{\tilde{S}_{i,d}}(n) = O(F(n+1))$. For a more detailed version of the proof we refer to [H16]. □

Lemma 4.2. *There exists a constant C such that $F(n)$ can be computed in $ZNTIME(F(n)^C)$.*

Proof. We define the functions $F_b(n)$ in the same way as in [H16]. Thus, $F_b(n)$ is the function which is computed when we replace all the guesses $b_j, b_j - p_{(j)}$ which occur in the computation

of $F(n)$ by guesses $\min\{b_j, b+1\}$ and $\min\{b_j - p_{(j)}, b+1\}$ respectively. Functions $F(n)$ and $F_b(n)$ are related as follows: If $F(n) = n^a$ for some integer a, then $F_b(n) = F(n)$ for all $b \geq 3a$.

The only difference is that in the case of P versus Σ_2^p in [H16], the functions $F_b(n)$ could be computed deterministically in time $F_b(n)^{O(1)}$. Since here we work under the assumption $NP = AP$, the functions $F_b(n)$ are now computable in time $F_b(n)^{O(1)}$ by what we called *zero-error nondeterministic machines*, i.e. in $ZNTIME(F_b(n)^{O(1)})$. This means that there exists a nondeterministic algorithm which has running time $F_b(n)^{O(1)}$ on every computation path and either returns "?" or some output, which is then the function value $F_b(n)$. For each n, there is at least one computation path of this algorithm which returns the value $F_b(n)$.

Now we can construct a nondeterministic algorithm for computing $F(n)$ as follows: For $b = 1, 2, \ldots$ this algorithm runs the algorithm for computing the function value $F_b(n)$, until it finds some b such that $F_b(n) = n^a, b \geq 3a$. Then it returns n^a, which is then the correct function value $F(n)$. Whenever some of the computations of function values $F_b(n)$ returns "?", the whole algorithm stops and also returns "?". □

Lemma 4.3. *(Padding Inequality)*
For every h-power n, $F(n^{1/h})^C \leq F(n)$, where C is the constant from Lemma 4.2.

Sketch of Proof. Suppose that n is an h-power such that $F(n^{1/h})^C > F(n)$. Suppose in stage $n^{1/h}$ of the construction of the union function F, some guess of the form $(\tilde{S}_{(i)}, b_i, l_i)$ or $(\tilde{S}_{(i)}, b_i - p_{(i)}, l_i)$ is violated and selected, and in stage n a guess of the form $(\tilde{S}_{(j)}, b_j, l_j)$ is violated and selected. Note that since n is an h-power, in stage n none of the guesses from the extended list of the form $(\tilde{S}_{(j)}, b_j - p_{(j)})$ is taken into account. Now it is straight forward to see that in each of the possible cases, this implies that $b_i > b_j$. But then the guess $(\tilde{S}_{(j)}, b_j, l_j)$ was already contained in the list $\mathcal{L}_{n^{1/h}}$ in stage $n^{1/h}$ of the construction. Since it is violated at input length n, Property $[\star]$ yields that the guess $(\tilde{S}_{(j)}, b_j - p_j, l_j)$ is violated sufficiently often within the stages in the interval $I_{n,d}$, i.e. in stages $n^{1/h} + 1, \ldots$ such that eventually this guess would have been selected earlier, a contradiction. For further details we refer to [H16]. □

Lemma 4.4. *(Padding Lemma)*
$NTIM\tilde{E}_{\sqrt{}}(F) = ATIM\tilde{E}(F)$ implies $NTIM\tilde{E}_{\sqrt{}}(F^{C^2}) = ATIM\tilde{E}(F^{C^2})$.

In order to prove the Padding Lemma, we will first show that the Property $[\star]$ satisfies certain closure properties.

Lemma 4.5. *(Closure Properties of Property $[\star]$)*
Let $t\colon \mathbb{N} \to \mathbb{N}$ be a function that satisfies Property $[\star]$ with parameters c, d, p.

(a) For every $k \in \mathbb{N}$ the function t^k satisfies Property $[\star]$ with some parameters c_k, d_k, p_k.

(b) Let the function $T(n)$ be defined as

$$\begin{aligned} T(n^{h^2} - 1) &= t(n) \text{ for all } n, \\ T(m) &= m^\gamma \text{ for some constant } \gamma \text{ otherwise} \end{aligned}$$

Then the function $T(n)$ satisfies Property $[\star]$ with some parameters c_T, d_T, p_T

Proof. Both the proof of (a) and (b) are technical yet straight forward. In this extended abstract we only give the proof of (b). Suppose that we have already chosen the parameters c_T, d_T, p_T. Suppose that $T(N) > aN^b$, where $c_T \leq a \leq b \leq \frac{\log(N)}{c_T}$. If we choose c_T sufficiently large compared to the constant γ, we can conclude that this implies that N is of the form $N = n^{h^2} - 1$

and $T(N) = T(n^{h^2}-1) = t(n) > a(n^{h^2}-1)^b \geq an^{h^2(b-1)}$. We want to assure that this implies $c_t \leq a \leq h^2 \cdot (b-1) \leq \frac{\log(n)}{c_t}$. Since

$$h^2(b-1) \leq h^2 b \leq h^2 \frac{\log(N)}{c_T} = h^2 \frac{\log(n^{h^2}-1)}{c_T} \leq h^2 \cdot \frac{h^2 \cdot \log(n)}{c_T},$$

it is sufficient to require that

$$c_t \leq c_T,\ \frac{h^4 \cdot \log(n)}{c_T} \leq \frac{\log(n)}{c_t} \ \equiv\ h^4 \cdot c_t \leq c_T \tag{3}$$

Moreover, in order to apply Property $[\star]$ for the function t, we have to assure that $N \geq 2^{c_T^2}$ implies $n \geq 2^{c_t^2}$. We have $2^{c_T^2} \leq N = n^{h^2} - 1 \leq n^{h^2}$, thus it suffices to require that

$$c_T \geq \frac{c_t}{h} \tag{4}$$

It is clear that (3) implies (4). We assume now that (3) holds. Then Property $[\star]$ for the function t yields that there exist non h-power integers

$$m_1, \ldots m_{\lceil \frac{\log\log n}{c_t} \rceil} \in I_{n,d} \text{ with } t(m_l) > (a-p_t) m_l^{h^2(b-1)-p_t}, l = 1, \ldots \lceil \frac{\log\log n}{c_t} \rceil,$$

which means

$$T(m_l^{h^2}-1) \ >\ (a-p_t) m_l^{h^2(b-1)-p_t},\ l = 1, \ldots \lceil \frac{\log\log n}{c_t} \rceil$$

Thus we want to assure that $(a-p_t) m_l^{h^2(b-1)-p_t} \geq (a-p_T)(m_l^{h^2}-1)^{b-p_T}$. For this purpose, it is sufficient to require that

$$h^2(b-1) - p_t \geq h^2(b-p_T),\ \text{ which is equivalent to } h^2 + p_t \leq h^2 p_T. \tag{5}$$

We can satisfy (5) by choosing $p_T \geq p_t$. Moreover we have to assure that sufficiently many integers $M_l = m_l^{h^2} - 1$ are within the interval I_{N,d_T}. Since the implication $m_l \in I_{n,d_t} \Rightarrow M_l \in I_{N,d_T}$ does not hold, we take the following approach. We require the number of integers $m_l \in I_{n,d_t}$ be sufficiently large compared to the required number of integers $M_l \in I_{N,d_T}$. By definition, we have

$$I_{N,d_T} \ =\ \left(N^{1/h},\ N^{1/h} \cdot \left(1 + \frac{\log(N)}{N^{1/(d_T \cdot h)}}\right)^{d_T} \right) \ =: (L_{N,d_T}, R_{N,d_T})$$

and

$$I_{n,d_t} \ =\ \left(n^{1/h},\ n^{1/h} \cdot \left(1 + \frac{\log(n)}{n^{1/(d_t \cdot h)}}\right)^{d_t} \right) \ =: (L_{n,d_t}, R_{n,d_t})$$

We require now that

$$\left\lceil \frac{\log\log(n)}{c_t} \right\rceil \geq 3 \cdot \left\lceil \frac{\log\log(N)}{c_T} \right\rceil \tag{6}$$

Then it is sufficient to replace the requirement $\{m^{h^2}-1 | m \in I_{n,d_t}\} \subseteq I_{N,d_T}$ (which we cannot satisfy) by the following weaker version:

$$L_{n,d_t} + 1 < m < R_{n,d_t} - \frac{\log\log(n)}{c_t} \ \Rightarrow\ M = m^{h^2} - 1 \in I_{N,d_T} \tag{7}$$

We observe that the first part of the implication in (7) is satisfied. Namely, if $m > n^{1/h} + 1$, then

$$M = m^{h^2} - 1 > (n^{1/h}+1)^{h^2} - 1 > N^{1/h} = (n^{h^2}-1)^{1/h} \ \equiv\ ((n^{1/h}+1)^{h^2} - 1)^h > n^{h^2} - 1$$

and this last inequality holds. For the second part, suppose that

$$m \;<\; n^{1/h}\left(1+\frac{\log(n)}{n^{1/(d_t\cdot h)}}\right)^{d_t} - \frac{\log\log(n)}{c_t} \tag{8}$$

We want to show that this implies

$$m^{h^2}-1 < (n^{h^2}-1)^{1/h}\left(1+\frac{\log(n^{h^2}-1)}{(n^{h^2}-1)^{1/(d_T\cdot h)}}\right)^{d_T} \tag{9}$$

For such m satisfying (8) we have

$$(m^{h^2}-1)^h \;<\; \left(\left(n^{1/h}\left(1+\frac{\log(n)}{n^{1/(d_t\cdot h)}}\right)^{d_t} - \frac{\log\log(n)}{c_t}\right)^{h^2} - 1\right)^h \tag{10}$$

We require c_T to be sufficiently large such that $N \geq 2^{c_T^2}$ implies that $\frac{\log\log(n)}{c_t} > (1+\frac{\log(n)}{n^{1/(d_t\cdot h)}})^{d_t}$ (note that since $N = n^{h^2}-1$, we have that n is monotone increasing in N). Then the right hand side in (10) is upper bounded by

$$\left(\left((n^{1/h}-1)\cdot\left(1+\frac{\log(n)}{n^{1/(d_t\cdot h)}}\right)^{d_t}\right)^{h^2} - 1\right)^h \leq (n^{h^2}-1)\cdot\left(1+\frac{\log(n)}{n^{1/(d_t\cdot h)}}\right)^{d_t\cdot h^3} - 1$$

Thus in order to show (9), it suffices to show that

$$(n^{h^2}-1)\cdot\left(1+\frac{\log(n)}{n^{1/(d_t\cdot h)}}\right)^{d_t\cdot h^3} - 1 \leq (n^{h^2}-1)\cdot\left(1+\frac{\log(n^{h^2}-1)}{(n^{h^2}-1)^{1/(d_T\cdot h)}}\right)^{d_T\cdot h} \tag{11}$$

Finally, in order to satisfy (11), we can just choose d_T sufficiently large such that the inequalities $d_t\cdot h^3 \leq d_T\cdot h$, $\log(n)\leq\log(n^{h^2}-1)$ and $n^{1/(d_t\cdot h)} \geq (n^{h^2}-1)^{1/(d_T\cdot h)}$ hold separately. This concludes the proof of the lemma. □

Lemma 4.6. *(Powers of F satisfy Property $[\star]$)*
For each $q\in\mathbb{N}$, there exist integers c_q, d_q, p_q such that the function $F^q(n)$ satisfies Property $[\star]$ with parameters c_q, d_q, p_q.

Proof. First we prove that F satisfies Property $[\star]$. The rest of the Lemma then follows from Lemma 4.5. Suppose we have already fixed integers c_1, d_1, p_1. Now suppose that $n\geq 2^{c_1^2}$ and $c_1 \leq a \leq b \leq \frac{\log(n)}{c_1}$ are such that $F(n) > an^b$. Say in stage n of the construction of the union function F, a guess $(\tilde{S}_{(i)}, b_i, l_i)$ or $(\tilde{S}_{(i)}, b_i - p_{(i)}, l_i)$ has been selected. By definition, in any of the resulting cases we have $F(n)\leq n^{\log^*(n)}$. Now it is easy to check that the associated interval I_{n,d_1} is contained in $I_{\log^*(n)}\cup I_{\log^*(n)-1}$, so one of the two intervals $I_{\log^*(n)}, I_{\log^*(n)-1}$ contains at least half of the integers in I_{n,d_1}. Suppose that $I_{\log^*(n)-1}$ contains at least half of the integers in I_{n,d_1}, the other case is treated analogously. The size of the interval I_{n,d_1} is at least $\log^{d_1}(n)$. Thus in $I_{n,d_1}\cap I_{\log^*(n)-1}$ there are at least $\frac{\log^{d_1}(n)}{2} - O(\log^*(n))$ integers m for which $F(m) > (m-1)^{\log^*(n)-1} > m^{\log^*(n)-2}$. Thus for a given d_1, it suffices to choose $p_1\geq 2$. Hence F satisfies Property $[\star]$. For further details we refer to [H16]. □

Proof of the Padding Lemma 4.4. Suppose $L \in ATIM\tilde{E}(F^{C^2})$. Consider

$$L' = \{x10^{|x|^{h^2}-|x|-2} | x \in L\}$$

Then we have $L' \in ATIME(F^{C^2}((n+1)^{1/h^2}))$. Now since F satisfies Property [$\star$], it follows from the Closure Lemma 4.5 (a) that the function F^{C^2} satisfies Property [$\star$]. Then it follows from the Closure Lemma 4.5 (b) that also the function $t(n) := F^{C^l}((n+1)^{1/h^l})$ satisfies Property [$\star$]. Moreover, the function $t(n)$ is computable in $ZNTIME(t^{1-\epsilon})$ for $\epsilon = 1 - C^{-1}$. Therefore we also have $L' \in ATIM\tilde{E}(F^{C^2}((n+1)^{1/h^l})) \subseteq ATIM\tilde{E}(F(n+1))$, where the last inclusion follows from the inequality $F^{C^l}(n^{1/h^l}) \leq F(n)$. Thus we have $L' \in \mathrm{NTIM\tilde{E}}_{\surd^-}(F(n+1)) = \mathrm{N\tilde{P}}_{\surd^-}$. Since L' is a padded version of L, this implies

$$L \in NP = \mathrm{N\tilde{P}}_{\surd^-} = \mathrm{NTIM\tilde{E}}_{\surd^-}(F) \subseteq \mathrm{NTIM\tilde{E}}_{\surd^-}(F^{C^2}) \tag{12}$$

This concludes the proof of the Lemma. □

5 A Separation Result

This section gives a proof of the following theorem, which is a variant of the separation result from [G96] for classes $\mathrm{NTIM\tilde{E}}_{\surd^-}(t)$ and $ATIM\tilde{E}(t)$.

Theorem 1. *(Separation Theorem)*
For every function $t\colon \mathbb{N} \to \mathbb{N}$ such that $t(n)$ is computable in $ZNTIME(t(n)^{1-\epsilon})$ for some $\epsilon > 0$,

$$NTIME_{\surd^-}(t) \subsetneq ATIME(t)$$

If additionally $t(n)$ satisfies Property [$\star$],

$$NTIME_{\surd^-}(t) \subsetneq ATIM\tilde{E}(t)$$

The proof consists of adapting the methods from [PPST83] and [G96] to the case of bounded nondeterministic versus alternating time complexity classes. In particular we will first provide the Simulation Lemma 5.1 and the Hierarchy Lemma 5.2.

Lemma 5.1. *(Simulation Lemma)*
For every $t(n)$ being computable in $ZNTIME(t^{1-\epsilon})$ for some $\epsilon > 0$,

$$NTIME_{\surd^-}(t \log^* t) \subseteq ATIME(t)$$

If additionally $t(n)$ satisfies Property [$\star$], then

$$NTIME_{\surd^-}(t \log^* t) \subseteq ATIM\tilde{E}(t)$$

Proof. Let $L \in \mathrm{NTIME}_{\surd^-}(t \log^* t)$. We let $b(n) = \lceil t(n)^{1/3} \rceil$ and $a(n) = \lceil t(n)^{2/3} \log^* t(n) \rceil$. According to [PPST83], a $t(n) \log^* t(n)$-time bounded machine S is called (a, b)-block preserving if the following holds: For every input x of length n, we partition both the time interval $[1, c \cdot t(n) \cdot \log^* t(n)]$ and the tapes of the machine S into $a(n)$ blocks of length $b(n)$. Then for every block in the time interval, the read/write-heads of the machine stay within a single block on their working tapes (for different tapes, this block might be different). As it was shown in [PPST83], for every $c \cdot t(n) \log^* t(n)$-time bounded deterministic machine there exists a $3 \cdot c \cdot t(n) \log^* t(n)$-time bounded (a, b)-block preserving machine S' such that $L(S') = L(S)$. The

same holds for alternating machines, and in that case the number of guesses and the number of alternations made by S' are the same as for S.

Thus we may assume that $L = L(S)$, where S is a $c \cdot t(n) \log^* t(n)$-time bounded (a, b)-block preserving nondeterministic machine with $\text{guess}_S(n) \leq c \cdot \sqrt{\text{time}_S(n)}$. Following the lines of [G96], we construct an $O(t(n))$-time bounded alternating machine S' for L as follows:

Input: x of length n

(1) Use the $ZNTIME\tilde{E}(t^{1-\epsilon})$-algorithm to compute $t(n)$. If this algorithm returns "?", reject. Otherwise, compute the block size $b(n) = \lceil t(n)^{2/3} \rceil$ and the number of blocks $a(n) = \lceil t(n)^{1/3} \log^* t(n) \rceil$.

(2) Existentially guess a binary string g of length $c \cdot \sqrt{t(n) \log^* t(n)}$ and a partition of g into blocks $g_1, \ldots g_{a(n)}$. Note that some of the strings g_j will be empty.

The string g contains the guesses made in the computation of machine S on input x. For $1 \leq j \leq a(n)$, g_j contains the guesses made by S in the block j of the time interval $[1, c \cdot t(n) \log^* t(n)]$.

(3) Perform phases 1 and 2 from the algorithm in [G96], where in step 3 of phase 1 and step 2 of phase 2, the guess strings g_j are used to simulate the blocks of the computation of machine S. Whenever for some block to be simulated, the number of bits in the associated guess string g_j is too small, then the simulation stops and rejects.

We obtain that S' is a $O(t(n))$-time bounded alternating machine with $L(S') = L(S)$. Moreover, we may assume that $\text{time}_{S'}(n) = c \cdot t(n)$ for all n, for some constant c. Thus the function $\text{time}_{S'}(n) = c \cdot t(n)$ satisfies Property $[\star]$. Therefore, there exists a machine $\tilde{S}_{j,d}$ with $L(\tilde{S}_{j,d}) = L$ and $\text{time}_{\tilde{S}_{j,d}}(n) = O(t(n))$, whence $L \in \text{ATIM}\tilde{\text{E}}(t(n))$. This concludes the proof of the Simulation Lemma. □

Lemma 5.2. *(Hierarchy Lemma)*
If $g = o(f)$ and f is computable in $ZNTIME(f)$, then $ATIME(g) \subsetneq ATIME(f)$. Moreover, if f also satisfies Property $[\star]$, then $ATIME(g) \subsetneq AT\tilde{IME}(f)$.

Proof. An $O(f)$-time bounded alternating machine can be constructed in the standard way which diagonalizes against all $ATIME(g)$-machines. Moreover, if f also satisfies Property $[\star]$, then the diagonalizing machine can be constructed in such a way that it makes precisely $f(n)$ steps on all inputs of length n, which implies that the running time function of this machine satisfies Property $[\star]$. □

Proof. (of the Theorem 1) First we note that if $t(n)$ satisfies Property $[\star]$ and is computable in $ZNTIME(t(n)^{1-\epsilon})$, then $ATIME(t) = AT\tilde{IME}(t)$.

Suppose now that $t(n)$ is computable in $ZNTIME(t(n)^{1-\epsilon})$ and furthermore, $\text{NTIME}_{\sqrt{}}(t) = ATIME(t)$. We have

$$\text{NTIME}_{\sqrt{}}(t) \subseteq \text{NTIME}_{\sqrt{}}(t\sqrt{\log^* t}) \subseteq \text{NTIME}_{\sqrt{}}(t \log^* t) \subseteq ATIME(t)$$

Thus by assumption, we have equality in this chain of inclusions. However, due to the Simulation Lemma 5.1,

$$\text{NTIME}_{\sqrt{}}(t\sqrt{\log^* t}) \subseteq ATIME\left(\frac{t}{\sqrt{\log^* t}}\right),$$

which now yields

$$ATIME\left(\frac{t}{\sqrt{\log^* t}}\right) = ATIME(t),$$

a contradiction to the Hierarchy Lemma 5.2. □

Corollary 2. $NP \neq AP$.

Proof. Suppose that $NP = AP = PSPACE$. Then this implies $NP = \mathrm{NTIME}\tilde{\mathrm{E}}_{\surd^-}(F) = ATIME\tilde{E}(F) = AP$. Making use of padding we obtain $\mathrm{NTIME}\tilde{\mathrm{E}}_{\surd^-}(F^{C^2}) = ATIME\tilde{E}(F^{C^2})$. Since F and therefore also F^{C^2} is computable in $ZNTIME(F^C)$, this is a contradiction to the Separation Theorem (Thm. 1). □

Discussion. We may ask if the methods presented here can be used to separate PH from $AP = PSPACE$, or even to prove strictness of PH. We are currently not able to answer this question. If we start from any assumption which does not directly imply $NP = coNP$, then it is not clear how to guarantee that the union function $F(n)$ is computable in time $F(n)^c$ for some constant c. Recall that $NP = coNP$ implies that the problem Check can be solved by a ZTIME-algorithm. This means whenever we have to solve an instance of the Check problem within the computation of $F(n)$, we can run both the nondeterministic algorithm for the instance of Check and for the instance of the complement. Thus there always exists a computation path that yields an output, and every path which does not reject yields the correct answer. If we cannot make use of the implication $NP = coNP$, this does not seem to work anymore.

Nevertheless, we can modify the construction presented here such as to obtain the following refinement. We assume $NP = \Sigma_2^p$. Again this implies $NP = \mathrm{NP}_{\surd^-} = \Sigma_2^p$. Now we start from a standard family (S_i) of Σ_2-machines and construct an associated subfamily $(\tilde{S}_{i,d})$ which contains again for each machine S_i and every integer d a machine $\tilde{S}_{i,d}$ whose running time function satisfies Property $[\star]$. The construction of the machines $\tilde{S}_{i,d}$ is structurally the same as before. Namely, the computation on an input of length n is split into time intervals

$$T_0 = \left(0, a_1 n^{b_1}\right], T_1 = \left(a_1 n^{b_1}, a_2 n^{b_2}\right], \ldots, T_{L-1} = \left(a_{L-1} n^{b_{L-1}}, a_L n^{b_L}\right], T_L = \left(a_L n^{b_L}, \infty\right)$$

For $l < L$, the first half of the time interval T_l is used to simulate the computation of the original machine S_i on the given input. The second half of the time interval is used to check if the predicate $P(i, d, n, a_{l+1}, b_{l+1})$ holds, i.e. if the machine is allowed to continue its computation in the next time interval T_{l+1}. Now the difference to the previous construction is the following. Suppose that the machine is in some interval T_l of its computation and has to use the second half of this interval to decide if it is allowed to continue the computation in the next interval T_{l+1}. For this purpose, the machine has to check if the associated instance of the predicate $P(i, d, n, a_{l+1}, b_{l+1})$ holds. Before, this was done nondeterministically, which might increase the number of alternations made during the computation. Now we proceed differently such as to avoid this. Namely, if at the beginning of the second halt of the interval T_l, the machine is in an existential state, then it tests existentially (i.e. nondeterministically) if the predicate $P(\,)$ holds. In all the branches which do not yield a decisive answer, the computation stops and the machine rejects. We know that there is at least one computation path which yields an answer, and this answer is then correct. However, if the machine is in a universal state at the beginning of the interval T_l, it also runs the nondeterministic algorithm for the predicate $P(\,)$, but now in every computation path where this does not yield a decisive answer, the machine stops its computation and accepts. Again, there is at least one path which yields an answer to the question if the predicate $P(\,)$ holds, and this answer is correct. In any such computation path which yields the correct answer, the machine stops and rejects or continues its computation, depending on whether the answer is no or yes. Altogether we obtain that if S_i is a nondeterministic machine, then $\tilde{S}_{i,d}$ is also nondeterministic, and if S_i is a Σ_2-machine, then $\tilde{S}_{i,d}$ is also a Σ_2-machine. Thus we obtain in this way that the assumption $\mathrm{NP} = \Sigma_2^p$ implies that $\mathrm{NP} = \mathrm{N}\tilde{\mathrm{P}}_{\surd^-} = \tilde{\Sigma}_2^p$. Now we construct

the union function F as before. We obtain $\tilde{\mathrm{NP}}_{\surd^-} = \mathrm{NTIME}\tilde{\mathrm{E}}_{\surd^-}(F) = \tilde{\Sigma}_2(F) = \tilde{\Sigma}_2^p$. F is now still computable in $\mathrm{ZNTIME}(F^C)$, and a padding construction yields $\mathrm{NTIME}\tilde{\mathrm{E}}_{\surd^-}(F^{C^2}) = \tilde{\Sigma}_2(F^{C^2})$. This will then be a contradiction to the following version of Gupta's result: For every function $t(n)$ such that t satisfies Property $[\star]$ and t is computable in $\mathrm{ZNTIME}(t^{1-\epsilon})$ for some constant $\epsilon > 0$, $\mathrm{NTIME}\tilde{\mathrm{E}}_{\surd^-}(t) \subsetneq \tilde{\Sigma}_2(t)$. Thus the assumption $\mathrm{NP} = \Sigma_2^p$ cannot hold.

References

[BDG89] J.L. Balcazar, J. Diaz, J. Gabarro, *Structural Complexity II*, Springer, 1989.

[B67] M. Blum, *A Machine-Independent Theory of the Complexity of Recursive Functions*, J. ACM, XIV, No. 2, 1967, pp. 322-336.

[BGW70] R.V. Book, S.A. Greibach, B. Wegbreit, *Time and Tape Bounded Turing Acceptors and AFL's*, J. Com. and Sys. Sci., 4 (1970), pp. 606-621.

[CKS81] A.K. Chandra, D.C. Kozen, L.J. Stockmeyer, *Alternation*, Journal of the ACM, 28 (1981), pp. 114-133.

[CS76] A.K. Chandra, L.J. Stockmeyer, *Alternation*, Proc. 17th Symp. on Foundations of Computer Science, 1976, pp.98-108.

[G96] S. Gupta, *Alternating Time Versus Deterministic Time: A Separation*, Math. Systems Theory 29, pp. 661-672 (1996).

[H16] M. Hauptmann, *On Alternation and the Union Theorem*, Preprint, submitted to Computational Complexity 11/2015, also available at Computing Research Repository (CoRR), arXiv:1602.047816, 2016.

[K81] R. Kannan, *Towards Separating Non-Deterministic Time from Deterministic Time*, FOCS, 22 (1981), pp. 335-343.

[K83] R. Kannan, *Alternation and the Power of Non-Determinism*, STOC, 15 (1983), pp. 344-346.

[K85] K. Kobayashi, *On proving time constructibility of functions*, Theoretical Computer Science 35, pp. 215-225, 1985.

[K80] D. Kozen, *Indexing of subrecursive classes*, Theoretical Computer Science 11, pp. 277-301, 1980.

[McCM69] E.M. McCreight, A.R. Meyer, *Classes of Computable Functions Defined by Bounds on Computation: Preliminary Report*, Proc. 1st Annual ACM Symposium on Theory of Computing, pp. 79-88, 1969.

[PPST83] W.J. Paul, N. Pippenger, E. Szemeredi, W.T. Trotter, *On Determinism versus Non-Determinism and Related Problems*, Proc. IEEE FOCS, pp. 429-438, 1983.

[PPR80] W.J. Paul, E.J. Prauss and R. Reischuk, *On Alternation*, Acta Informatica 14, pp. 243-255, 1980

[PR81a] W.J. Paul and R. Reischuk, *On Alternation, II*, Acta Informatica 14, pp. 391-403, 1980

[PR81b] W.J. Paul and R. Reischuk, *On Time versus Space, II*, J. Comp. and Sys. Sci. 22, pp. 312-327, 1981.

[S01] R. Santhanam, *On Separators, Segregators and Time versus Space*, Proceedings of the 16th Annual Conference on Computational Complexity pp. 286-294, 2001

[SFM78] J.I. Seiferas, M.J. Fischer and A.R. Meyer, *Separating Nondeterministic Time Complexity Classes*, Journal of the ACM, Vo. 25, No. 1, pp. 146-167, 1978.

[S76] L.J. Stockmeyer, *The polynomial-time hierarchy*, Theoretical Computer Science, vol.3, pp. 1-22, 1976.