Relating Polynomial Time to Constant Depth*

Heribert Vollmer
Theoretische Informatik
Universität Würzburg
Am Exerzierplatz 3
D-97072 Würzburg, Germany
vollmer@informatik.uni-wuerzburg.de

Abstract

Going back to the seminal paper [FSS84] by Furst, Saxe, and Sipser, analogues between polynomial time classes and constant depth circuit classes have been considered in a number of papers. Oracles separating polynomial time classes have been obtained by diagonalization making essential use of lower bounds for circuit classes. In this note we show how separating oracles can be obtained uniformly from circuit lower bounds *without the need of carrying out a particular diagonalization.* Our technical tool is the leaf language approach to the definition of complexity classes.

1 Introduction

A major breakthrough in proving lower bounds in complexity theory was achieved in the early eighties by Furst, Saxe, and Sipser [FSS84] who proved that the parity function requires super-polynomial size when computed by constant depth circuits, i.e. par $\notin$ AC^0. In their seminal paper they also pointed out a connection between the constant depth circuit class AC^0 and the polynomial time hierarchy [Sto97]. They showed that if one could prove that parity cannot be done by quasi-polynomial size constant depth circuits then there is an oracle separating the polynomial hierarchy from PSPACE. This bound (and therefore also the existence of such an oracle) was later achieved by Yao [Yao85].

The connection between AC^0 and the polynomial hierarchy, and more exactly between depth k AC^0 and the class Σ_k^p from the polynomial hierarchy, was made precise

*Some of the results of this paper were already announced at the 2nd Asian Computing Science Conference.

by Håstad [Hås88]. He gave in detail oracle constructions by diagonalization, where the diagonalization essentially depends on circuit lower bounds.

This approach has since then been employed fruitfully a number of times, see e.g. [Hås88, Ko89, Gre91, BU97]—to name only very few examples. However, all the times the actual construction of the oracle requires a diagonalization where one has to show by a stage construction that a certain *test language* is not contained in a given complexity class. That the diagonalization works is always a consequence of a circuit lower bound.

In this paper we prove a uniform diagonalization theorem. We give a general result which without further ado gives separating oracles from circuit lower bounds. Our result shows that once the main combinatorial work has been done (by giving the lower bound), the existence of a corresponding oracle follows immediately—to explicitly define a test language and give a stage construction of the oracle (as in all the above cited papers) thus becomes obsolete. We demonstrate the applicability of our method by reproving some of the older results in just a few lines. Though we will give only three examples in Sect. 5 it should be remarked that we know of no result in the literature where an oracle was constructed building on a circuit lower bound, which could not be obtained using our general theorem.

Our uniform diagonalization theorem is an application of the general oracle construction method from [BCS92, Ver93]. In order to be able to use it we point out some relations between polynomial time classes and constant depth classes which a reader interested in leaf language characterizations [HLS$^+$93, JMT94, BS97] might find interesting in its own.

We then turn to the "nagging question" [For97] if there is an oracle Y that separates P^{PP} from PSPACE. This is beyond the frontier of what can be achieved by the currently known separation techniques. We give an easy to state sufficient criterion for the existence of such an oracle, and moreover, we give the (to our best knowledge) first statement in terms of circuit classes which is *equivalent* to the existence of Y above. We consider this result interesting since it was not known before what the analogues of $\mathrm{P}^{(\cdot)}$ classes are in the circuit world.

In the same spirit we finally turn to the question of oracle separability of PSPACE from MidbitP and $\mathrm{PP}^{\oplus\mathrm{P}}$ and give equivalent statements in terms of circuits.

2 Preliminaries

We assume the reader is familiar with basic complexity theory notions; refer to the standard literature [BC94, Pap94, BDG95].

In the early nineties a general method to obtain oracle separations was given independently in [BCS92, Ver93]. This became later known as the leaf language approach to the definition of complexity classes [HLS$^+$93, Pap94]. In the sequel we use it as our main technical tool. For this, let M be a nondeterministic Turing machine, halting on every path, with some order on the nondeterministic choices. Then, $\mathrm{leafstring}^{M}(x)$ is the concatenation of the symbols printed at the leaves of the computation tree of M on input x. Call a computation tree of a machine M *balanced*, if all of its computation paths have the same length, and moreover, if we identify every path with the string over $\{0,1\}$ describing the sequence of nondeterministic choices on this path, then there is some string z such that all paths y with $|y| = |z|$ and $y \leq z$ (in lexicographic ordering) exist, but no such path with $y > z$ exists. Now given a language A, this language defines the class $\mathrm{BalancedLeaf}^{P}(A)$ of all languages B for which there exists a nondeterministic polynomial time machine M whose computation tree is always balanced, such that $x \in B \iff \mathrm{leafstring}^{M}(x) \in A$. Thus A in a sense defines the acceptance criterion for the class $\mathrm{BalancedLeaf}^{P}(A)$. (This class is denoted by $\mathcal{C}[A, \overline{A}]$ in [BCS92]; *caveat:* Papadimitriou [Pap94] uses the same notation but presupposes full binary computation trees.) Let $\mathcal{C}$ be a class of languages. The class $\mathrm{BalancedLeaf}^{P}(\mathcal{C})$ consists of the union over all $A \in \mathcal{C}$ of the classes $\mathrm{BalancedLeaf}^{P}(A)$.

The leaf language notion was very recently generalized in [BS97, Vol97] as follows: Let $\mathcal{C}$ be a complexity class and A be a set of binary strings. Then $A \cdot \mathcal{C}$ consists of all languages L for which there exist some $B \in \mathcal{C}$ and a function $f \in \mathrm{FP}$ such that for all x, $x \in L \iff (c_B(x,1)c_B(x,2)\cdots c_B(x,f(x))) \in A$. It can be seen that $A \cdot \mathrm{P} = \mathrm{BalancedLeaf}^{P}(A)$. Below, we will use the following shorthand:

$$(A)\mathcal{C} = A \cdot \mathcal{C}$$
$$(A, A_1, \ldots, A_k)\mathcal{C} = A \cdot (A_1, \ldots, A_k)\mathcal{C}$$

3 Circuit Classes

In the following, $g_1, \ldots, g_k$ will denote families of boolean functions, i.e. $g_i = (g_i^n)_{n\in\mathbb{N}}$, where $g_i^n \colon \{0,1\}^n \to \{0,1\}$. We will freely identify a family g_i of boolean functions with the subset of $\{0,1\}^*$ whose characteristic function is given by the family, i.e. with the set $\left\{ x \mid g_i^{|x|}(x) = 1 \right\}$.

We have to use the following families of boolean functions:

- $\vee = (\vee^n)_{n\in\mathbb{N}}$, where $\vee^n(x_1, \ldots, x_n) = 1 \iff \sum_{i=1}^{n} x_i > 0$.
- $\wedge = (\wedge^n)_{n\in\mathbb{N}}$, where $\wedge^n(x_1, \ldots, x_n) = 1 \iff \sum_{i=1}^{n} x_i = n$.
- $\mathsf{u} = (\mathsf{u}^n)_{n\in\mathbb{N}}$, $\mathsf{u}^n(x_1, \ldots, x_n) = 0$ if $\sum_{i=1}^{n} x_i = 0$, $\mathsf{u}^n(x_1, \ldots, x_n) = 1$ if $\sum_{i=1}^{n} x_i = 1$, and $\mathsf{u}^n(x_1, \ldots, x_n)$ is undefined otherwise. This means that if we construct circuits using u gates, we have to ensure the promise that they never have more than one input wire which carries the value 1.
- $\mathrm{mod}_k = (\mathrm{mod}_k^n)_{n\in\mathbb{N}}$, where $\mathrm{mod}_k^n(x_1, \ldots, x_n) = 1 \iff \sum_{i=1}^{n} x_i \not\equiv 0 \pmod{k}$. We set $\mathrm{par} = \mathrm{mod}_2$.
- $\mathrm{maj} = (\mathrm{maj}^n)_{n\in\mathbb{N}}$, where $\mathrm{maj}^n(x_1, \ldots, x_n) = 1 \iff \sum_{i=1}^{n} x_i > \frac{n}{2}$.

We want to consider families of unbounded fan-in circuits, where the gate types are such families of boolean functions. Let $C[g_1, \ldots, g_k]$ be the class of all languages L for which there exists a family $C = (C_n)_{n\in\mathbb{N}}$ of unbounded fan-in boolean circuits of polynomial size and constant depth, where more precisely we require that the circuits are leveled, the depth is exactly k, and on level i we have only gates of type g_i for $1 \leq i \leq k$. On the input level we have input variables as well as their negations. $qC[g_1, \ldots, g_k]$ is defined analogously but now we allow quasi-polynomial size, i.e. the size of C_n must be bounded by some $2^{\log^c n}$ for $c \in \mathbb{N}$. Circuits as just described are called "stratified circuits" in the literature [MT93, Reg97]. When we want to place a restriction on the fan-in of gates on certain levels, we denote the maximal fan-in in parentheses after the gate type, for example $C[g_1, \ldots, g_k(f)]$ would refer to the class $C[g_1, \ldots, g_k]$ where our circuits additionally fulfill the property that the g_k gates on level 1 have fan-in bounded by f.

Let us consider some examples. We have $AC^0_k = C[\underbrace{\vee, \wedge, \ldots}_{\text{k times}}]$, and $TC^0_k = C[\underbrace{\text{maj}, \text{maj}, \ldots}_{\text{k times}}]$. For the purpose of this paper, a depth $k+1$ perceptron will be a circuit of the form $C[\text{maj}, \underbrace{\vee, \wedge, \ldots}_{\text{k times}}]$ or $C[\text{maj}, \underbrace{\wedge, \vee, \ldots}_{\text{k times}}]$. (Sometimes arbitrary weighted threshold functions are allowed as output gate, but in our context this will make no difference.) Let S_5 denote the set of all sequences of permutations on 5 elements which multiply out to the identity permutation, suitably encoded over the binary alphabet. It is known that S_5 is complete for NC^1 under uniform projections [Bar89]. Thus we have $NC^1 = \bigcup_k C[S_5, \underbrace{\vee, \wedge, \ldots}_{\text{k times}}]$. If in the above examples we only require quasi-polynomial size we get the classes qAC^0, qTC^0, and qNC. Observe that in the quasi-polynomial context there is only one analogue of the possibly distinct classes L (logarithmic space), NC^1, and NC: this is just the class qNC which equals polylogarithmic space on Turing machines. For more background on quasi-polynomial size circuit complexity, we want to refer the reader to Barrington's excellent survey [Bar92].

We will require our circuit families C to be DLOGTIME-uniform [BIS90, BI94, Bar92] meaning that the direct connection of C is can be recognized in time logarithmic in the size of the circuit. Observe that this means that for polynomial size circuits questions about C_n can be answered in time logarithmic in n whereas for quasi-polynomial size the time is polylogarithmic in n. Our computation model for sub-linear time here is the standard one, i.e. machines with index tape and random access input tape. The index tape is not erased when an input query is made. For more background, refer to [RV97].

Theorem 3.1. *For any $g_1, \ldots, g_k$, we have*

$$(g_1, \ldots, g_k)P = \text{BalancedLeaf}^P(C[g_1, \ldots, g_k]) = \text{BalancedLeaf}^P(qC[g_1, \ldots, g_k]).$$

Proof. To prove $(g_1, \dots, g_k)\mathrm{P} \subseteq \mathrm{BalancedLeaf}^{\mathrm{P}}(\mathrm{C}[g_1, \dots, g_k])$, consider a language $A \in (g_1, \dots, g_k)\mathrm{P}$. We can think of A as being recognized by a polynomial time nondeterministic Turing machine M which on input x produces a computation tree which consists first of k levels of sequences of nondeterministic branches. In these levels the results of the nondeterministic computations are evaluated according to the languages $g_1, \dots, g_k$. The bottom part of the computation tree consists out of polynomially long deterministic paths. M is a generalized Turing machine in the sense of Goldschlager and Parberry [GP86]. Now we just have to consider the circuit family C that immediately reflects the evaluation scheme of this machine M. This family is certainly uniform since to answer questions about the direction connection language we have to follow polynomially long computation paths of M which requires time logarithmic in the size of C.

To prove $\mathrm{BalancedLeaf}^{\mathrm{P}}(\mathrm{qC}[g_1, \dots, g_k]) \subseteq (g_1, \dots, g_k)\mathrm{P}$, we are given a nondeterministic machine M producing a leaf word and a circuit family C deciding the leaf language. Define a machine M' which starting with the output gate incrementally constructs the circuit by calling a subroutine deciding the direct connection language (this can be done in time polynomial in the input length), and then branches accordingly. In the end of each path of M' the machine will have to answer questions about single bits of the leaf string of M. This can be achieved by simulating the appropriate path of M. The sequence of M's nondeterministic guesses on this path is given by the index of the input bit. ❑

4 Reducibilities

In order to state the main result from [BCS92, Ver93] we have to consider a reducibility notion among leaf languages.

Define A to be polylogarithmic time bit-reducible to B, in symbols: $A \leq_m^{plt} B$, if there are two functions f, g computable deterministically in polylogarithmic time where f is zero-one-valued, such that for all x,

$$x \in A \iff f(x,1)f(x,2)\cdots f(x,g(x)) \in B.$$

The following was proved in [BCS92, Ver93]:

Proposition 4.1. *For sets* A *and* B*,* $\mathrm{BalancedLeaf}^{\mathrm{P}}(A)^Y \subseteq \mathrm{BalancedLeaf}^{\mathrm{P}}(B)^Y$ *for all oracles* Y *if and only if* $A \leq_m^{plt} B$.

Thus to show that there exists an oracle Y for which $\mathrm{BalancedLeaf}^{\mathrm{P}}(A)^Y \not\subseteq \mathrm{BalancedLeaf}^{\mathrm{P}}(B)^Y$ one has to show that $A \not\leq_m^{plt} B$. We want to use this criterion in the following to get oracle separations from circuit lower bounds. For this end, we have to consider reductions defined by circuits. Let $\mathcal{C}$ be one of the circuits classes defined in Sect. 3. We say that $A \leq_m^{\mathcal{C}} B$ if A is many-one reducible to B where the reduction function f can be computed by $\mathcal{C}$ circuits. This means that every bit of $f(x)$ can be computed by a $\mathcal{C}$ circuit, and in the case of polynomial size circuits, the length of $f(x)$ is polynomial in the length of x; and in the case of quasi-polynomial size circuits, the length of $f(x)$ is quasi-polynomial in the length of x.

Lemma 4.2. *Let* A, B *be any languages such that* $A \leq_m^{plt} B$. *Then the following statements hold.*

1. $A \leq_m^{qC[\vee,\wedge(\log^{O(1)} n)]} B$.
2. $A \leq_m^{qC[u,\wedge(\log^{O(1)} n)]} B$.
3. $A \leq_m^{qC[\mathrm{mod}_k,\wedge(\log^{O(1)} n)]} B$ *for any* $k > 1$.
4. $A \leq_m^{qC[\mathrm{mod}_k,\vee(\log^{O(1)} n)]} B$ *for any prime* $k > 1$.

Proof. Suppose $A \leq_m^{plt} B$. Then there is some function f computable in polylogarithmic time producing the bits of the image of the reduction. Given now an input x suppose we want to compute the i-th bit of the image of the reduction, i.e. we want to compute $f(x, i)$. Due to the time restriction of this computation only a polylogarithmic part of the input x can be accessed. Which part of the input is actually looked at however depends on the particular input. But we can certainly describe the outcome of every such possible computation by an AND-gate of polylogarithmic fan-in. The inputs of such a gate are one possible set of accessed input bits. Now $f(x, i)$ is one if and only if one of these ANDs evaluates to true. This proves the first statement.

To prove statement 2, we only have to observe that due to the construction above always at most one AND-gate can evaluate to true, since given an actual input it is determined which path the machine will take, and only the AND gate corresponding to this path will output a 1.

This observation also proves statement 3. For statement 4, we remark that by Fermat's little theorem we can—if k is a prime number—easily go to the complementary case which by de Morgan transforms the AND gates into OR gates.

The circuit family is uniform in all four cases. It consists of a top gate (of type $\vee$, u, or mod_k) connected to all gates on the next level. To decide connections of an AND gate we have to follow paths of the computation of $f(x, i)$ to find out which inputs will be looked at. Since this computation is polylogarithmic time bounded, the uniformity follows. ❑

For use in the upcoming section, we notice the following observation which is immediate from the definitions.

Lemma 4.3. *Let* $\overline{g}$ *and* $\overline{g}'$ *be any sequences of gate types. Let* $A \leq_m^{qC[\overline{g}]} B$. *If* $B \in qC[\overline{g}']$, *then* $A \in qC[\overline{g}', \overline{g}]$.

This means that e.g. qAC^0, qTC^0, and qNC are closed under $\leq_m^{plt}$.

Combining Proposition 4.1, Theorem 3.1, and Lemma 4.2, we now get the following theorem:

Theorem 4.4. *Let* $\overline{g}$ *be one of the sequences* $(\vee, \wedge(\log^{O(1)} n))$, $(u, \wedge(\log^{O(1)} n))$, $(\mathrm{mod}_k, \wedge(\log^{O(1)} n))$, *or* $(\mathrm{mod}_k, \vee(\log^{O(1)} n))$ *(for* k *prime in the last gate sequence). Let* $\overline{g}_1, \overline{g}_2$ *be any sequences. If* $C[\overline{g}_1] \not\subseteq qC[\overline{g}_2, \overline{g}]$ *then there is an oracle* Y *such that* $(\overline{g}_1)P^Y \not\subseteq (\overline{g}_2)P^Y$.

5 Applications

5.1 The counting hierarchy vs. PSPACE

Let us start by first looking at a very simple example where the relevant classes are closed under $\leq_m^{plt}$.

Theorem 5.1. qNC $=$ qTC^0 *if and only if for all oracles* Y, PSPACE^Y *collapses to the counting hierarchy relative to* Y.

Proof. For the direction from left to right, if qNC $=$ qTC^0 then trivially every problem in qNC polylogarithmic time bit-reduces to a problem in qTC^0 and thus the right hand side follows from Theorem 3.1 and Proposition 4.1.

For the direction from right to left, we first conclude from Proposition 4.1 that all of qNC polylogarithmic time bit-reduces to qTC^0, but by Lemma 4.2 and Lemma 4.3 the latter class is closed under this reducibility. ❑

The proof actually shows more: For the direction from left to right a weaker assumption is sufficient.

Corollary 5.2. NC^1 $\subseteq$ qTC^0 *if and only if for all oracles* Y, PSPACE^Y *collapses to the counting hierarchy relative to* Y.

Combining both results we get an interesting consequence which was already noted in [AH94].

Corollary 5.3. NC^1 $\subseteq$ qTC^0 *if and only if* qNC $=$ qTC^0.

5.2 The Mod-hierarchy vs. PSPACE

In this subsection, again the relevant circuit classes will be closed under $\leq_m^{plt}$. Thus the proof of the Theorem 5.4 below is completely analogous to the proof of Theorem 5.1.

Let MOD-PH be the oracle hierarchy that we obtain if we take NP and all classes Mod$_k$P ($k \in \mathbb{N}$) as building blocks. From Theorem 3.1 we immediately see that MOD-PH $=$ BalancedLeaf^P(ACC^0) $=$ BalancedLeaf^P(qACC^0).

Theorem 5.4. $\mathrm{qNC} = \mathrm{qACC}^0$ *if and only if* $\mathrm{NC}^1 \subseteq \mathrm{qACC}^0$ *if and only if for all oracles* Y, PSPACE^Y *collapses to the* MOD-PH *hierarchy relative to* Y.

The results of Corollary 5.3 and Theorem 5.4 obviously hold in a more general context, which leads us to the next corollary. A similar observation was made in [Bar92], but there $\mathcal{C}$ is required to be closed under a stronger reduction (qAC^0 Turing reduction).

Corollary 5.5. *Let* $\overline{g}$ *be any sequences of gate types and let* $\mathcal{C}$ *be a class closed under* $\leq_m^{plt}$. *Then the following holds:* $\mathrm{C}[\overline{g}] \subseteq \mathcal{C}$ *if and only if* $\mathrm{qC}[\overline{g}] = \mathcal{C}$.

Special cases are Corollary 5.3, but also $\mathrm{TC}^0 \subseteq \mathrm{qACC}^0 \iff \mathrm{qTC}^0 = \mathrm{qACC}^0$, and many more.

5.3 $\oplus$P vs. $\mathrm{PP}^{\mathrm{PH}}$

We next turn to three examples where we show how to obtain known relativizations very easily using our uniform diagonalization theorem.

Theorem 5.6. *There is an oracle* Y *such that* $\oplus\mathrm{P}^Y \neq \mathrm{PP}^{\mathrm{PH}^Y}$.

Proof. This follows from Green's result [Gre91] that

$$\mathrm{par} \notin \bigcup_k \mathrm{qC}[\mathrm{maj}, \underbrace{\vee, \wedge, \ldots}_{k \text{ times}}]$$

and Theorem 4.4. ❑

5.4 The $\mathrm{PP}^{\mathrm{PH}}$ hierarchy

Theorem 5.7. *There is an oracle* Y *such that* $\Sigma_k^{p,Y} \nsubseteq \mathrm{PP}^{\Sigma_{k-2}^{p,Y}}$.

Proof. Let f_k^n be the Sipser function [BS90]. From Theorem 3.1 we see that $\Sigma_k^p = \mathrm{BalancedLeaf}^{\mathrm{P}}(f_k^n)$.

Berg and Ulfberg [BU97] show that

$$f_k^n \notin \mathrm{qC}[\mathrm{maj}, \underbrace{\vee, \wedge, \ldots}_{k-2 \text{ times}}, \wedge(\log^{O(1)} n)] \cup \mathrm{qC}[\mathrm{maj}, \underbrace{\wedge, \vee, \ldots}_{k-2 \text{ times}}, \wedge(\log^{O(1)} n)].$$

Thus from Lemma 4.2 we see that f_k^n cannot be $\leq_m^{plt}$-reducible to depth $k-1$ perceptrons. The claim now follows from Theorem 4.4. ❑

König's Lemma now immediately shows

Corollary 5.8 [BU97]. *There is a relativized world, where the hierarchy* $\left(\mathrm{PP}^{\Sigma_i^p}\right)_{i \in \mathbb{N}}$ *is infinite.*

5.5 P^{NP} vs. PP

A lot of circuit lower bounds (then leading to oracle separations) have been obtained using the so called *polynomial method in circuit complexity* [Bei93]. Our uniform separation method is amenable to these results. We give an example.

Theorem 5.9. *There is an oracle* Y *such that* $\mathrm{P}^{\mathrm{NP}^Y} \not\subseteq \mathrm{PP}^Y$.

Proof. Wagner [Wag90] showed that the class P^{NP} can be characterized by nondeterministic polynomial time Turing machines which accept their input if and only if the rightmost accepting path has an odd number. Defining

$$\mathrm{ODDMAXBIT} =_{\mathrm{def}} \left\{ (x_1 \cdots x_n) \;\middle|\; \max\left\{ i \;\middle|\; x_i = 1 \right\} \equiv 1 \pmod 2 \right\},$$

we thus get: $\mathrm{BalancedLeaf}^{\mathrm{P}}(\mathrm{ODDMAXBIT}) = \mathrm{P}^{\mathrm{NP}}$.

Beigel [Bei94] showed that ODDMAXBIT is not represented by the sign of a low degree polynomial with small coefficients. Stated in terms of circuits this means: $\mathrm{ODDMAXBIT} \notin \mathrm{qC}[\mathrm{maj}, \bigwedge(\log^{O(1)} n)]$.

Suppose now that for all oracles Y, $\mathrm{P}^{\mathrm{NP}^Y} \subseteq \mathrm{PP}^Y$; then by Proposition 4.1 and Lemma 4.2, $\mathrm{ODDMAXBIT} \le_m^{\mathrm{qC}[u, \bigwedge(\log^{O(1)} n)]} \mathrm{maj}$, thus $\mathrm{ODDMAXBIT} \in \mathrm{qC}[\mathrm{maj}, u, \bigwedge(\log^{O(1)} n)] = \mathrm{qC}[\mathrm{maj}, \bigwedge(\log^{O(1)} n)]$, which contradicts Beigel's lower bound. ❑

5.6 P^{PP} vs. PSPACE

Define $B =_{\mathrm{def}} \left\{ (w_1, \ldots, w_k) \;\middle|\; (\forall i)|w_i| = \log k \wedge \text{the parity of the median of the } w_i \text{ is one} \right\}$.

Theorem 5.10. $\mathrm{BalancedLeaf}^{\mathrm{P}}(B) = \mathrm{P}^{\mathrm{PP}}$.

Proof. Define the class MedP to consist of those functions f for which there exists a nondeterministic polynomial time Turing transducer M (i.e., a machine which produces an output string on each of its paths) such that for every x, $f(x)$ is the median of M's outputs on input x. It is known [Vol94] that there is a function $f_0 \in \mathrm{MedP}$ such that the set $\left\{ x \;\middle|\; f_0(x) \equiv 0 \pmod 2 \right\}$ is complete for P^{PP} under polynomial time many one reductions.

The claim of the theorem now follows by simple normalization of a machine M_0 for f_0, i.e. producing equal length computation paths and inserting missing paths (thus making M_0 balanced), and producing equal length outputs distributed bit by bit over neighboring paths. ❑

If we now look at the set S_5, we get a very concise condition for the existence of an oracle separating P^{PP} from PSPACE.

Theorem 5.11. *There is an oracle* Y *such that* $P^{PP^Y} \neq PSPACE^Y$ *if the language* S_5 *cannot be computed by* $qC[B, u, \wedge(\log^{O(1)} n)]$ *circuits.*

Proof. Immediate from the above characterization of P^{PP} and Theorem 4.4. ❑

Corollary 5.12. *The following statements are equivalent.*

1. *For all oracles* Y *we have* $P^{PP^Y} = PSPACE^Y$.
2. $qNC = \bigcup_k qC[B, \underbrace{\vee, \wedge, \ldots}_{k \text{ times}}]$.
3. $NC^1 \subseteq \bigcup_k qC[B, \underbrace{\vee, \wedge, \ldots}_{k \text{ times}}]$.

Proof. (1) $\Longrightarrow$ (2): From Proposition 4.1 and Lemma 4.2, we see that statement 1 implies $S_5 \leq_m^{qC[\vee, \wedge(\log^{O(1)} n)]} B$, which directly yields statement (2).

(2) $\Longrightarrow$ (3) is obvious.

(3) $\Longrightarrow$ (1): From Theorem 3.1 and Proposition 4.1, we see that statement 3 implies that relative to all oracles, PSPACE $\subseteq P^{PP^{PH}}$. However a result in [TW92] shows that relativizably the equality $P^{PP^{PH}} = P^{PP}$ holds. This implies statement 1.

❑

5.7 MidbitP vs. PSPACE

The class MidbitP was introduced in [GKR+95]. Instead of repeating the original definition we simply give a leaf language definition. For background on the class refer to [GKR+95].

Definition 5.13. MidbitP $=$ BalancedLeaf$^P(M)$, where $M =_{def} \{ \overline{x} \mid \overline{x} = x_1 \cdots x_k$ for some $k \in \mathbb{N}, x_1, \ldots, x_k \in \{0,1\}$ and the middle bit in $\sum_{i=1}^k x_i$ is $1 \}$.

Theorem 5.14. *There is an oracle* Y *such that* $MidbitP^Y \neq PSPACE^Y$ *if the language* S_5 *cannot be computed by* $qC[M, u, \wedge(\log^{O(1)} n)]$ *circuits.*

Proof. Immediate from Theorem 4.4. ❑

Corollary 5.15. *The following statements are equivalent.*

1. *For all oracles* Y *we have* $\mathrm{MidbitP}^{Y} = \mathrm{PSPACE}^{Y}$.
2. $\mathrm{qNC} = \bigcup_k \mathrm{qC}[M, \underbrace{\vee, \wedge, \ldots}_{k \text{ times}}]$.
3. $\mathrm{NC}^1 \subseteq \bigcup_k \mathrm{qC}[M, \underbrace{\vee, \wedge, \ldots}_{k \text{ times}}]$.

Proof. Similar to the proof of Corollary 5.12. For the implication (3) $\Longrightarrow$ (1) we this time use the fact that PH is low for MidbitP [GKR+95]. ❑

5.8 $\mathrm{PP}^{\oplus \mathrm{P}}$ vs. PSPACE

Theorem 5.16. *There is an oracle* Y *such that* $\mathrm{PP}^{\oplus \mathrm{P}^{Y}} \neq \mathrm{PSPACE}^{Y}$ *if the language* S_5 *cannot be computed by* $\mathrm{qC}[\mathrm{maj}, \mathrm{par}, \wedge(\log^{O(1)} n)]$ *circuits.*

Proof. Follows from Theorem 4.4. Note that this time we use statement 3 of Lemma 4.2, and we have to melt the two neighboring levels of par-gates. ❑

Corollary 5.17. *The following statements are equivalent.*

1. *For all oracles* Y *we have* $\mathrm{PP}^{\oplus \mathrm{P}^{Y}} = \mathrm{PSPACE}^{Y}$.
2. $\mathrm{qNC} = \bigcup_k \mathrm{qC}[\mathrm{maj}, \mathrm{par}, \underbrace{\vee, \wedge, \ldots}_{k \text{ times}}]$.
3. $\mathrm{NC}^1 \subseteq \bigcup_k \mathrm{qC}[\mathrm{maj}, \mathrm{par}, \underbrace{\vee, \wedge, \ldots}_{k \text{ times}}]$.

Proof. Similar to the proof of Corollary 5.12. For the implication (3) $\Longrightarrow$ (1) we this time use the fact that PH is low for $\mathrm{PP}^{\oplus \mathrm{P}}$ which can be seen applying the techniques leading to the first part of Toda's theorem [Tod91]. ❑

Finally, let us remark that since $\mathrm{PP}^{\oplus \mathrm{P}} \subseteq \mathrm{MidbitP} \subseteq \mathrm{P}^{\mathrm{PP}}$ [Tod91], the last three subsections reflect different degrees of our unability to separate PSPACE from a superclass of PP.

References

[AH94] E. Allender and U. Hertrampf. Depth reduction for circuits of unbounded fan-in. *Information & Computation*, 112:217–238, 1994.

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

[Bar92] D. A. Mix Barrington. Quasipolynomial size circuit classes. In *Proceedings 7th Structure in Complexity Theory*, pages 86–93. IEEE Computer Society Press, 1992.

[BC94] D. P. Bovet and P. Crescenzi. *Introduction to the Theory of Complexity*. International Series in Computer Science. Prentice Hall, London, 1994.

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

[BDG95] J. L. Balcázar, J. Díaz, and J. Gabarró. *Structural Complexity I*. Texts in Theoretical Computer Science. Springer Verlag, Berlin Heidelberg, 2nd edition, 1995.

[Bei93] R. Beigel. The polynomial method in circuit complexity. In *Proceedings 8th Structure in Complexity Theory*, pages 82–95. IEEE Computer Society Press, 1993. Revised version, 1995.

[Bei94] R. Beigel. Perceptrons, PP, and the polynomial hierarchy. *Computational Complexity*, 4:339–349, 1994.

[BI94] D. A. Mix Barrington and N. Immerman. Time, hardware, and uniformity. In *Proceedings 9th Structure in Complexity Theory*, pages 176–185. IEEE Computer Society Press, 1994.

[BIS90] D. A. Mix Barrington, N. Immerman, and H. Straubing. On uniformity within NC^1. *Journal of Computer and System Sciences*, 41:274–306, 1990.

[BS90] R. B. Boppana and M. Sipser. The complexity of finite functions. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume A, pages 757–804. Elsevier, 1990.

[BS97] B. Borchert and R. Silvestri. The general notion of a dot operator. In *Proceedings 12th Conference on Computational Complexity*, pages 26–44. IEEE Computer Society Press, 1997.

[BU97] C. Berg and S. Ulfberg. A lower bound for perceptrons and an oracle separation of the PP^{PH} hierarchy. In *Proceedings 12th Conference on Computational Complexity*, pages 165–172. IEEE Computer Society Press, 1997.

[For97] L. Fortnow. Counting complexity. In A. Selmand and L. A. Hemaspaandra, editors, *Complexity Theory Retrospective II*, pages 81–107. Springer Verlag, New York, 1997.

[FSS84] M. Furst, J. B. Saxe, and M. Sipser. Parity, circuits, and the polynomial-time hierarchy. *Mathematical Systems Theory*, 17:13–27, 1984.

[GKR+95] F. Green, J. Köbler, K. Regan, T. Schwentick, and J. Torán. The power of the middle bit of a #P function. *Journal of Computer and System Sciences*, 50:456–467, 1995.

[GP86] L. M. Goldschlager and I. Parberry. On the construction of parallel computers from various bases of boolean functions. *Theoretical Computer Science*, 43:43–58, 1986.

[Gre91] F. Green. An oracle separating $\oplus$P from $\mathrm{PP}^{\mathrm{PH}}$. *Information Processing Letters*, 37:149–153, 1991.

[Hås88] J. Håstad. *Computational Limitations of Small Depth Circuits*. MIT Press, Cambridge, 1988.

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

[JMT94] B. Jenner, P. McKenzie, and D. Thérien. Logspace and logtime leaf languages. In *9th Annual Conference Structure in Complexity Theory*, pages 242–254, 1994.

[Ko89] K.-I. Ko. Relativized polynomial time hierarchies having exactly k levels. *SIAM Journal on Computing*, 18:392–408, 1989.

[MT93] A. Maciel and T. Thérien. Threshold circuits for iterated multiplication: using AC^0 for free. In *Proceedings 10th Symposium on Theoretical Aspects of Computer Science*, volume 665 of *Lecture Notes in Computer Science*, pages 545–554. Springer Verlag, 1993.

[Pap94] C. H. Papadimitriou. *Computational Complexity*. Addison-Wesley, Reading, MA, 1994.

[Reg97] K. Regan. Polynomials and combinatorial definitions of languages. In L. A. Hemaspaandra and A. L. Selman, editors, *Complexity Theory Retrospective II*, pages 261–293. Springer Verlag, 1997.

[RV97] K. Regan and H. Vollmer. Gap-languages and log-time complexity classes. *Theoretical Computer Science*, 1997. To appear.

[Sto97] L. Stockmeyer. The polynomial-time hierarchy. *Theoretical Computer Science*, 3:1–22, 197.

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

[TW92] S. Toda and O. Watanabe. Polynomial time 1-Turing reductions from #ph to #p. *Theoretical Computer Science*, 100:205–221, 1992.

[Ver93] N. K. Vereshchagin. Relativizable and non-relativizable theorems in the polynomial theory of algorithms. *Izvestija Rossijskoj Akademii Nauk*, 57:51–90, 1993. In Russian.

[Vol94] H. Vollmer. On different reducibility notions for function classes. In *Proceedings 11th Symposium on Theoretical Aspects of Computer Science*, volume 775 of *Lecture Notes in Computer Science*, pages 449–460. Springer Verlag, 1994.

[Vol96] H. Vollmer. Relations among parallel and sequential computation models. In *Proceedings 2nd Asian Computing Science Conference*, volume 1179 of *Lecture Notes in Computer Science*, pages 23–32. Springer Verlag, 1996.

[Vol97] H. Vollmer. A generalized quantifier concept in computational complexity theory. Technical report, Institut für Informatik, Universität Würzburg, 1997.

[Wag90] K. W. Wagner. Bounded query classes. *SIAM Journal on Computing*, 19:833–846, 1990.

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