The Polynomial Time Function Hierarchy [1] [2]

J. Ramachandran
Computer and Information Sciences
Ohio State University
Columbus, Ohio 43210

[1]This work was supported in part by NSF Grant CCR-8909071
[2]Part of this research was done when the author was a visitor at New Mexico State University

Abstract

We study Krentel's polynomial time function hierarchy (PFH) which classifies and characterizes optimization functions using deterministic oracle transducers and NP sets as oracles. We introduce a quantifier syntax to describe PFH, and explicitly introduce the Π_k^{MM} levels of the hierarchy. We discuss complete functions, metric reducibility, and relationships between different levels of PFH in analogy to Stockmeyer's polynomial time hierarchy (PH). We show that the closure of Σ_k^{MM} under metric reductions coincides with that under Turing reductions. In contrast, we show that these two reducibilities differ for FDEXT. We then proceed to build a theory of closure properties of PFH under operators such as sum, product, majority, plurality, etc. For each choice of an exponential arity operator we consider two forms of closure: closure without context and closure under context. We present a series of results, both positive (closure is possible) and negative (closure implies collapse), on the closure of PFH for each choice of operator and form of closure.

Contents

1 Introduction **3**
- 1.1 The polynomial time function hierarchy . . . 4
- 1.2 PFH and closure . . . 5

2 Definitions **8**

3 An equivalent quantifier characterization of PFH **9**

4 The Structure of PFH **11**
- 4.1 Complete functions and metric reducibility . . . 12
- 4.2 Inclusions within PFH . . . 13
- 4.3 Turing versus metric reductions . . . 16

5 Closure Properties **17**
- 5.1 Operators on function classes . . . 17
- 5.2 Choosing operators . . . 18
- 5.3 Closure without context . . . 19
- 5.4 Closure under context . . . 19

6 Some useful lemmas **21**

7 Complexity of exponential arity operators **23**
- 7.1 A majority algorithm . . . 23
- 7.2 Recognizing the graph of an operator . . . 24

8 Closure without context of exponential arity operators **25**
- 8.1 OptP closure under max and min . . . 25
- 8.2 Our main theorem of this section . . . 25
- 8.3 Closure of OptP implies a collapse . . . 26
- 8.4 A collapse implies closure of OptP . . . 28

9 Closure under context of exponential arity operators **29**
- 9.1 OptP closure under max and min . . . 29
- 9.2 Our main theorem of this section . . . 29
- 9.3 Closure of OptP under context implies a collapse . . . 30
- 9.4 A collapse implies closure of OptP under context . . . 32

10 Corollaries to our main theorems **33**

A Appendix **36**
- A.1 Proofs from Section 3 . . . 36
- A.2 Proofs from Section 4 . . . 37
- A.3 Proof of Theorem 4.17 in Section 4 . . . 40
- A.4 Proofs from Section 6 . . . 41

1 Introduction

Structural complexity theory is concerned with the classification of problems based on their *hardness*, and in defining *classes* of problems, where membership implies a commonality in the resources that the individual solutions require. NP is one such class of decision problems accepted by nondeterministic polynomial time Turing machines, and contains a large number of important and interesting problems such as TSP, SAT, and the Hamiltonian circuit problem. Rather than consider the *decision* version of these problems it is sometimes more desirable to define *search* problems that find an explicit solution, or indeed, *optimal* solutions: Find the optimal TSP tour, find the lexicographically least satisfying assignment of a given boolean formula (if one exists), or find the Hamiltonian circuit of least weight.

Krentel, in [8] and [9], defined and studied optimization function classes defined analogously to the languages classes of Stockmeyer's polynomial time hierarchy (PH), and therefore naturally called the polynomial time *function* hierarchy (PFH). His initial motivation was that of showing finer distinctions between complete sets in NP by examining optimization versions of NP decision problems. This led to a characterization of functions computing *optimization* versions of NP decision problems, using deterministic oracle transducers and NP sets as oracles. The differences in the number of oracle queries made to compute optimal solutions formed a distinction between two given NP complete problems. For example, finding the largest clique in a graph on n nodes takes $\log n$ queries but finding the lexicographically largest satisfying assignment for an n variable boolean formula takes n queries, and it is unknown if the number of queries in the latter case can be improved.

As there are two natural *quantifiers* associated with optimization, *maximization* and *minimization*, Krentel constructed a more general analogy between languages at the k^{th} level of the polynomial time hierarchy and optimization functions built using k alternations of maxima and minima operators. The function analogs of languages in the k^{th} level of the polynomial time hierarchy, Σ_k^P, form the class Σ_k^{MM}.

The basis for the analogy between these two hierarchies is the relationship between their lowest levels; i.e., between P and PF, the class of functions computable in polynomial time; and between NP and OptP, the class of functions computing the optimal values for NP optimization problems. An OptP function is computed by allowing each branch of an NP computation tree to output a number in binary, and then taking the maximum output over all the branches. For example, the lexicographically largest satisfying assignment to a given boolean formula can be computed in this way: Make each branch guess a different truth assignment, and if an assignment is found to satisfy the formula, output the assignment as a binary string, otherwise output 0.

Another direction for exploring the properties of decision problems in NP are the *counting* analogs of these problems, captured by functions from Valiant's much studied #P. In general, complexity seems to increase as we progress from decision problems, to optimal-solution search problems, to counting-the-number-of-solutions problems.

We introduce the problems studied in greater detail in the following sections.

1.1 The polynomial time function hierarchy

In the first half of the paper we re-examine Krentel's polynomial time function hierarchy and present results on its relation to PH. We present Krentel's original definition, and an equivalent quantifier based definition that allows us to flesh out many of the structural relationships between the function classes in PFH. We explicitly introduce the Π_k^{MM} levels of PFH, consisting of *minimization* analogs of the *maximization* problems in Σ_k^{MM}, the k^{th} level of PFH.

We first attempt to build a complete analogy between PH and PFH, hoping that for every known relationship in PH an analogous relationship between the function classes of PFH holds. We extend the analogy in many ways showing that the interleaved structure of PFH is almost the same as PH, yet reveal a deep difference. Recall the notions of Turing reducibility, many-one reducibility, and function and language oracles. Krentel introduced the notion of *metric reducibility* between functions in analogy to many-one reducibility between languages. However we show that this choice may be inappropriate because metric reductions and Turing reductions to Σ_k^{MM}, the k^{th} level of PFH, are equivalent within PFH.

Theorem 1.1 *If function f is Turing reducible to some function g in Σ_k^{MM}, then there exists a function $g' \in \Sigma_k^{MM}$ such that f is metric reducible to g'.*

Thus, when using function oracles from Σ_k^{MM}, Turing reductions can be replaced by metric reductions. This is not true for many-one reducibility. In fact Kadin in [7] has shown that this query saving property does not seem to hold for language oracles: Recall that $\mathrm{P}^{\mathrm{NP}[k]}$ is the class of languages acceptable with at most k queries to an NP-complete oracle, and the query hierarchy $\mathrm{QH} = \cup_k \mathrm{P}^{\mathrm{NP}[k]}$.

Theorem 1.2 *[7] If, for any k, $\mathrm{P}^{\mathrm{NP}[k]} = \mathrm{P}^{\mathrm{NP}[k+1]}$, then $\mathrm{PH} = \mathrm{P}^{\mathrm{NP}^{\mathrm{NP}}[\log n]}$, a subclass of the Δ_3^P level of* PH.

We do show that in general Turing reducibilities and metric reducibilities to a function oracle outside PFH differ. We present a pair of functions f and g in FDEXT, such that f is Turing reducible to g, but f is not metric reducible to g. Our proof uses Kolmogorov complexity, and is simpler than similar proofs in the literature ([10]) that use diagonalization arguments.

1.2 PFH and closure

In the second and more important half of this paper we present results on the closure properties of the function classes in PFH. An operator is a function mapping sequences of positive integers to a positive integer. An operator τ on a function f defines a new function whose operands are each the value of f on some input. The resultant function's complexity is the issue at hand.

Definition 1.3

1. *Let $f : (\Sigma^*)^i \to \mathcal{N}$ be a function, and $\tau : \mathcal{N}^* \to \mathcal{N}$ be an operator. We define the* operator class $\widehat{\tau}[f]$ *as the class of functions g for which there exists some polynomial time computable function $e : \Sigma^* \to \mathcal{N}$ such that*

$$\forall x \in (\Sigma^*)^{i-1} \ [\ g(x) = \tau(f(1,x), f(2,x), \ldots, f(e(x),x))\ ]$$

   In general, for any class of functions $\mathcal{CF}$, let $\widehat{\tau}[\mathcal{CF}]$ be the class $\cup_{f \in \mathcal{CF}} \{\widehat{\tau}[f]\}$.

2. *A function g from operator class $\widehat{\tau}[f]$ is* polynomial time computable *if there exists a polynomial p such that $g(x) = \tau(f(1,x), f(2,x), \ldots, f(e(x),x))$ is computable in time $p(|f(1,x)| + |f(2,x)| + \cdots + |f(e(x),x)|)$ and $|g(x)| \leq p(|x|)$. Note that $e(x)$ can be exponentially high.*

Given a choice of τ and $\mathcal{CF}$, we are interested in comparing the complexity of the class $\widehat{\tau}[\mathcal{CF}]$ to $\mathcal{CF}$. Informally our results cover questions constructed by fixing the following parameters:

1. An operator τ such as maximum, minimum, sum, product, majority, plurality or median of a set of positive integers, S. The operator τ could potentially have exponentially many operands, given that $e(x)$ is polynomial time computable not polynomially bounded in value.

2. A choice of the actual *form* of closure to be studied, either *closure without context* or *closure under context.*

We informally explain point 2: Closure without context, which has been studied in the literature ([12]), considers a class of functions $\mathcal{F}$ closed under an operator τ (of arity bounded by $e(\cdot)$) if and only if for all functions $f \in \mathcal{F}$, there exists a function $g \in \mathcal{F}$, such that

$$\forall x \in \Sigma^* \ [\tau(f(1,x), \ldots, f(e(x),x)) = g(x)]$$

This is a strong requirement, and for most choices of τ, closure is unlikely, often implying a strong collapse between two classes of intuitively different complexity. For example, Ogiwara and Hemachandra in [12] present a number of negative results on closure without context (which they simply called closure) such as the following: #P, although closed under addition, is not closed under subtraction, unless the counting hierarchy CH is in UP.

Ogiwara, Theirauf, Toda and Watanabe ([13]) finding closure without context too restrictive, defined the notion of *closure under context* to study the closure of #P functions under a variety of operators. Closure of #P under subtraction requires, for example, that the difference of two #P functions be computable by another #P function. As just stated, this does not seem possible. Yet, given the values of two #P functions it is a trivial matter to compute their difference. So, while it may be that, for some choice of f and g in #P, the function $h(x) = f(x) - g(x)$ may not be in #P, it is always in PF(#P) (in fact in PF(#P[1])). Pragmatically speaking, allowing the extra work of PF seems entirely reasonable, because application of a "trivial" operator should not create functions of higher complexity than #P.

Definition 1.4 *Let $\mathcal{DF}$ and $\mathcal{CF}$ be function classes.*

1. *A function h is in $\mathcal{DF} \circ \mathcal{CF}$ if there exist functions $f \in \mathcal{DF}$ and $g \in \mathcal{CF}$ such that $\forall x \in \Sigma^* \ [h(x) = f(g(x))]$. (Here the symbol $\circ$ denotes composition.)*

2. Let $\tau : \mathcal{N}^* \to \mathcal{N}$ be an operator. Then h is in $\mathcal{DF} \circ \hat{\tau}[\mathcal{CF}]$ if there exists an $f : (\Sigma^*)^i \to \mathcal{N} \in \mathcal{CF}$, an $e \in \mathrm{PF}$, and $g : \mathcal{N} \to \mathcal{N} \in \mathcal{DF}$, such that,

$$\forall x \in (\Sigma^*)^{i-1}\ [\ h(x) = g(\tau(f(1,x), f(2,x), \ldots, f(e(x), x)))\ ]$$

3. $\mathcal{CF}$ is closed under τ in context $\mathcal{DF} \circ \mathcal{CF} \iff \mathcal{DF} \circ \hat{\tau}[\mathcal{CF}] \subseteq \mathcal{DF} \circ \mathcal{CF}$.

The notion of *context* is presented by the class $\mathcal{DF} \circ \mathcal{CF}$ in the above definition, and allows us more flexibility in demonstrating closure. Ogiwara, Theirauf, Toda and Watanabe studied the closure of #P under the context PF $\circ$ #P, assuming that the operator τ under consideration has exponential arity. They showed that closure of #P under τ in context PF is unlikely for several interesting choices of operators: majority, plurality, maximization, and median. A typical result is as follows:

Theorem 1.5 #P *is closed under majority in context* PF $\circ$ #P *if and only if* PP(PP) $=$ P(#P[1]).

It is helpful to consider the simplified case when the arity of the operator τ is polynomially bounded in value. Ogiwara and Hemachandra in [12] proved that OptP is probably not closed under every polynomial arity, polynomial time computable operator τ:

Theorem 1.6 *[12] If* OptP *is closed under every polynomial arity polynomial time computable operator then* PH $=$ NP.

The above statement is made without any context class $\mathcal{DF} \circ \mathcal{CF}$. If we consider the question under context PF $\circ$ OptP, it is easy to show, using the fact that several OptP computations can be combined into a single OptP computation ($\mathrm{PF}^{\mathrm{OptP}} \subseteq \mathrm{PF}^{\mathrm{OptP}[1]}$, Proposition 4.12):

Theorem 1.7 OptP *is closed under all polynomially bounded arity, polynomial time computable operators, under the context* PF $\circ$ OptP.

Now shifting our attention to operators that are allowed to take exponentially long sequences of operands, we present the following results:

In Section 8 we prove that it is unlikely that OptP remains closed without context under an operator such as summation: The sum of exponentially many OptP functions is in OptP if and only if PP(NP) $=$ NP.

Similarly, in Section 9 on the closure of OptP under exponential arity operators in the context of PF $\circ$ OptP, we present a negative result for several operators: closure under context implies an unlikely collapse. For example,

Theorem 1.8 OptP *is closed under majority in context* PF $\circ$ OptP *if and only if* PP(NP) $=$ P(OptP[1]).

In section 10 we conclude with a concise table of our results. The results in this paper form the basis for other results on approximating PFH functions ([14], [16].)

2 Definitions

In this section we present definitions and notation for the function classes we will be working with. We use structural complexity theory definitions for Turing machines, reducibilities, and complxity classes such as P or NP, as found in [3].

The following definitions are due to Krentel.

Definition 2.1 *[8] An* NP *metric Turing machine N is a nondeterministic polynomial time bounded TM such that every computation path writes a binary number and accepts. For $x \in \Sigma^*$, we write* $\mathrm{opt}^N(x)$ *for the largest value (for a maximization problem) output by any computation of N on input x.*

Definition 2.2 *[8] A function $f : \Sigma^* \to \mathcal{N}$ is in* OptP *if there exists an* NP *metric Turing machine N such that for all $x \in \Sigma^*, f(x) = \mathrm{opt}^N(x)$.*

Definition 2.3 *[8] Let $f, g : \Sigma^* \to \mathcal{N}$. A metric reduction from f to g is a pair of polynomial time computable functions (T_1, T_2) where $T_1 : \Sigma^* \to \Sigma^*$ and $T_2 : \Sigma^* \times \mathcal{N} \to \mathcal{N}$ such that for all $x \in \Sigma^*, f(x) = T_2(x, g(T_1(x)))$.*

Definition 2.4 *[9] A polynomial time k-alternating max-min Turing transducer M is a polynomial time bounded TM with k alternations of max and min functions, beginning with max, such that every terminal state outputs a whole number in binary. For every $x \in \Sigma^*$, we write* $\mathrm{val}^M(x) \in \mathcal{N}$ *for the value of the root node of M's computation on x.*

The definition of a polynomial time k-alternating *min-max* Turing transducer is immediate from the above, by beginning the alternations with a min. The computation tree of such a transducer M can be viewed as a circuit of sorts with the root being the output of the entire circuit and each gate being either a min or a max gate of its inputs. The path with the most alternations of min and max has k or less such alternations. The inputs to the circuit are the numbers computed at the bottom of the tree.

Definition 2.5 *[9]. A function $f : \Sigma^* \to \mathcal{N}$ is said to be polynomial k-alternating max-min if there exists a polynomial k-alternating max-min machine M such that, $\forall\ x \in \Sigma^*$, $f(x) = \mathrm{val}^M(x)$.*

Krentel named the class of polynomial k-alternating max-min functions Σ_k^{MM}. Note that $\Sigma_1^{MM} = \mathrm{OptP}$.

All the above definitions have relativized counterparts. A relativized complexity class has access to an *oracle*, presented as either a language or a function. A Turing machine with access to a language oracle uses a special query tape on which to write a query string, and from a special query state, makes a transition to either a YES state if the query string is in the oracle language, or to a NO state otherwise, in one step at no cost. Access to a function oracle is made using two special tapes, an INPUT tape for a query string x and an OUTPUT tape for the value of the function on x, which is written in time linear in the length of the oracle function's value.

Recall that the k^{th} level of the polynomial hierarchy is Σ_k^P.

Definition 2.6 *[9]. A function $f : (\Sigma^*)^m \to \mathcal{N}$ is in $\mathrm{F}\Delta_k^P$ if f is computable in polynomial time with an oracle for Σ_{k-1}^P.*

Finally, Krentel defined an infinite hierarchy of functions, analogous to Stockmeyer's polynomial-time hierarchy, called the *polynomial time function hierarchy*, abbreviated PFH.

Definition 2.7 *[9]* $\mathrm{PFH} = \bigcup_{k \in \mathcal{N}}\ \Sigma_k^{MM}$.

All reductions in the following discussion are polynomial time bounded.

3 An equivalent quantifier characterization of PFH

Krentel introduced the Σ_k^{MM} functions in terms of alternating max-min transducers, a machine based characterization. We now present an equivalent "max-min quantifier" characterization of Σ_k^{MM}. This is analogous to the alternating bounded quantifier characterization of the polynomial time hierarchy. We introduce the following notation for defining function classes.

Definition 3.1 *Let $R : (\Sigma^*)^i \to \mathcal{N}$ be a function of arity i, and let p be a polynomial.*

1. *The operator* $\max_p$, *applied to R, defines a new function $f : (\Sigma^*)^{i-1} \to \mathcal{N}$, as follows: $f(y_1, \ldots, y_{i-1}) = \max\{R(y_1, \ldots, y_{i-1}, y_i) : |y_i| \leq p(|y_1|)\}$. We use the notation $f(y_1, \ldots, y_{i-1}) = \max_p\ y_i\ [R(y_1, \ldots, y_{i-1}, y_i)]$.*
2. *The operator* $\min_p$, *applied to R, defines a new function $f : (\Sigma^*)^{i-1} \to \mathcal{N}$, as follows: $f(y_1, \ldots, y_{i-1}) = \min\{R(y_1, \ldots, y_{i-1}, y_i) : |y_i| \leq p(|y_1|)\}$. We use the notation $f(y_1, \ldots, y_{i-1}) = \min_p\ y_i\ [R(y_1, \ldots, y_{i-1}, y_i)]$.*
3. *For any class of functions $\mathcal{C}$, a function $h : (\Sigma^*)^{i-1} \to \mathcal{N}$ is in* $\max \mathcal{C}$ *if there exists a polynomial p and a function $f : (\Sigma^*)^i \to \mathcal{N} \in \mathcal{C}$ such that for all $x \in (\Sigma^*)^i$, $h(x) = \max_p\ y\ f(x, y)$. The class* $\min \mathcal{C}$ *is defined analogously.*

The following proposition is straightforward.

Proposition 3.2 *A function $f : (\Sigma^*)^n \to \mathcal{N}$ is in Σ_k^{MM} if and only if there exists a polynomial p and a function $R : (\Sigma^*)^{k+1} \to \mathcal{N}$ in* PF *such that for all $\langle x_1, \ldots, x_n \rangle \in (\Sigma^*)^n$,*

$$f(x_1, \ldots, x_n) = \max_p\ y_1\ \ldots\ Q_k\ y_k\ [R(\langle x_1, \ldots, x_n \rangle, y_1, \ldots, y_k)].$$

Proof. The proof is in the appendix. □

Using Proposition 3.2 we now present an alternate characterization of PFH using alternating max/min quantifiers. By Proposition 3.2 it is easy to see the Σ_k^{MM} classes defined below are identical to those in Krentel's definition. Our definition of the $\mathrm{F}\Delta_k^P$ classes is also identical to Krentel's. Finally, in explicit analogy to the Π_k^P classes of PH, we add new classes to PFH, the Π_k^{MM} classes, corresponding to minimization problems.

Definition 3.3 PFH *consists of the classes Σ_k^{MM}, Π_k^{MM}, and $\mathrm{F}\Delta_k^P$ for each $k \geq 0$, where*

1. $\Sigma_0^{MM} = \Pi_0^{MM} = \mathrm{F}\Delta_0^P = \mathrm{PF}$.
2. $\Sigma_{k+1}^{MM} = \max\ \Pi_k^{MM}$.

3. $\Pi_{k+1}^{MM} = \min\ \Sigma_k^{MM}$.

4. $\mathrm{F}\Delta_{k+1}^{P} = \mathrm{PF}^{\Sigma_k^P}$.

5. $\mathrm{PFH} = \bigcup_{k \geq 0} \Sigma_k^{MM}$.

The superscript *MM* denotes the alternating nature of *max* and *min* quantifiers in the definition above. It is straightforward to define the relativized PFH.

We retain Krentel's definition of $\mathrm{F}\Delta_{k+1}^{P}$ as the class of functions computed by deterministic oracle Turing transducers that use a Σ_k^P language as an oracle; a more true analogy would have used a Σ_k^{MM} function oracle, but we will later show that these two definitions of $\mathrm{F}\Delta_{k+1}^{P}$ are indeed equivalent.

We now proceed to establish some simple analogies with PH.

Proposition 3.4 *For every $k \geq 0$*

1. $\mathrm{F}\Delta_1^P = \mathrm{PF}$.

2. $\Sigma_k^{MM} = \max\ \Sigma_k^{MM}$.

3. $\Pi_k^{MM} = \min\ \Pi_k^{MM}$.

4. $\mathrm{F}\Delta_{k+1}^{P} = \mathrm{PF}^{\Pi_k^P}$.

5. *If $f \in \mathrm{F}\Delta_k^P$ and if g is metric reducible to f, then $g \in \mathrm{F}\Delta_k^P$.*

6. *If $\Pi_k^{MM} \subseteq \Sigma_k^{MM}$ then, $\forall i \geq k,\ \Sigma_i^{MM} = \Sigma_k^{MM}$.*

7. *If $\Sigma_k^{MM} \subseteq \Pi_k^{MM}$ then, $\forall i \geq k,\ \Pi_i^{MM} = \Pi_k^{MM}$.*

Proof. The proof is in the appendix. □

4 The Structure of PFH

The polynomial time function hierarchy has *complete* problems, as does PH, but under a different notion of reducibility.

4.1 Complete functions and metric reducibility

A function f is said to be *metric complete* for Σ_k^{MM} if $f \in \Sigma_k^{MM}$ and every function in Σ_k^{MM} is metric reducible to f.

Definition 4.1 *Let M be a max-min Turing transducer, and let $x \in \Sigma^*$, and $t \in \mathcal{N}$. Let $\mathcal{KF}_k$ be the function defined as $\mathcal{KF}_k(\langle M, x, 0^t\rangle)$ = "the value output by M on input x, running for t steps, and making at most k alternations on any path, starting from a* max.*"*

Proposition 4.2 *$\mathcal{KF}_k$ is a metric complete function for Σ_k^{MM}.*

We call $\mathcal{KF}_k$ the *canonical metric complete function* for Σ_k^{MM}. It is straightforward to define a canonical metric complete function for Π_k^{MM}.

Definition 4.3 *Let M be a deterministic function oracle Turing transducer, and let $x \in \Sigma^*$, and $t \in \mathcal{N}$. Let $\mathcal{KF}_k$ be the oracle function. Let $\mathcal{PF}^{\mathcal{KF}}$ be the function defined as $\mathcal{PF}^{\mathcal{KF}}(\langle M, x, 0^t\rangle)$ = "the value output by M on input x, running for t steps, querying oracle $\mathcal{KF}_k$."*

The above function is complete for $\mathrm{F}\Delta_k^P$. Krentel cites examples of natural metric complete functions for Σ_k^{MM}.

Definition 4.4

1. LEXMAXSAT*: given a boolean formula, compute the lexicographic largest satisfying assignment if one exists.*

2. MAXIMUM DOUBLE KNAPSACK*: given positive integers $x_1, \ldots, x_n, y_1, \ldots, y_m, N$, compute the largest $M \leq N$ such that $M = \sum_{i \in S_1} x_i$ for some $S_1 \subseteq \{1, \ldots, n\}$ and $M \neq \sum_{i \in S_2} y_i$ for all $S_2 \subseteq \{1, \ldots, m\}$.*

3. MAXIMUM k-QUANTIFIED CIRCUIT*: given a boolean circuit*

$$C(x_1^1, \ldots, x_n^1, x_1^2, \ldots, x_n^2, \ldots, x_1^k, \ldots, x_n^k)$$

   compute the maximum $x_1^1, \ldots, x_n^1$ satisfying

$$\forall x_1^2 \ldots \forall x_n^2 \exists x_1^3 \ldots \exists x_n^3 \ldots Q x_1^k \ldots Q x_n^k \ [C(x_1^1, \ldots, x_n^1, \ldots, x_1^k, \ldots, x_n^k) = 1].$$

Theorem 4.5 *[8][9]*

1. LEXMAXSAT *is metric complete for* OptP$=\Sigma_1^{MM}$.

2. MAXIMUM DOUBLE KNAPSACK *is metric complete for* Σ_2^{MM}.

3. MAXIMUM k-QUANTIFIED CIRCUIT *is metric complete for* Σ_k^{MM}.

The minimization analogs of the above natural complete problems are complete for the Π_k^{MM} levels of PFH; e.g, LEXMINSAT is complete for Π_1^{MM}, where LEXMINSAT, when given a boolean formula, computes the lexicographic smallest satisfying assignment, if one exists.

A language A is many-one reducible to a language B if there exists an $f : \Sigma^* \to \Sigma^* \in$ PF such that $\forall x \in \Sigma^*$, $[x \in A \Longleftrightarrow f(x) \in B]$. A set A is NP-complete if (1) A is in NP, and (2) every set in NP is many-one reducible to A. Krentel introduced metric reducibility between functions as an analog of many-one reducibility between languages.

4.2 Inclusions within PFH

We will now present some relationships between the classes of PFH using metric reducibility, and show why, in a sense, it is too powerful a reduction when considering distinctions between Σ_k^{MM} and Π_k^{MM} levels of PFH. NP, as is every level of PH, is closed under many-one reductions. It is not clear that the analogy is true for the function class OptP; i.e., it is not known whether OptP is closed under metric reductions. We show that PFH collapses, if this were so.

In the following proposition if Q_i is a $\max_p$ then $\overline{Q_i}$ is a $\min_p$ and vice versa.

Proposition 4.6 *Let $k \geq 1$ and let $g \in \Sigma_k^{MM} \cup \Pi_k^{MM}$ be bounded in length by some polynomial p, and let g be defined as $g(x) = Q_1\ y_1 \ldots Q_k\ y_k\ [R(x, y_1, \ldots, y_k)]$. Let g' be the function defined by*

$$g'(x) = 2^{p(|x|)} - (\overline{Q_1}\ y_1 \ldots \overline{Q_k}\ y_k\ [2^{p(|x|)} - R(x, y_1, \ldots, y_k)])$$

Then $g(x) = g'(x)$.

Proof. The proof is in the appendix. $\square$

The following proposition follows directly Proposition 4.6.

Proposition 4.7 *For all $g \in \Sigma_k^{MM}$, there exists an $f \in \Pi_k^{MM}$ such that g is metric reducible to f; and for all $g \in \Pi_k^{MM}$, there exists an $f \in \Sigma_k^{MM}$ such that g is metric reducible to f.*

Krentel showed in [8] that metric reductions are closed under composition. As a consequence $\mathrm{F}\Delta_k^P$ is closed under metric reductions. However, recall that all Σ_k^{MM} functions are metric reducible to the canonical metric complete function $\mathcal{KF}_k$ for Σ_k^{MM}. But, by Proposition 4.7, we have that Π_k^{MM} functions too are metric reducible to the same complete function.

Corollary 4.8 *If* Σ_k^{MM} (Π_k^{MM}) *is closed under metric reductions, then* $\Sigma_k^{MM} = \Pi_k^{MM}$.

It is with regard to Corollary 4.8 that we suggest metric reductions are too powerful to be a true analog of many-one reducibility, because it is known that Σ_k^P is closed under many-one reducibility.

We will now present a theorem from [9] that constructs a metric reduction from any $\mathrm{F}\Delta_{k+1}^P$ function f to a Σ_k^{MM} function that depends only on f.

Theorem 4.9 *[9] Let* $f \in \mathrm{F}\Delta_{k+1}^P$. *Then* f *can be expressed as* $f(x) = h(x, g(x))$ *where* $g \in \Sigma_k^{MM}$ *and* $h : \Sigma^* \times \mathcal{N} \to \mathcal{N}$ *is computable in polynomial time.*

Proof. The proof is in the appendix. $\square$

Krentel stated the following corollary of Theorem 4.9.

Corollary 4.10 *[9]* $\mathrm{F}\Delta_{k+1}^P \subseteq \{f : \Sigma^* \to \mathcal{N} : f$ *is metric reducible to some* $g \in \Sigma_k^{MM}\}$.

We also have the following:

Proposition 4.11 *For all* $k \geq 0$*, every function in* $\mathrm{F}\Delta_k^P$ *is metric reducible to some function in* Π_k^{MM}.

Proof. Every $f \in \mathrm{F}\Delta_k^P$ is metric reducible to some $g \in \Sigma_k^{MM}$. By Proposition 4.7, g is metric reducible to $h \in \Pi_k^{MM}$. The statement now follows from closure, under composition, of metric reductions. $\square$

Proposition 4.12 $\forall\ k \geq 0$, $\mathrm{PF}^{\Sigma_k^{MM}[1]} = \mathrm{PF}^{\Sigma_k^{MM}} = \mathrm{PF}^{\Sigma_k^P} = \mathrm{PF}^{\Pi_k^{MM}} = \mathrm{PF}^{\Pi_k^{MM}[1]}$

Proof. By Theorem 4.9 and Proposition 4.11, we have: $\mathrm{PF}^{\Sigma_k^{MM}} = \mathrm{PF}(\mathrm{PF}^{\Sigma_k^P}) = \mathrm{F}\Delta_k^P = \mathrm{PF}^{\Sigma_k^{MM}[1]}$, and $\mathrm{PF}^{\Pi_k^{MM}} = \mathrm{PF}(\mathrm{PF}^{\Pi_k^P}) = \mathrm{F}\Delta_k^P = \mathrm{PF}^{\Pi_k^{MM}[1]}$. $\square$

We now state a lemma that complements Theorem 4.9.

Lemma 4.13 $\Sigma_k^{MM} \subseteq \mathrm{F}\Delta_{k+1}^P$.

Proof. The proof is in the appendix. $\square$

We extend the analogy between PFH and PH:

Theorem 4.14 *For all* $k \geq 0$, $\Sigma_k^{MM} \cup \Pi_k^{MM} \subseteq \mathrm{F}\Delta_{k+1}^P \subseteq \Sigma_{k+1}^{MM} \cap \Pi_{k+1}^{MM}$.

Proof. The first inclusion, $\Sigma_k^{MM} \cup \Pi_k^{MM} \subseteq \mathrm{F}\Delta_{k+1}^P$, follows from Lemma 4.13, Proposition 3.4, and Proposition 4.7. We show the second inclusion by induction.

Basis For $k = 0$, $\mathrm{F}\Delta_1^P = \mathrm{PF} \subseteq \Sigma_1^{MM} \cap \Pi_1^{MM}$.

Induction hypothesis Assume $\mathrm{F}\Delta_k^P \subseteq \Sigma_k^{MM} \cap \Pi_k^{MM}$.

Induction step We will show $\mathrm{F}\Delta_{k+1}^P \subseteq \Pi_{k+1}^{MM}$. The inclusion $\mathrm{F}\Delta_{k+1}^P \subseteq \Sigma_{k+1}^{MM}$ is proven similarly. Let $f \in \mathrm{F}\Delta_{k+1}^P$ be bounded by polynomial p, and let f be metric reducible to a h in Σ_k^{MM} as witnessed by PF functions T_1 and T_2; i.e., $f(x) = T_2(x, h(T_1(x)))$. Let $h(x) = \max_p\, y\ [R(x,y)]$ where $R \in \Pi_{k-1}^{MM}$. Letting $T_1(x) = x'$, define R' as:

$$R'(x, y_1, y_2) = (R(x', y_1) - R(x', y_2)) \times 2^{p(|x'|)} + T_2(x, R(x', y_2))$$

R' is in $\mathrm{F}\Delta_k^P$, and therefore, by the induction hypothesis, is in $\Sigma_k^{MM} \cap \Pi_k^{MM}$. Let $g(x, y_1, y_2) \in \Sigma_k^{MM}$ be such that $g(x, y_1, y_2) = R'(x, y_1, y_2)$. Note that:

$$\begin{aligned} f(x) &= \min\nolimits_{p'}\, y_2\, \max\nolimits_p\, y_1\ [R'(x, y_1, y_2)] \\ &= \min\nolimits_{p'}\, y_2\, \max\nolimits_p\, y_1\ [g(x, y_1, y_2)] \end{aligned}$$

This is from the fact that

$$\max\nolimits_p\, y_1\ [R'(x, y_1, y_2)] = (h(x') - R(x', y_2)) \times 2^{p(|x'|)} + T_2(x, R(x', y_2))$$

which, in turn, is minimized over all y_2 when $R(x', y_2) = h(x')$.

Finally, g's first quantifier is a max and can be merged, by Proposition 3.4.3, with the quantifier before it, showing $f(x) \in \Pi_{k+1}^{MM}$.

$\square$

The above results in the following picture of PFH, Figure 1, an exact analogy with PH, except for the fact that one query to a function oracle suffices at the $\mathrm{F}\Delta_k^P$ levels.

Figure 1: The polynomial time function hierarchy

$$\Sigma_3^{MM} \qquad\qquad\qquad\qquad \Pi_3^{MM}$$

$$\mathrm{PF}^{\Sigma_2^{MM}[1]}$$

$$\Sigma_2^{MM} \qquad\qquad\qquad\qquad \Pi_2^{MM}$$

$$\mathrm{PF}^{\mathrm{OptP}[1]}$$

$$\mathrm{OptP} \qquad\qquad\qquad\qquad \Pi_1^{MM}$$

$$\mathrm{PF}$$

4.3 Turing versus metric reductions

We now study the differences between Turing and metric reductions:

Theorem 4.15 *If f is Turing reducible to a g in Σ_k^{MM}, then there exists a $g' \in \Sigma_k^{MM}$ such that f is metric reducible to g'.*

Thus, when using function oracles from Σ_k^{MM}, Turing reductions can be replaced by metric reductions. Kadin [7] has shown that this query saving property does not seem to hold for language oracles. Recall that $\mathrm{P}^{\mathrm{NP}[k]}$ is the class of languages acceptable with at most k queries to an NP oracle, and that the query hierarchy $\mathrm{QH} = \cup_k \; \mathrm{P}^{\mathrm{NP}[k]}$.

Theorem 4.16 *[7] If, for any k, $\mathrm{P}^{\mathrm{NP}[k]} = \mathrm{P}^{\mathrm{NP}[k+1]}$, then $\mathrm{PH} = \mathrm{P}^{\mathrm{NP}^{\mathrm{NP}}[\log n]}$, a subclass of the Δ_3^P level of* PH.

In general this query saving property is not true even for function oracles, since Turing reductions to a function oracle and metric reductions to the same function oracle may have differing power, as we show in our next result. Recall that FDEXT is the class of functions computable in time $2^{\mathcal{O}(n)}$.

Theorem 4.17 *There exist functions f and g in* FDEXT *such that f is Turing reducible to g, but f is not metric reducible to g.*

Proof. The proof is in the appendix. $\square$

5 Closure Properties

We now proceed to study the closure properties of $\mathcal{F} \subseteq$ PFH under various operators τ, where $\mathcal{F}$ is one of the classes Σ_k^{MM}, Π_k^{MM} or $\mathrm{F}\Delta_k^P$.

5.1 Operators on function classes

An operator is a function mapping sequences of positive integers to a positive integer. An operator τ on a function f defines a new function where τ's operands are each the value of f on various inputs. The resultant function's complexity is the issue at hand.

We first state some assumptions to clarify our notation used:

1. Recall the existence of a polynomial time computable bijection β between natural numbers in $\mathcal{N} - \{0\}$ and strings in Σ^*. Henceforth, informally, in the following discussion all function arguments are assumed to be strings; i.e. the number i is actually represented by the string $\beta(i)$.

2. We note that all functions considered are of arity 1, but abuse the notation $f(\langle x, y, z\rangle)$ by writing $f(x, y, z)$ instead, where $\langle \cdot, \cdots, \cdot \rangle$ is a pairing function on strings. This is without loss of generality as all functions considered are of high enough complexity to allow polynomial time pairing of arguments before a function call, or uniform projection of a paired input string into the individual components.

Definition 5.1

1. *Let $f : (\Sigma^*)^i \to \mathcal{N}$ be a function, and $\tau : \mathcal{N}^* \to \mathcal{N}$ be an operator. We define the* operator class $\widehat{\tau}[f]$ *as the class of functions g for which there exists some polynomial time computable function $e : \Sigma^* \to \mathcal{N}$ such that*

$$\forall x \in (\Sigma^*)^{i-1} \; [\; g(x) = \tau(f(1,x), f(2,x), \ldots, f(e(x),x)) \;]$$

In general, for any class of functions $\mathcal{CF}$, let $\widehat{\tau}[\mathcal{CF}]$ be the class $\cup_{f \in \mathcal{CF}} \; \{\, \widehat{\tau}[f] \,\}$.

2. *A function g from operator class $\widehat{\tau}[f]$ is* polynomial time computable *if there exists a polynomial p such that $g(x) = \tau(f(1,x), f(2,x), \ldots, f(e(x),x))$ is computable in time $p(|f(1,x)| + |f(2,x)| + \cdots + |f(e(x),x)|)$ and $|g(x)| \leq p(|x|)$. Note that $e(x)$ can be exponentially high.*

Given a choice of τ and $\mathcal{CF}$, we are interested in comparing the complexity of the class $\widehat{\tau}[\mathcal{CF}]$ to $\mathcal{CF}$.

5.2 Choosing operators

As described above, any function $g : \Sigma^* \to \mathcal{N}$ from operator class $\widehat{\tau}[\mathcal{CF}]$ depends on some *sequence length function* e computable in polynomial time. The sequence length function e in the following discussion will compute the arity of any given operator τ with respect to input string x. Let p be a polynomial bounding e's computation time.

As discussed in the introduction, operators with polynomial bounded arity are not the focus of our study as they are easier to handle. We choose instead to study operators with exponential bounded arity; i.e., $0 \leq e(x) \leq 2^{p(|x|)}$.

The choice of the actual operator is important. These are examples of operators τ that we will consider:

1. $\max(x_1, \ldots, x_m) =$ the largest number in $\{x_1, \ldots, x_m\}$.
2. $\min(x_1, \ldots, x_m) =$ the smallest number in $\{x_1, \ldots, x_m\}$.
3. $\mathrm{sum}(x_1, \ldots, x_m) = x_1 + \cdots + x_m$.
4. $\mathrm{product}(x_1, \ldots, x_m) = x_1 \times \cdots \times x_m$.
5. $\mathrm{maj}(x_1, \ldots, x_m) = \begin{cases} y, & \text{if more than half the elements of } \{x_1, \ldots, x_m\} \text{ equal } y, \text{ or} \\ ?, & \text{a symbol not in } \mathcal{N}, \text{ otherwise.} \end{cases}$
6. $\mathrm{mid}(x_1, \ldots, x_m) =$ the middle value $x_{i_{\lceil m/2 \rceil}}$ in the sorted order of $x_1, \ldots, x_m$.
7. set-plu$(x_1, \ldots, x_m)$= the set of most commonly occurring values in the set $\{x_1, \ldots, x_m\}$. For example, set-plu$(1,1,3,3,3,5,5,5) = \{3,5\}$.
8. $\mathrm{plu}(x_1, \ldots, x_m) =$ the smallest number in set-plu$(x_1, \ldots, x_m)$.

   For example, $\mathrm{plu}(1,1,3,3,3,5,5,5) = 3$.

9. $\text{maj}_w(x_1,\ldots,x_m)^1 = \begin{cases} y, & \text{if more than half the elements of } \{x_1,\ldots,x_m\} \text{ equal } y, \text{ or} \\ \text{unspecified otherwise.} \end{cases}$

We are interested in two forms of closure, closure without context and closure under context.

5.3 Closure without context

Our first series of results in this section are on those closure properties which were the focus of the theory of closure properties, for function classes, developed by Ogiwara and Hemachandra in [12].

Let $f \in \mathcal{F}$ be a function of arbitrary arity i, where $f : (\Sigma^*)^i \to \mathcal{N}$. Let $e : \Sigma^* \to \mathcal{N}$ be in PF, let p be a polynomial bounding e's computation time, and let x be an input string encoding the last $i-1$ arguments to f. Then the application of an $e(\cdot)$-arity operator τ on the function $f \in \mathcal{F}$ creates the function g in $\hat{\tau}[\mathcal{CF}]$, where

$$\forall x \in (\Sigma^*)^{i-1} \; [g(x) = \tau(f(1,x), f(2,x), \ldots, f(e(x),x))]$$

We define the notion of *closure without context*:

Definition 5.2 *Let $\mathcal{CF}$ be a class of functions. $\mathcal{CF}$ is closed under operator τ if $\hat{\tau}[\mathcal{CF}] \subseteq \mathcal{CF}$; i.e., for all $f \in \mathcal{CF}$, $f : (\Sigma^*)^i \to \mathcal{N}$, and for all $e \in \mathrm{PF}$, g is in $\mathcal{CF}$, where, for all $x \in (\Sigma^*)^{i-1}$, $g(x) = \tau(f(1,x), f(2,x), \ldots, f(e(x),x))$.*

Closure without context of $\mathcal{CF}$ under τ requires, for any appropriate choice of $f \in \mathcal{CF}$, a $\mathcal{CF}$ function that computes $\tau(f(1,x), f(2,x), \ldots, f(e(x),x))$.

5.4 Closure under context

We also study closure properties in a more flexible setting, by studying the notion of *closure under context*. Ogiwara, Theirauf, Toda and Watanabe, in [13], introduce this means of studying closure properties under certain contexts that allowed more flexibility in defining what "closure" entails. The questions asked by Ogiwara and Hemachandra in [12] are of the form: "Is $\hat{\tau}[\mathcal{CF}]$ in $\mathcal{CF}$?". Even "primitive" operators fail to satisfy this requirement: for example, #P, although closed under addition, is not closed under subtraction, unless the counting hierarchy CH is in UP (a class between P and NP, related to *uniquely* satisfiable boolean formulae).

[1]Formally, maj_w is a relation on $\mathcal{N}^* \times \mathcal{N}$, but is a partial function when majority is defined.

Closure of #P under subtraction requires, for example, that the difference of two #P functions be computable by another #P function. As just stated, this is seemingly not possible. Yet, given the values of two #P functions it is a trivial matter to compute their difference. So, while it may be that, for some choice of f and g in #P, the function $h(x) = f(x) - g(x)$ may not be in #P, it is always in PF(#P) (in fact in PF(#P[1])). Pragmatically speaking, allowing the extra work of PF seems entirely reasonable, because application of a "trivial" operator should not create functions of higher complexity that #P.

In [13] Ogiwara, Theirauf, Toda and Watanabe introduced the notion of "context". This notion considers closure at a higher level where minor oscillations can be absorbed, revealing the more drastic changes in complexity caused by applying an operator. By a context we mean a complexity class $\mathcal{DF}(\mathcal{CF})$, defined by applying some operation from the class $\mathcal{DF}$ to $\mathcal{CF}$. Recall the definition of a *function oracle.*

Definition 5.3 *Let $\mathcal{DF}$ and $\mathcal{CF}$ be function classes. A function h is in $\mathcal{DF}(\mathcal{CF})$ if there exists a $g \in \mathcal{CF}$ and an $f \in \mathcal{DF}^g$ such that $\forall x \in \Sigma^*$ $[h(x) = f(x)]$. A function h is in $\mathcal{DF} \circ \mathcal{CF}$ if there exist functions $f \in \mathcal{DF}$ and $g \in \mathcal{CF}$ such that $\forall x \in \Sigma^*$ $[h(x) = f(g(x))]$. (Here the symbol $\circ$ denotes composition.)*

Definition 5.4 *Let $\mathcal{DF}$ and $\mathcal{CF}$ be classes of functions. Let $\tau : \mathcal{N}^* \to \mathcal{N}$ be an operator. Then h is in $\mathcal{DF} \circ \widehat{\tau}[\mathcal{CF}]$ if there exists an $f : (\Sigma^*)^i \to \mathcal{N} \in \mathcal{CF}$, an $e \in \mathrm{PF}$, and $g : \mathcal{N} \to \mathcal{N} \in \mathcal{DF}$, such that,*

$$\forall x \in (\Sigma^*)^{i-1} \; [\; h(x) = g(\tau(f(1,x), f(2,x), \ldots, f(e(x),x))) \;]$$

Definition 5.5 *Let $\mathcal{DF}$ and $\mathcal{CF}$ be classes of functions. Let $\tau : \mathcal{N}^* \to \mathcal{N}$ be an operator.*

1. *$\mathcal{CF}$ is closed under τ in context $\mathcal{DF}(\mathcal{CF}) \iff \mathcal{DF} \circ \widehat{\tau}[\mathcal{CF}] \subseteq \mathcal{DF}(\mathcal{CF})$.*

2. *$\mathcal{CF}$ is closed under τ in context $\mathcal{DF} \circ \mathcal{CF} \iff \mathcal{DF} \circ \widehat{\tau}[\mathcal{CF}] \subseteq \mathcal{DF} \circ \mathcal{CF}$.*

Ogiwara, Theirauf, Toda and Watanabe applied this notion to the counting class #P and studied the closure of #P under various operators τ, in context PF $\circ$ #P. We will shift the emphasis from #P to OptP.

It is easy to show, using Theorem 4.15, that:

Proposition 5.6 PF(OptP) = PF $\circ$ OptP = PF(OptP[1]).

This shows that OptP is closed under an operator τ in context PF $\circ$ OptP if and only if functions in τ(OptP) are metric reducible to OptP. Thus, we connect the notion of *closure under context* with our earlier discussion on *metric reductions*; namely, that *they are one and the same notion.*

Informally the statement "OptP is closed under τ without context" means $\tau(\mathrm{OptP}) \subseteq$ OptP, and the statement "OptP is closed under τ in context PF $\circ$ OptP" means $\mathrm{PF}\circ\tau(\mathrm{OptP}) \subseteq \mathrm{F}\Delta_2^P$. Thus, the question: "Is $\mathcal{DF}(\tau(\mathcal{CF}))$ contained in $\mathcal{DF}\circ\mathcal{CF}$?[2]", is a *higher* notion of closure, in that we have more flexibility in investigating the actual change in complexity that occurs by applying the operator τ to $\mathcal{CF}$, by appealing to functions in a higher level of PFH.

6 Some useful lemmas

Let $e : \Sigma^* \to \mathcal{N}$ be in PF, let $f : (\Sigma^*)^2 \to \mathcal{N}$, let $g : \mathcal{N}^* \to \mathcal{N}$, and let p be a polynomial. Let $h \in \widehat{g}[f]$ be the function

$$h(x) = g(f(1,x), \ldots, f(e(x),x)),$$

and let h be bounded in length by polynomial p.

We define sets G, the graph of $g \circ f$, and B, the bits encoding the graph of $g \circ f$, as follows:

$$\begin{aligned} G &= \{\langle x,y\rangle \mid g(f(1,x),\ldots,f(e(x),x)) = y\} \text{ and} \\ B &= \{\langle x,j,b\rangle \mid \exists y\ [\langle x,y\rangle \in G, \text{ and the } j^{th} \text{ bit of } y \text{ is } b]\} \end{aligned}$$

The following lemma is from [13], with OptP substituted for #P.

Lemma 6.1

1. *B is in NP^G and g is in PF_{tt}^B.*

2. *If $B \in \mathrm{NP}$, then $g \in \mathrm{OptP}$.*

3. *If $B \in \mathrm{P}^{\mathrm{OptP}[1]}$, then $g \in \mathrm{PF}^{\mathrm{OptP}[1]}$.*

Proof.

1. The proof is straightforward.

[2]read "Is $\mathcal{CF}$ closed under τ in context $\mathcal{DF} \circ \mathcal{CF}$?".

2. The PF_{tt}^{B} algorithm for g makes $|g(x)|$ pairs of queries to B of the form: $\langle x, j, 0\rangle \in B$? or $\langle x, j, 1\rangle \in B$?. For each pair one and only one query is answered YES. As $B \in \mathrm{NP}$, we can guess a witness to the positive answer for each pair, and output $g(x)$ only on paths that can verify all the positive correctly. Hence $g \in \mathrm{OptP}$.

3. Use the fact that several OptP computations can be combined into a single OptP computation ($\mathrm{PF}^{\mathrm{OptP}} \subseteq \mathrm{PF}^{\mathrm{OptP}[1]}$, Proposition 4.12)

Toran in [19] shows:

Proposition 6.2 *A set L is in* PP(NP) *if and only if there exists a set A in* NP, *and a polynomial p, such that for every $x \in \Sigma^*$, it holds that*

$$x \in L \iff \|\{w \in \Sigma^{=p(|x|)} \mid \langle x, w\rangle \in A\}\| > 2^{p(|x|)-1}$$

We now consider two hypotheses, which recur in our main results, as a negative consequence of closure of OptP under counting operators like maj:

1. For closure without context, the hypothesis $\mathrm{PP(NP)} = \mathrm{NP}$.

2. For closure under context $\mathrm{PF} \circ \mathrm{OptP}$, the hypothesis $\mathrm{PP(NP)} = \mathrm{P(OptP[1])}$.

Lemma 6.3 *Some useful facts (that also relativize):*

1. $\mathrm{P}(\mathrm{P}^{A[1]}) \subseteq \mathrm{P}^{A}$.

2. *From Theorem 4.9:* $\mathrm{P(NP)} \subseteq \mathrm{P(OptP[1])}$.

3. *From Krentel ([8]):* $\mathrm{OptP} \subseteq \mathrm{PF}^{\mathrm{NP}}$.

4. *From Toda ([18]):* $\mathrm{PH} \subseteq \mathrm{P(PP)}$.

5. *From Toda ([18]):* $\mathrm{PP} \subseteq \Sigma_k^P \Longrightarrow \mathrm{PH} = \Sigma_k^P$.

Lemma 6.4 $\mathrm{PP(NP)} = \mathrm{NP} \iff \mathrm{PH(PP(NP))} = \mathrm{NP}$.

Proof. The proof is in the appendix. □

Lemma 6.5 $\mathrm{PP(NP)} = \mathrm{P(OptP[1])} \iff \mathrm{PH(PP(NP))} = \mathrm{P(OptP[1])}$.

Proof. The proof is in the appendix. □

7 Complexity of exponential arity operators

We use the definitions and notation of Ogiwara, Theirauf, Toda and Watanabe for continuity.

7.1 A majority algorithm

We first construct an algorithm for computing a typical operator with exponential arity. We choose the following function:

$$h(x) = \mathrm{maj}(f(1,x), f(2,x), \ldots, f(e(x),x)).$$

The following algorithm for computing h illustrates why our results are negative with regard to closure. The algorithm requires access to the power of the counting class PF(PP), and closure implies that this access can be substituted for by the PFH class OptP. Toda presents strong evidence for the general belief that counting classes such as PP are more powerful that PH, by showing PH $\subseteq$ P(PP), [18].

Theorem 7.1 *There exists an* PF(PP(NP)) *algorithm for computing* h.

Proof. Let $f \in$ OptP be bounded by polynomial p, and let R be in PF such that $f(i,x) = \max_p\, y\ [R(i,x,y)]$. If $R'(i,x,y,v)$ is the predicate "$R(i,x,y) \geq v$", then the NP predicate $g(i,x,v) = \exists_p\, y\ [R'(i,x,y,v)]$ witnesses $f(i,x) \geq v$.

Let $h'(x,v) = \|\{i \mid 1 \leq i \leq e(x) \text{ and } g(i,x,v)\}\|$. If a majority value $h(x) = v$ does exist, then, for all $v' \leq v$, $h'(x,v) \geq \left\lceil \frac{e(x)+1}{2} \right\rceil$, and, for all $v' > v$, $h'(x,v) < \left\lceil \frac{e(x)+1}{2} \right\rceil$. The language $H = \{\langle x,v\rangle \mid h'(x,v) \geq \left\lceil \frac{e(x)+1}{2} \right\rceil\}$ is in PP(NP) and therefore we can compute v by binary search over $\{0,1,\ldots,2^{p(|x|)}\}$, using H as an oracle.

Having obtained v we can verify that v indeed is the majority value with one more query to a PP(NP) oracle. Let $\overline{f}(i,x) = \max_p\, y\ [2^{p(|x|)} - R(i,x,y)]$, and let $\overline{R'}(i,x,y,v)$ be the P predicate "$(2^{p(|x|)} - R(i,x,y)) \geq v$". Let the NP predicate $\overline{g}(i,x,v) = \exists_p\, y\ [\overline{R}'(i,x,y,v)]$ witness $\overline{f}(i,x) \geq v$. Consider the #P(NP) function $t(x,v)$ computed by

$$t(x,v) = \|\{i \mid (0 \leq i \leq e(x)) \ \wedge\ (g(i,x,v+1) \vee \overline{g}(i,x,2^{p(|x|)} - v + 1))\}\|$$

The #P(NP) function t counts the number of i's such that $f(i,x)$ has an optimal value different from v, and t can be as large as $e(x)$. Therefore, v is a majority value if and only if $t(x,v) < \left\lceil \frac{e(x)+1}{2} \right\rceil$. One query to a PP(NP) oracle can determine this. Thus $h \in$ PF(PP(NP)). $\square$

7.2 Recognizing the graph of an operator

For the operators under consideration, we could prove similar upper bounds on the complexity of computing the function

$$g(x) = \tau(f(1,x), f(2,x), \ldots, f(e(x),x)),$$

as was just done for maj. However, a simpler approach is to use the ideas of Lemma 6.1 resulting in an $\mathrm{PF}_{tt}^{\mathrm{NP}^G}$ algorithm for g, where G is the graph of g. Although this simpler technique may give bounds that are somewhat high, they are sufficient for our purposes.

Here are upper bounds for computing some other operators.

Corollary 7.2 *Let f be in* OptP *and let $e(x) \in$* PF *be a function bounding the arity of operator τ. Let $g(x) = \tau(f(1,x), f(2,x), \ldots, f(e(x),x))$, and let G be the graph of g; i.e, $G = \{\langle x, y\rangle \mid g(x) = y\}$.*

1. *If $\tau =$ maj, then G is in* PP(NP).
2. *If $\tau = \mathrm{maj}_w$, then G is in* PP(NP).
3. *If $\tau =$ sum, then G is in* P(PP(NP)).
4. *If $\tau =$ mid, then G is in* P(PP(NP)).
5. *If $\tau =$ plu, then G is in* co-NP(PP(NP)).

Proof.

1. By Theorem 7.1.
2. By Theorem 7.1, as maj_w is reducible to maj.
3. G is in P(#P(P(NP))). As $\mathrm{P}(\#\mathrm{P}^A) \subseteq \mathrm{P}(\mathrm{PP}^A)$, G is in P(PP(NP)).
4. Note that $g(x) = y$ if and only if
   (a) $\exists\, i$ such that $f(i,x) = y$,
   (b) $\|\{i \leq e(x) \mid f(i,x) < y\}\| \leq \lceil e(x)/2 \rceil$, and
   (c) $\|\{i \leq e(x) \mid f(i,x) > y\}\| < \lceil e(x)/2 \rceil$.

Therefore G is in P(PP(NP)).

5. Note that $g(x) = y$ if and only if

   (a) $\forall y' \; [\| \{i \leq e(x) \mid f(i,x) = y'\} \| \leq \| \{i \leq e(x) \mid f(i,x) = y\} \|]$, and

   (b) $\forall y' < y \; [\| \{i \leq e(x) \mid f(i,x) = y'\} \| < \| \{i \leq e(x) \mid f(i,x) = y\} \|]$.

   Therefore G is in co-NP(PP(NP)).

□

8 Closure without context of exponential arity operators

We first consider OptP's closure under max or min, without context.

8.1 OptP closure under max and min

As the maximum of exponentially many OptP functions is another OptP function, by Proposition 3.4.2, we have:

Proposition 8.1 OptP *is closed under* max.

In contrast however, by Proposition 3.4.6, note that:

Proposition 8.2 *If* OptP *is closed under* min, *then* $\Pi_2^{MM} = \Sigma_1^{MM} = \mathrm{OptP}$. *Thus, closure of* OptP *under* min *implies a collapse of* PFH *to* OptP.

8.2 Our main theorem of this section

We now present the main theorem of this section.

Theorem 8.3 *The following statements are equivalent:*

1. PP(NP) = NP.

2. OptP *is closed under* sum: $\widehat{\mathrm{sum}}[\mathrm{OptP}] \subseteq \mathrm{OptP}$.

3. OptP *is closed under* maj: $\widehat{\mathrm{maj}}[\mathrm{OptP}] \subseteq \mathrm{OptP}$.

4. OptP *is closed under* maj_w: $\widehat{\mathrm{maj}_w}[\mathrm{OptP}] \subseteq \mathrm{OptP}$.

5. OptP *is closed under* mid: $\widehat{\mathrm{mid}}[\mathrm{OptP}] \subseteq \mathrm{OptP}$.

6. OptP *is closed under* plu*:* $\widehat{\text{plu}}[\text{OptP}] \subseteq \text{OptP}$.

We proceed by showing individually that (1) $\Longleftarrow$ (2), (1) $\Longleftarrow$ (3), (1) $\Longleftarrow$ (4), (1) $\Longleftarrow$ (5), and (1) $\Longleftarrow$ (6) respectively; and then show individually that (1) $\Longrightarrow$ (2), (1) $\Longrightarrow$ (3), (1) $\Longrightarrow$ (4), (1) $\Longrightarrow$ (5), and (1) $\Longrightarrow$ (6) respectively.

We proceed in a manner similar to [13].

8.3 Closure of OptP implies a collapse

Recall that the sum operator sums its operands; the maj operator returns the value that occurs more than half the time amongst its operands (or ? if no such value exists); the maj_w operator returns the value that occurs more than half the time amongst its operands (or is not a function if no such value exists); the mid operator, given a list of values, returns the middle value from the sorted arrangement of the list; and that the plu operator returns the least of the subset of elements that occur the most often in S.

Lemma 8.4 *Let* τ *be any of* $\{\text{sum}, \text{maj}, \text{maj}_w, \text{mid}, \text{plu}\}$*. If* OptP *is closed under* τ*, then* $\text{NP} = \text{PP}(\text{NP})$.

Proof. Choose $e(x)$, the arity of the operator τ under examination, to be exponentially bounded by some polynomial p, $e(x) = 2^{p(|x|)}$. Also, let L be in PP(NP) as assumed. We must show that $L \in \text{NP}$ under our hypothesis.

By proposition 6.2 there exists a polynomial p and an $A \in \text{NP}$, such that, for all $x \in \Sigma^*$ $[x \in L \iff \|\{w \in \Sigma^{=p(|x|)} \mid \langle x, w\rangle \in A\}\| > 2^{p(|x|)-1}]$. Let A be accepted by nondeterministic machine M, bounded by polynomial q.

Define f as follows. For each $x \in \Sigma^*$ and each $i \in \mathcal{N}$, $0 \leq i \leq 2^{p(|x|)}$, let the OptP function $f(i,x) = \max_q y\ [R(x, w_i, y)]$, where w_i is the i^{th} smallest string of length $p(|x|)$, and $R(x, w_i, y)$ is 1 if M accepts $\langle x, w_i\rangle$ on path y, and 0 otherwise; let $f(i,x) = 0$ otherwise. Note that, from the choice of A, q and M, we have that $x \in L$ if and only if for more than half the strings w_i of length $p(|x|), \langle x, w\rangle \in A$, i.e. $f(i,x) = 1$.

1. $\tau = \text{sum}$.

   Define the function g as

$$g(x) = \text{sum}(f(1,x), f(2,x), \ldots, f(2^{p(|x|)}, x))$$

Then $x \in L \iff g(x) > 2^{p(|x|)-1}$. By hypothesis, g is computable in OptP; i.e., there exists an metric transducer that computes g. Then it is easy to modify this metric transducer to create an NP acceptor M_g that, along every path, computes a candidate value v for $g(x)$ and accepts x on this path if and only if $v > 2^{p(|x|)-1}$. Thus $L \in \mathrm{NP}$.

2. $\tau = \mathrm{maj}$.

   Define the function g as

$$g(x) = \mathrm{maj}(f(1,x), f(2,x), \ldots, f(2^{p(|x|)}, x))$$

   Now consider any string x in Σ^*. Then $x \in L \iff g(x) = 1$. By hypothesis, g is computable in OptP; i.e., there exists an metric transducer that computes g. Then it is easy to modify this metric transducer to create an NP acceptor M_g that, along every path, computes a candidate value v for $g(x)$ and accepts x on this path if and only if $v = 1$. Thus $L \in \mathrm{NP}$.

3. $\tau = \mathrm{maj}_w$.

   Define the function g as

$$g(x) = \mathrm{maj}_w(f(1,x), f(2,x), \ldots, f(2^{p(|x|)}, x))$$

   Now consider any string x in Σ^*. Then $x \in L \iff g(x) = 1$. By hypothesis, g is computable in OptP; i.e., there exists an metric transducer that computes g. Then it is easy to modify this metric transducer to create an NP acceptor M_g that, along every path, computes a candidate value v for $g(x)$ and accepts x on this path if and only if $v = 1$. Thus $L \in \mathrm{NP}$.

4. $\tau = \mathrm{mid}$.

   Define the function g as

$$g(x) = \mathrm{mid}(f(1,x), f(2,x), \ldots, f(2^{p(|x|)}, x))$$

   Now consider any string x in Σ^*. Then $x \in L \iff g(x) = 1$. By hypothesis, g is computable in OptP; i.e., there exists an metric transducer that computes g. Then it is easy to modify this metric transducer to create an NP acceptor M_g that, along every path, computes a candidate value v for $g(x)$ and accepts x on this path if and only if $v = 1$. Thus $L \in$ NP.

5. $\tau =$ plu.

   Define the function g as

   $$g(x) = \text{plu}(f(1,x), f(2,x), \ldots, f(2^{p(|x|)}, x))$$

   Now consider any string x in Σ^*. Then $x \in L \iff g(x) = 1$. By hypothesis, g is computable in OptP; i.e., there exists an metric transducer that computes g. Then it is easy to modify this metric transducer to create an NP acceptor M_g that, along every path, computes a candidate value v for $g(x)$ and accepts x on this path if and only if $v = 1$. Thus $L \in$ NP.

8.4 A collapse implies closure of OptP

The following is the converse to Lemma 8.4.

Lemma 8.5 *Let τ be any of* {sum, maj, maj$_w$, mid, plu}*. If* NP = PP(NP) *then* OptP *is closed under τ.*

Proof. Assume NP = PP(NP). Let f be in OptP, let $e(x) \in$ PF, and let τ be an operator. Let

$$g(x) = \tau(f(1,x), f(2,x), \ldots, f(e(x), x)).$$

Let $G = \{\langle x,y \rangle \mid g(x) = y\}$ be the graph of g, and let

$$B = \{\langle x,j,b \rangle \mid \exists y\ [\langle x,y \rangle \in G \text{ and the } j^{th} \text{ bit of } y \text{ is } b]\}.$$

1. $\tau =$ sum.

   By Corollary 7.2, G is in P(PP(NP)). Therefore $B \in \text{NP}^G \in$ NP(PP(NP)). By Lemma 6.4, and the assumption of the theorem, B is in NP. g is in OptP, by Lemma 6.1. Therefore OptP is closed under sum.

2. $\tau =$ maj.

   By Corollary 7.2, G is in PP(NP), and therefore $B \in \text{NP}^G \in$ NP(PP(NP)). By Lemma 6.4, B is in NP, and by Lemma 6.1, $g \in$ OptP. Hence OptP is closed under maj.

3. $\tau = \mathrm{maj}_w$.

   By Corollary 7.2, G is in PP(NP), and therefore $B \in \mathrm{NP}^G \in \mathrm{NP(PP(NP))}$. By Lemma 6.4, B is in NP, and by Lemma 6.1, $g \in$ OptP. Hence OptP is closed under maj_w.

4. $\tau =$ mid.

   By Corollary 7.2, G is in P(PP(NP)), and $B \in \mathrm{NP}^G \in \mathrm{NP(PP(NP))}$. By Lemma 6.4, B is in NP, and by Lemma 6.1, $g \in$ OptP. Hence OptP is closed under mid.

5. $\tau =$ plu.

   By Corollary 7.2, G is in co-NP(PP(NP)). Since $B \in \mathrm{NP}^G$, B is also in Σ_2^P(PP(NP)). By Lemma 6.4, $B \in$ NP, implying in turn by Lemma 6.1 that $g \in$ OptP. Hence OptP is closed under plu.

□

9 Closure under context of exponential arity operators

9.1 OptP closure under max and min

Proposition 8.1 showed that OptP was closed under max, without regard to context. Trivially we have:

Proposition 9.1 OptP *is closed under* max *under context* PF $\circ$ OptP.

In contrast, again:

Proposition 9.2 *If* OptP *is closed under* min *in context* PF $\circ$ OptP, *then* $\Pi_2^{MM} = \mathrm{F}\Delta_2^P$. *Thus, closure under context* PF $\circ$ OptP *of* min *implies* PFH *collapses to* $\mathrm{F}\Delta_2^P$.

9.2 Our main theorem of this section

We now present the main theorem of this section.

Theorem 9.3 *The following statements are equivalent:*

1. PP(NP) = P(OptP[1])).

2. OptP *is closed under* sum *in context* PF $\circ$ OptP*:* $\widehat{\mathrm{sum}}$[OptP] $\subseteq$ PF(OptP[1])*.*

3. OptP *is closed under* maj *in context* PF $\circ$ OptP*:* $\widehat{\mathrm{maj}}$[OptP] $\subseteq$ PF(OptP[1]).

4. OptP *is closed under* maj_w *in context* PF $\circ$ OptP*:* $\widehat{\mathrm{maj}_w}$[OptP] $\subseteq$ PF(OptP[1]).

5. OptP *is closed under* mid *in context* PF $\circ$ OptP*:* $\widehat{\mathrm{mid}}$[OptP] $\subseteq$ PF(OptP[1]).

6. OptP *is closed under* plu *in context* PF $\circ$ OptP*:* $\widehat{\mathrm{plu}}$[OptP] $\subseteq$ PF(OptP[1]).

We proceed by showing individually that (1) $\Longleftarrow$ (2), (1) $\Longleftarrow$ (3), (1) $\Longleftarrow$ (4), (1) $\Longleftarrow$ (5), and (1) $\Longleftarrow$ (6) respectively; and then show individually that (1) $\Longrightarrow$ (2), (1) $\Longrightarrow$ (3), (1) $\Longrightarrow$ (4), (1) $\Longrightarrow$ (5), and (1) $\Longrightarrow$ (6) respectively.

We proceed in a manner similar to [13].

9.3 Closure of OptP under context implies a collapse

Lemma 9.4 *Let* τ *be any of* $\{\mathrm{sum}, \mathrm{maj}, \mathrm{maj}_w, \mathrm{mid}, \mathrm{plu}\}$*. If* OptP *is closed under* τ *in context* PF $\circ$ OptP*, then* P(OptP[1]) = PP(NP).

Proof. Choose $e(x)$, the arity of the operator τ under examination, to be exponentially bounded by some polynomial p, $e(x) = 2^{p(|x|)}$. Also, let L be in PP(NP) as assumed. We must show that $L \in$ P(OptP[1]) under our hypothesis.

By proposition 6.2 there exists a polynomial p, and an $A \in$ NP, such that, for all inputs $x \in \Sigma^*$ $[x \in L \iff \|\{w \in \Sigma^{=p(|x|)} \mid \langle x, w\rangle \in A\}\| > 2^{p(|x|)-1}]$. Let A be accepted by nondeterministic machine M, bounded by polynomial q.

Define f as follows. For each $x \in \Sigma^*$ and each $i \in \mathcal{N}$, $0 \leq i \leq 2^{p(|x|)}$, let the OptP function $f(i, x) = \max_q y\ [R(x, w_i, y)]$, where w_i is the i^{th} smallest string of length $p(|x|)$, and $R(x, w_i, y)$ is 1 if M accepts $\langle x, w_i\rangle$ on path y, and 0 otherwise; let $f(i, x) = 0$ otherwise. Note that, from the choice of A, q and M, we have that $x \in L$ if and only if for more than half the strings w_i of length $p(|x|), \langle x, w\rangle \in A$, i.e. $f(i, x) = 1$.

1. $\tau = \mathrm{sum}$.

   Define the function g as

$$g(x) = \mathrm{sum}(f(1, x), f(2, x), \ldots, f(2^{p(|x|)}, x))$$

   Then $x \in L \iff g(x) > 2^{p(|x|)-1}$. By hypothesis, g is computable in PF $\circ$ OptP; i.e., there exists an PF(OptP[1]) machine that computes g. Then it is easy to modify this transducer to create an acceptor M_g that computes $g(x)$ and accepts x if and only if $g(x) > 2^{p(|x|)-1}$. This creates a P(OptP[1]) algorithm for L.

(We insert a parenthetical digression here to note how OptP differs from #P with regard to closure under the operators sum and maj_w. In [13], #P is shown to be closed under maj_w in context PF $\circ$ #P. The proof depended on #P being closed under addition of exponentially many #P functions. By the above proof, it is unlikely that this property applies to OptP, and hence the technique used to show closure of #P under maj_w under context PF $\circ$ #P fails to apply to a similar proof for OptP. Indeed we show quite the opposite to be true, namely closure implies collapse and vice versa. We now return to the other cases.)

2. $\tau = \mathrm{maj}$.

   Define the function g as

   $$g(x) = \mathrm{maj}(f(1,x), f(2,x), \ldots, f(2^{p(|x|)}, x))$$

   Now consider any string x in Σ^*. Then $x \in L \iff g(x) = 1$. By hypothesis, there exists an PF(OptP[1]) machine that computes g. Then it is easy to modify this transducer to create an acceptor M_g that computes $g(x)$ and accepts x if and only if $g(x) = 1$. Thus $L \in$ P(OptP[1]).

3. $\tau = \mathrm{maj}_w$.

   Define the function g as

   $$g(x) = \mathrm{maj}_w(f(1,x), f(2,x), \ldots, f(2^{p(|x|)}, x))$$

   Now consider any string x in Σ^*. Then $x \in L \iff g(x) = 1$. By hypothesis, there exists an PF(OptP[1]) machine that computes g. Then it is easy to modify this transducer to create an acceptor M_g that computes $g(x)$ and accepts x if and only if $g(x) = 1$. Thus $L \in$ P(OptP[1]).

4. $\tau = \mathrm{mid}$.

   Define the function g as

   $$g(x) = \mathrm{mid}(f(1,x), f(2,x), \ldots, f(2^{p(|x|)}, x))$$

Now consider any string x in Σ^*. Then $x \in L \iff g(x) = 1$. By hypothesis, there exists an PF(OptP[1]) machine that computes g. Then it is easy to modify this transducer to create an acceptor M_g that computes $g(x)$ and accepts x if and only if $g(x) = 1$. Thus $L \in$ P(OptP[1]).

5. $\tau =$ plu.

   Define the function g as

$$g(x) = \text{plu}(f(1,x), f(2,x), \ldots, f(2^{p(|x|)}, x))$$

   Now consider any string x in Σ^*. Then $x \in L \iff g(x) = 1$. By hypothesis, there exists an PF(OptP[1]) machine that computes g. Then it is easy to modify this transducer to create an acceptor M_g that computes $g(x)$ and accepts x if and only if $g(x) = 1$. Thus $L \in$ P(OptP[1]).

9.4 A collapse implies closure of OptP under context

The following is the converse to Lemma 9.4.

Lemma 9.5 *Let τ be any of* {sum, maj, maj$_w$, mid, plu}. *If* P(OptP[1]) = PP(NP) *then* OptP *is closed under τ in context* PF $\circ$ OptP.

Proof. Assume P(OptP[1]) = PP(NP). Let f be in OptP, let $e(x) \in$ PF, and let τ be an operator. Let

$$g(x) = \tau(f(1,x), f(2,x), \ldots, f(e(x), x)).$$

Let $G = \{\langle x, y\rangle \mid g(x) = y\}$ be the graph of g, and let

$$B = \{\langle x, j, b\rangle \mid \exists y\ [\langle x, y\rangle \in G \text{ and the } j^{th} \text{ bit of } y \text{ is } b]\}.$$

1. $\tau =$ sum.

   By Corollary 7.2, G is in P(PP(NP)). Therefore $B \in \text{NP}^G \in$ NP(PP(NP)). By Lemma 6.5, and the assumption of the theorem, B is in P(OptP[1]). By Lemma 6.1, $g \in$ PF(OptP[1]). Therefore OptP is closed under sum in context PF $\circ$ OptP.

2. $\tau =$ maj.

By Corollary 7.2, G is in PP(NP), and therefore $B \in \mathrm{NP}^G \in \mathrm{NP}(\mathrm{PP}(\mathrm{NP}))$. By Lemma 6.5, B is in P(OptP[1]), and by Lemma 6.1, $g \in \mathrm{PF}(\mathrm{OptP}[1])$. Hence OptP is closed under maj in context PF $\circ$ OptP.

3. $\tau = \mathrm{maj}_w$.

   By Corollary 7.2, G is in PP(NP), and therefore $B \in \mathrm{NP}^G \in \mathrm{NP}(\mathrm{PP}(\mathrm{NP}))$. By Lemma 6.5, B is in P(OptP[1]), and by Lemma 6.1, $g \in \mathrm{PF}(\mathrm{OptP}[1])$. Hence OptP is closed under maj_w in context PF $\circ$ OptP.

4. $\tau = \mathrm{mid}$.

   By Corollary 7.2, G is in P(PP(NP)), and $B \in \mathrm{NP}^G \in \mathrm{NP}(\mathrm{PP}(\mathrm{NP}))$. By Lemma 6.5, B is in P(OptP[1]), and by Lemma 6.1, $g \in \mathrm{PF}(\mathrm{OptP}[1])$. Hence OptP is closed under mid in context PF $\circ$ OptP.

5. $\tau = \mathrm{plu}$.

   By Corollary 7.2, G is in co-NP(PP(NP)). Since $B \in \mathrm{NP}^G$, B is also in $\Sigma_2^P(\mathrm{PP}(\mathrm{NP}))$. By Lemma 6.5, $B \in \mathrm{P}(\mathrm{OptP}[1])$, implying in turn by Lemma 6.1 that $g \in \mathrm{PF}(\mathrm{OptP}[1])$. Hence OptP is closed under plu in context PF $\circ$ OptP.

$\square$

10 Corollaries to our main theorems

We summarize our results in Table 10 using a polynomial p and OptP.

More generally, we can modify all the details of the above proofs on exponential arity operators to yield the following:

Corollary 10.1 *For all integers i and j the following statements are equivalent:*

1. $\mathrm{PP}(\Sigma_i^P) = \Sigma_j^P$.
2. $\widehat{\mathrm{sum}}[\Sigma_i^{MM}] \subseteq \Sigma_j^{MM}$.
3. $\widehat{\mathrm{maj}}[\Sigma_i^{MM}] \subseteq \Sigma_j^{MM}$.
4. $\widehat{\mathrm{mid}}[\Sigma_i^{MM}] \subseteq \Sigma_j^{MM}$.

Table 1: Closures and collapses on exponential arity operators

τ	τ's arity $e(x) \leq 2^{p(\|x\|)}$	
	$\tau(\mathrm{OptP}) \subseteq \mathrm{OptP}$	$\mathrm{PF}(\tau(\mathrm{OptP})) \subseteq \mathrm{PF} \circ \mathrm{OptP}$
max	yes	yes
min	yes $\Longleftrightarrow$ PH = NP	yes $\Longleftrightarrow$ PH = Δ_2^P
product	no[a]	no[a]
{ sum, mid, maj$_w$, maj, plu }	yes $\Longleftrightarrow$ PP(NP) = NP	yes $\Longleftrightarrow$ PP(NP) = Δ_2^P

[a] Note that $|\tau(x)|$ can be exponential in $|x|$

5. $\widehat{\mathrm{plu}}[\Sigma_i^{MM}] \subseteq \Sigma_j^{MM}$.

Corollary 10.2 *The following statements are equivalent:*

1. $\mathrm{PP}(\Sigma_i^P) = \mathrm{P}(\Sigma_j^{MM}[1])$.

2. $\widehat{\mathrm{sum}}[\Sigma_i^{MM}] \subseteq \mathrm{PF}(\Sigma_j^{MM}[1])$.

3. $\widehat{\mathrm{maj}}[\Sigma_i^{MM}] \subseteq \mathrm{PF}(\Sigma_j^{MM}[1])$.

4. $\widehat{\mathrm{mid}}[\Sigma_i^{MM}] \subseteq \mathrm{PF}(\Sigma_j^{MM}[1])$.

5. $\widehat{\mathrm{plu}}[\Sigma_i^{MM}] \subseteq \mathrm{PF}(\Sigma_j^{MM}[1])$.

Note that the above results show a qualitative difference between min and max on one hand, and the above *counting* operators on the other. This seems to be primarily due to counting operators requiring the power of the class PP, which as Toda has shown, [18], is very powerful relative to PH. The operators max and min on the other hand do not require PP and can be computed with oracles from PH itself. The above corollaries hold with Π^{MM} substituted for Σ^{MM} everywhere.

Acknowledgments

I thank Tim Long and Ming Jye-Sheu for many helpful discussions.

References

[1] Allender, E., and Rubinstein, R. P-printable sets. Submitted for publication, 1986.

[2] Amir, A., Beigel, R., and Gasarch, W. Some connections between bounded query classes and non-uniform complexity. In *Proc. Fifth Annual IEEE Structure in Complexity Theory Conference* (1990), pp. 232–243.

[3] Balcázar, J., Diaz, J., and Gabarró, J. *Structural Complexity I.* Springer-Verlag, Berlin, 1988.

[4] Balcázar, J., Diaz, J., and Gabarró, J. *Structural Complexity II.* Springer-Verlag, Berlin, 1990.

[5] Cai, J., and Hemachandra, L. Enumerative counting is hard. *Information and Computation 82* (1989), 34–44.

[6] Hopcroft, J., and Ullman, J. *Introduction to Automata Theory, Languages, and Computation.* Addison-Wesley, Reading, Massachusetts, 1979.

[7] Kadin, J. *Restricted Turing reducibilities and the structure of the polynomial time hierarchy.* PhD thesis, Cornell University, 1988.

[8] Krentel, M. The complexity of optimization problems. *J. Computer Systems Sci. 36* (1988), 490–509.

[9] Krentel, M. Generalizations of OptP to the Polynomial Hierarchy. *Submitted to Theoretical Computer Science* (1989).

[10] Ladner, R., Lynch, N., and Selman, A. A comparison of polynomial time reducibilities. *Theor. Comput. Sci. 1* (1975), 103–123.

[11] Li, M., and Vitanyi, P. *An Introduction to Kolmogorov Complexity and its applications.* Springer-Verlag, 1993.

[12] Ogiwara, M., and Hemachandra, L. A complexity theory for feasible closure properties. In *Proc. 6th annual Structure in Complexity Theory conference* (1991), pp. 16–29.

[13] Ogiwara, M., Thierauf, T., Toda, S., and Watanabe, O. On closure properties of #P in the context of PF∘#P. In *Proc. 8th annual Structure in Complexity Theory conference* (1993), pp. 139–146.

[14] Ramachandran, J. Set bit enumeration is hard. In *Proceedings, Seventh Annual Structure in Complexity Theory* (1992), pp. 58–70. IEEE Computer Society Press.

[15] Schöning, U. *Complexity and Structure.* Lecture Notes in Computer Science Vol. 211. Springer-Verlag, 1986.

[16] Schwentick, T., and Ramachandran, J. A Note on Set Bit Enumeration. Tech. rep., The Ohio State University, Department of Computer & Information Science, The Ohio State University, Columbus, Ohio, 43210, 1994.

[17] Stockmeyer, L. The polynomial-time hierarchy. *Theor. Comput. Sci. 3* (1976), 1–22.

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

[19] Torán, J. An oracle characterization of the counting hierarchy. In *Proc. 3rd Structure in Complexity Theory Conference, IEEE* (1988), pp. 213–223.

A Appendix

A.1 Proofs from Section 3

Proposition A.1 *(Proposition 3.2) A function $f : (\Sigma^*)^n \to \mathcal{N}$ is in $\Sigma_k^{MM} \iff$ there exists a polynomial p and a function $R : (\Sigma^*)^{k+1} \to \mathcal{N}$ in* PF *such that for all $\langle x_1, \ldots, x_n \rangle \in (\Sigma^*)^n$,*

$$f(x_1, \ldots, x_n) = \max_p \; y_1 \; \ldots \; Q_k \; y_k \; [R(\langle x_1, \ldots, x_n \rangle, y_1, \ldots, y_k)].$$

Proof. We prove the the case for the Σ_k^{MM} levels, the proof is analogous for the Π_k^{MM} levels.

Clearly every function expressed using k alternating bounded quantifiers on a polynomial time computation R, can be computed by a k-alternating max-min transducer, and therefore is in Σ_k^{MM}.

For the other direction, let f be a k-alternating max-min function and let N be the k-alternating max-min machine computing f. Also, on input x, let p be a polynomial bounding the length of the longest sequence, over all computation paths of N, of identical gates along a single path. By definition, as N's running time is polynomially bounded, such a p exists.

Define N', another k-alternating max-min machine, that "expands" the computational paths of N so that the computation tree is a full binary tree of depth $k \times p(|x|)$. Informally, on input x, N' guesses a string Y of this length and uses it to simulate N as follows: Break Y into k pieces $y_1, \ldots, y_k$, $|y_i| = p(|x|)$ and use y_1's bits for the nondeterministic choices to walk down a path of N until we reach the first min gate on this path. Discard the rest of y_1, and use y_2 until the next alternation, and so on until we have reached the bottom of N's computation tree. Output the same value as N does on this path.

We can now define the function R such that $R(x, y_1, \ldots, y_k) =$ "the value output on the computation path of N using guess string $Y = y_1 \ldots y_k$", as described above. Note that R is in PF. Clearly N' and N compute the same value and the proposition follows from the definition of a k-alternating max-min function. □

Proposition A.2 *(Proposition 3.4) For every $k \geq 0$*

1. $\mathrm{F}\Delta_1^P = \mathrm{PF}$.
2. $\Sigma_k^{MM} = \max \; \Sigma_k^{MM}$.
3. $\Pi_k^{MM} = \min \; \Pi_k^{MM}$.

4. $\mathrm{F}\Delta_{k+1}^P = \mathrm{PF}^{\Pi_k^P}$.

5. If $f \in \mathrm{F}\Delta_k^P$ *and if* g *is metric reducible to* f*, then* $g \in \mathrm{F}\Delta_k^P$.

6. If $\Pi_k^{MM} \subseteq \Sigma_k^{MM}$ *then,* $\forall i \geq k,\ \Sigma_i^{MM} = \Sigma_k^{MM}$.

7. If $\Sigma_k^{MM} \subseteq \Pi_k^{MM}$ *then,* $\forall i \geq k,\ \Pi_i^{MM} = \Pi_k^{MM}$.

Proof. (1) holds as $\mathrm{F}\Delta_1^P = \mathrm{PF}^{\mathrm{P}} = \mathrm{PF}$. (2) and (3) hold because we may merge consecutive identical quantifiers, replacing them by a single quantifier bounded by a higher polynomial, and appropriately modify the function being quantified. (4) holds because we may switch the states we go to on a query, on the basis of the answer being a YES or a NO and instead of querying the oracle $A \in \Sigma_k^P$, query $\overline{A} \in \Pi_k^P$. (5) holds as $\mathrm{PF}(\mathrm{PF}^{\Sigma_k^P}) = \mathrm{PF}(\Sigma_k^P)$. (6) holds as follows: Let $f \in \Sigma_{k+1}^{MM}$. Then $f = \max_p y\ [g(x,y)]$, for some $g \in \Pi_k^{MM}$. By hypothesis there exists some $g' \in \Sigma_k^{MM}$ such that $\forall x, y,\ g(x,y) = g'(x,y)$. By (2) the first quantifier of g' can be absorbed into the first quantifier for f, for some higher choice of polynomial. Therefore $f \in \Sigma_k^{MM}$. (7) is similar. □

A.2 Proofs from Section 4

We present proofs on inclusion relationships in PFH.

Proposition A.3 *(Proposition 4.6) Let* $k \geq 1$ *and let* $g \in \Sigma_k^{MM} \cup \Pi_k^{MM}$ *be bounded in length by some polynomial* p*, and let* g *be defined as* $g(x) = Q_1\ y_1 \ldots Q_k\ y_k\ [R(x, y_1, \ldots, y_k)]$*. Let* g' *be the function defined by*

$$g'(x) = 2^{p(|x|)} - (\overline{Q_1}\ y_1 \ldots \overline{Q_k}\ y_k\ [2^{p(|x|)} - R(x, y_1, \ldots, y_k)])$$

Then $g(x) = g'(x)$.

Proof. We proceed by induction.

Basis 1. Let $g(x) = \max_p y\ [R(x,y)]$. By the definition of maximum and minimum, $g(x) = (2^{p(|x|)} - \min_p y\ [2^{p(|x|)} - R(x,y)]) = g'(x)$.

2. Let $g(x) = \min_p y\ [R(x,y)]$. Again, by the definition of maximum and minimum, $g(x) = (2^{p(|x|)} - \max_p y\ [2^{p(|x|)} - R(x,y)]) = g'(x)$.

Induction hypothesis Let $g \in \Sigma_{k-1}^{MM} \cup \Pi_{k-1}^{MM}$ be defined

$$g(x) = Q_1\ y_1 \ldots Q_{k-1}\ y_{k-1}\ R'(x, y_1, \ldots, y_{k-1})$$

where $|g(x)| \leq p(|x|)$, and let g' be the function defined by

$$g'(x) = 2^{p(|x|)} - (\overline{Q_1}\ y_1 \ldots \overline{Q_{k-1}}\ y_{k-1}\ [2^{p(|x|)} - R'(x, y_1, \ldots, y_{k-1})])$$

Then $g(x) = g'(x)$.

Induction step Let $g \in \Sigma_k^{MM}$ (the case for $g \in \Pi_k^{MM}$ is similar):

$$g(x) = \max_p\ y_1\ \min_p\ y_2\ \ldots Q_k\ y_k\ [R(x, y_1, \ldots, y_k)]$$

Consider the 2-ary function (with inputs x and y_1) defined by the sequence of quantifiers from $\min_p\ y_2$ onwards on R. If we substitute another PF function R' for R, where $R'(x, y_1, \ldots, y_k) = 2^{p(|x|)} - R(x, y_1, \ldots, y_k)$, and re-organize the terms, using the induction hypothesis, we obtain $g(x)$

$$\begin{aligned} &= \max_p\ y_1 (2^{p(|x|)} - (\max_p\ y_2 \ldots \overline{Q_k}\ y_k\ [2^{p(|x|)} - R(x, \ldots, y_k)])) \\ &= 2^{p(|x|)} - (\min_p\ y_1 \ldots (\overline{Q_k})_{p(|x|)}\ y_k\ [2^{p(|x|)} - R(x, y_1, \ldots, y_k)]) \end{aligned}$$

by the definition of maximum and minimum, as in the basis.

□

For completeness we include the lemma from [9] in its entirety.

Theorem A.4 *(Theorem 4.9)[9] Let $f \in \mathrm{F}\Delta_{k+1}^P$. Then f can be expressed as $f(x) = h(x, g(x))$ where $g \in \Sigma_k^{MM}$ and $h : \Sigma^* \times \mathcal{N} \to \mathcal{N}$ is computable in polynomial time.*

Proof. Let f be a function in $\mathrm{F}\Delta_{k+1}^P$. Then there is a deterministic oracle Turing transducer M and a set $A \in \Sigma_k^P$ such that M, using A as an oracle, computes f. Define $g(x)$ to be the correct sequence of oracle answers, interpreted as a binary number, in M's computation on x relative to A. Clearly it suffices to show $g \in \Sigma_k^{MM}$. To avoid any problems with leading zeros in $g(x)$, modify M so that it first asks a dummy query whose answer is always 'yes'.

Suppose M runs in polynomial time $p(n)$ for some polynomial p. Let $m = p(|x|)$ and let $w = w_1 w_2 \ldots w_m \in \{0,1\}^m$ be a sequence of possible oracle answers for $M^A(x)$. The question "given w and x, simulate $M(x)$, using $w_1 w_2 \dots w_m$ for the oracle answers, compute all of the queries and ask if all of the 'yes' instances are correct" is a single Σ_k^P question. Thus there is a polynomial time computable predicate R such that "all of the 'yes' answers in $w_1 w_2 \dots w_m$ are correct" if and only if

$$\exists_{|y_1| \leq m}\, y_1\, \forall_{|y_2| \leq m}\, y_2 \dots (Q_k)_{|y_k| \leq m}\, y_k\; R(x, w, y_1, y_2, \dots, y_k)$$

where Q_k depends on k.

Now we claim that the maximum $w_1 \dots w_m$ such that all of the 'yes' answers are are correct is the correct sequence of oracle answers for $M^A(x)$. Consider w_1 first. If the answer to M's first query is 'yes' then all of the 'yes' answers are correct in the sequence $10 \dots 0$, so $w_1 = 1$. Conversely, if M's first oracle answer is a 'no' then the maximum can be no more than $01 \dots 1$, so $w_1 = 0$. In either case the value of w_1 is correct. By the same argument, the value of w_2 is correct, given that w_1 is correct. Continuing by induction, $w_1, \dots, w_m$ is the correct sequence of oracle answers.

We can now compute the oracle answers for $M(x)$ by the following alternating max-min Turing machine. Find the maximum w such that $|w| = m$ and

$$\exists_{|y_1| \leq m}\, y_1\, \forall_{|y_2| \leq m}\, y_2 \dots (Q_k)_{|y_k| \leq m}\, y_k\; R(x, w, y_1, y_2, \dots, y_k).$$

holds. We can replace each $\exists_{|y_i| \leq m}\, y_i$ by a *max* and each $\forall_{|y_i| \leq m}\, y_i$ by a *min* and compute the following PF function on each path.

Output w if $R(x, w, y_1, y_2 \dots y_k)$ is true or else output 0.

Note that the first two optimizations (on w and on y_1) are both *max* and therefore contribute only one alternation; the remaining quantifiers yield $k-1$ alternations, hence this function is in Σ_k^{MM}. $\square$

Lemma A.5 *(Lemma 4.13)* $\Sigma_k^{MM} \subseteq \mathrm{F}\Delta_{k+1}^P$.

Proof. We proceed by induction

Basis $\Sigma_1^{MM} \subseteq \mathrm{F}\Delta_2^P$ is shown [8].

Induction hypothesis $\Sigma_{k-1}^{MM} \subseteq \mathrm{F}\Delta_k^P$

Induction step Let $f \in \Sigma_k^{MM}$ be defined as

$$f(x) = \max_p\ y_1 \ldots Q_k\ y_k\ [R(x, y_1, \ldots, y_k)]$$

Then we can abbreviate this to $f(x) = \max_p\ y_1\ [g(x, y_1)]$, where $g \in \Pi_{k-1}^{MM}$. As previously shown in Lemma 4.7, g is metric reducible to a g' in Σ_{k-1}^{MM}. Therefore, by induction hypothesis, $g' \in \mathrm{F}\Delta_k^P$ and by part 5 of Proposition 3.4, $g \in \mathrm{F}\Delta_k^P$.

Define language L as follows:

$$L = \{\langle x, v\rangle \mid \exists\ y \in \Sigma^{=p(|x|)}[g(x, y) \geq v]\}$$

Since $L \in \mathrm{NP}^g$ and $g \in \mathrm{PF}^{\Sigma_{k-1}^P}$, it follows that $L \in \mathrm{NP}^{\Sigma_{k-1}^P} = \Sigma_k^P$ As $f(x) = v$ such that $\langle x, v\rangle \in L$ but $\langle x, v-1\rangle \notin L$, we need only do binary search over the range $[\langle x, 0\rangle \ldots \langle x, 2^{kp(|x|)}\rangle]$ of query strings to L. Hence f is in $\mathrm{F}\Delta_{k+1}^P$.

□

A.3 Proof of Theorem 4.17 in Section 4

Definition A.6 *[11] Let ϕ be a universal partial function computed by transducer T_ϕ, let t be a time bound, and s be a space bound. The resource bounded Kolmogorov complexity of x conditional to ϕ, denoted $C_\phi^{t,s}(x)$, is the length of the shortest program p such that $\phi(p)$ is x. The computation of ϕ by T_ϕ runs in time $t(|x|)$ and space $s(|x|)$.*

Definition A.7 *[11]* $\mathrm{K}[l(n), t(n)]$ *is the set of strings x, such that $|C_\phi^{t(n),\infty}(x)| \leq l(n)$. A string $w \in \Sigma^{=n}$ is* incompressible *if no program of length $n - 2\log n$ exists that produces w from the empty input string; i.e., $w \notin \mathrm{K}[n - 2\log n, \infty]$.*

Recall that FDEXT is the class of functions computable in time $2^{\mathcal{O}(n)}$.

Theorem A.8 *(Theorem 4.17) There exist functions f and g in* FDEXT *such that f is Turing reducible to g, but f is not metric reducible to g.*

Proof. Note that $f \in \mathrm{PF}^{g[1]}$ if f is metric reducible to g, and that $f \in \mathrm{PF}^g$ if f is Turing reducible to g. Let $f(0^n)$ be the lexicographically first string of length n not in $\mathrm{K}[\log n, 2^n]$. The function f is in FDEXT, as we need only run the n programs of length $\log n$, for 2^n steps each, to produce $S = \mathrm{K}[\log n, 2^n]$, from which it is easy to pick the lexicographically least $x \notin S$. Let $g(0^{\langle n,i\rangle})$ be the parity of the first i bits of $f(0^n)$. The function g is also in FDEXT, as we need only compute f and count the parity of its first i bits. Then $f \in \mathrm{PF}^g$ as the values of $g(0^{\langle n,1\rangle}), g(0^{\langle n,2\rangle}), \ldots, g(0^{\langle n,n\rangle})$ determine $f(0^n)$. Assume f is metric reducible to g. Then there exist PF functions $T_1 : \Sigma^* \to \mathcal{N}$ and $T_2 : (\Sigma^*)^2 \to \mathcal{N}$ such that $f(x) = T_1(x, g(T_2(x)))$. Also, there exist machines M_1 and M_2 bounded by polynomial p that compute T_1 and T_2 respectively. For each $n \in \mathcal{N}$, the following is a short description of a program that produces $f(0^n)$:

1. $\log n$ bits for n,
2. a constant number of bits to describe M_1, M_2 and this description, and
3. a single bit b to represent the value of $g(T_2(0^n))$, and
4. the computation of $f(0^n) = T_1(0^n, b)$.

Thus $f(0^n)$ is in $\mathrm{K}[c, p(n)]$. But for large enough n, $c < \log n$ and $p(n) < 2^n$, contradicting the incompressibility of $f(0^n)$. Hence f is not metric reducible to g. $\square$

A.4 Proofs from Section 6

Lemma A.9 *(Lemma 6.4)* PP(NP) = NP $\iff$ PH(PP(NP)) = NP.

Proof. Assume PP(NP) = NP. Lemma 6.3, together with our hypothesis, yields PP $\subseteq$ NP $\implies$ PH = NP. Also PP(NP) = NP implies PH(PP(NP)) $\subseteq$ PH(NP) $\subseteq$ PH. Since NP $\subseteq$ PH(PP(NP)), we have PH(PP(NP)) = NP. Combining the above we get PP(NP) = NP $\iff$ PH(PP(NP)) = NP. $\square$

Lemma A.10 *(Lemma 6.5)* PP(NP) = P(OptP[1]) $\iff$ PH(PP(NP)) = P(OptP[1]).

Proof. Assume PP(NP) = P(OptP[1]). Lemma 6.3, together with our hypothesis, yield

$$\mathrm{PH} \subseteq \mathrm{P(PP(NP))} = \mathrm{P(P(OptP[1]))} \subseteq \mathrm{P(OptP)} \subseteq \mathrm{P(NP)} \subseteq \mathrm{P(OptP[1])}$$

implying PH $\subseteq$ P(OptP[1]) and thus PH = Δ_2^P. Therefore, the hypothesis PP(NP) = P(OptP[1]) implies:

$$\mathrm{PH(PP(NP))} \subseteq \mathrm{PH(P(OptP[1]))} \subseteq \mathrm{PH} = \mathrm{P(OptP[1])}$$

Since P(OptP[1]) $\subseteq$ PH(PP(NP)), we have PH(PP(NP)) = P(OptP[1]) $\square$