A Complexity Theory for Feasible Closure Properties\textsuperscript{1}

\textit{Mitsunori Ogiwara} \\
Department of Information Sciences \\
Tokyo Institute of Technology \\
Tokyo 152

\textit{Lane A. Hemachandra\textsuperscript{2}} \\
Department of Computer Science \\
University of Rochester \\
Rochester, NY 14627

March, 1991

\textsuperscript{1}This paper incorporates and extends the research described in "A Complexity Theory for Closure Properties," M. Ogiwara and L. Hemachandra, Department of Information Science Technical Report C-99, Tokyo Institute of Technology, 1990.

\textsuperscript{2}Research supported in part by the National Science Foundation under grants CCR-8957604 and CCR-8996198, and by the International Information Science Foundation under grant 90-1-3-228.
Abstract

The study of the complexity of sets encompasses two complementary aims: (1) establishing—usually via explicit construction of algorithms—that sets are feasible, and (2) studying the relative complexity of sets that plausibly might be feasible but are not currently known to be feasible (such as the NP-complete sets and the PSPACE-complete sets).

For the study of the complexity of closure properties, a recent flurry of results [GNW90, Ogi90, Tod90b, BHW91, BRS, FR] has established an analog of (1); these papers explicitly demonstrate many closure properties possessed by PP and C=P (and the proofs implicitly give closure properties of the function class #P). The present paper presents and develops, for function classes such as #P, SpanP, OptP, and MidP, an analog of (2): a general theory of the complexity of closure properties.

In particular, we show that subtraction is hard for the closure properties of each of these classes: each is closed under subtraction if and only if it is closed under every polynomial-time operation. Previously, no property—natural or unnatural—had been known to have this behavior. We also prove other natural operations hard for the closure properties of #P, SpanP, OptP, and MidP, and we explore the relative complexity of operations that seem not to be #P-hard, such as maximum, minimum, decrement, and median. Moreover, for each of #P, SpanP, OptP, and MidP, we give a natural complete characterization—in terms of the collapse of complexity classes—of the conditions under which that class has every feasible closure property.
1 Introduction

Motivated by the modern theory of the complexity of sets, and in particular by NP-completeness/NP-hardness theory, this paper constructs an analogous general theory of the complexity of closure properties. We formalize the notion of closure property, and introduce the notion of a hard closure property for a class—a property so hard that the class has that closure property if and only if it has every feasible closure property.

We will focus our attention on the following classes. (1) $\#P$: $\#P$ is Valiant's [Val79] class of counting functions, which has recently been shown by Toda to be deeply related to the structure of the polynomial hierarchy [Tod89]. (2) SpanP: SpanP is the class of functions counting the number of distinct outputs of polynomial-time nondeterministic Turing machines [KST89,K89,Sch90]. (3) OptP: OptP is the class of functions computing the maximum value of the outputs of polynomial-time nondeterministic Turing machines; OptP is a formalization of the intuitive notion of an NP optimization problem [Kre88]. (4) MidP: MidP is the class of functions computing the middle (median) value of the outputs of polynomial-time nondeterministic Turing machines; Toda has shown that a single access to a MidP oracle suffices to simulate the polynomial hierarchy [Tod90a].

There are many closure properties that these classes provably have, and many that the classes have not yet been proven to have. The former type of property—closure properties that a class $\mathcal{F}$ provably possesses—has been exhaustively studied, yet few nontrivial natural closure properties of them have been found. It is the latter type of property—properties that classes have not yet been shown to possess—on which we will concentrate in this paper. What is the relative complexity of such closure properties? Is there some such closure property that is the "hardest," in the sense that if it is easy, then all other closure properties are easy? If so, is there a natural closure property of this form? The analogy to the (nonconstructive) theory of NP-hardness will soon be made explicit. Before further discussion, we introduce and justify our basic definitions.

Definition 1.1 1. Let $f$ be a function such that, for some $i$, $f$ maps from $\mathbb{N}^i$ to $\mathbb{N}$. We say that $f$ is a closure property (of arity $i$).

2. Let $\mathcal{CF}$ be a class of functions. We say that a closure property $f$ is a $\mathcal{CF}$-closure property if $f \in \mathcal{CF}$. As a shorthand, we'll use "P-closure property," rather than "PF-closure property," to describe closure properties computable in polynomial time.

\footnote{For example, in [Reg82,CGH89,BGH90]; note also the striking recent closure results for the language classes $\text{PP}$ [BRS,BHW91,FR] and $\text{C}_{\text{app}}$, $\text{P}$ [GNW90,OG90,Tod90].}
3. Let $\mathcal{DF}$ be a class of functions. We say that $\mathcal{DF}$ has closure property $f$ (of arity $i$) if:

$$g_1, \ldots, g_i \in \mathcal{DF} \implies h \in \mathcal{DF},$$

where $h(x) = f(g_1(x), \ldots, g_i(x))$.

4. Let $\mathcal{DF}$ and $\mathcal{CF}$ be classes of functions. We say that $\mathcal{DF}$ is $\mathcal{CF}$-closed if $\mathcal{DF}$ has every $\mathcal{CF}$-closure property.

5. Let $\mathcal{DF}$ and $\mathcal{CF}$ be classes of functions. Let $f$ be a $\mathcal{CF}$-closure property. We say that $f$ is hard for the $\mathcal{CF}$-closure properties of $\mathcal{DF}$ (for short, "a $\mathcal{DF}$-hard $\mathcal{CF}$-closure property," or, in the case that $\mathcal{CF} = \mathcal{PF}$, simply "$\mathcal{DF}$-hard") if it holds that: $\mathcal{DF}$ has closure property $f \iff \mathcal{DF}$ is $\mathcal{CF}$-closed.

Having proposed these definitions, we are now obligated to argue that they are natural and capture the intuitively appropriate notions. We first address two most pressing concerns: first, that the notion of closure property proposed above is too broad, and second, that the nonconstructive nature of $\mathcal{F}$-hardness is problematic.

The first concern is easily seen to be insubstantial. On one hand, broad and flexible formalization of new notions is both routine and desirable; Ladner, Lynch, and Selman’s abstract definition of truth-table reductions comes immediately to mind as a sterling example [LLS75]. Though a more restrictive notion might capture a few specific closure properties, allowing general function composition permits the full range of computational possibilities to be included in our notion of closure property. Note also that adopting a broad definition of closure property makes the task of this paper more challenging; the broader the notion of closure property, the harder it is to establish the claim that a given closure property is a hard closure property.

We now turn to the second point: the nonconstructive nature of Part 5 of Definition 1.1. Nonconstructive definitions can indeed cause problems. For example, it is a simple logical truth that if $\mathcal{DF}$ is not $\mathcal{CF}$-closed, then our definition causes every $\mathcal{CF}$-closure property not possessed by $\mathcal{DF}$ to (trivially) be hard for the $\mathcal{CF}$-closure properties of $\mathcal{DF}$. We argue that this is not a flaw in the definition. It seems a flaw only under a misapplication of the definition. Since our definition parallels the nonconstructive definition of NP-hardness, and since to our knowledge a detailed discussion of this objection has not appeared in the literature, we defend our notion by defending the definition of nonconstructive NP-completeness.
Sahni ([Sah74], see also [SG76]) defines a set $S \in \text{NP}$ to be nonconstructively NP-complete if it holds that: $S \in \text{P} \iff \text{P = NP}$.

As first noted by Book, Wrathall, Selman, and Dobkin [BWSD77]: if $\text{P} \neq \text{NP}$, then every set in $\text{NP} - \text{P}$ is nonconstructively NP-complete; since it is well-known that if $\text{P} \neq \text{NP}$ there exist sets in $\text{NP} - \text{P}$ that are not complete in the standard sense [Lad75], this might seem to argue against the naturalness of Sahni's definition. However, such an argument blurs a crucial distinction: the difference between the sets that in fact are nonconstructively NP-complete (regardless of whether we know this fact), and the sets that we—using current techniques—can prove to be nonconstructively NP-complete. Sahni's definition—and also the definition of this paper—clearly was done with the latter interpretation in mind, rather than the former. The objection raised, however, clearly views the definitions within the former model. Thus, nonconstructive definitions—both of NP-completeness and of hardness of closure properties—seek to give evidence of the complexity of a set or a property, and do so within the vocabularies of evidence available today to complexity theorists.

Thus, in the case of interest to this paper, it is irrelevant that all $\text{P}$-closure properties that a class $\mathcal{F}$ lacks are (trivially) $\mathcal{F}$-hard if $\mathcal{F}$ is not $\text{P}$-closed. What is relevant is that, when we prove that a given closure property is $\mathcal{F}$-hard, we have indeed given evidence that it is the logically least likely of closure properties.

Finally, we must note that the term nonconstructive is itself misleading, as applied to the case of $\mathcal{F}$-hard closure properties. This point is somewhat subtle; we completely characterize the question of whether $\mathcal{F}$ is $\text{P}$-closed in terms of a complexity class relation, and in fact, our proofs imply that if that relation holds constructively, then there is a constructive, uniform process for implementing all $\text{P}$-closure properties. Thus, though the definition of nonconstructive hardness is indeed nonconstructive, the implications of nonconstructive hardness are themselves constructive.

---

2In fact, the actual term Sahni used was "P-complete." At the time his paper was first drafted (1973), the term "NP-complete" had not yet been proposed by Knuth [Knu74]. However, Sahni's definition was intended as an alternative to Karp's reduction-based notion of what we now call NP-completeness, and which Sahni calls polynomial-completeness. The relationship between nonconstructive and constructive notions of NP-completeness has since been studied by Book, Wrathall, Selman, and Dobkin [BWSD77] and Geske, Huynh, and Selman [GHS87]. One should note that closure properties are quite different than languages; though one could build a constructive, reduction-based theory of hard closure properties (namely, by discussing reductions from properties to properties, where a property would be embodied as a machine that itself mapped from machine names to machine names), this would be an unnatural and awkward approach. Nonetheless, it is not hard to see that the proofs in this paper "preserve constructivity," and such a reduction-based theory could be built upon our techniques.
The preceding discussion is without point unless there are some P-closure properties of \( \mathcal{F} \) that we can prove \( \mathcal{F} \)-hard. Are there? And if so, are there natural \( \mathcal{F} \)-hard P-closure properties? It is not apparent, \textit{a priori}, that any one property should be so complex that its possession would cause \( \mathcal{F} \) to have all closure properties. Nonetheless, the main result of this paper is that subtraction and division are both hard for the P-closure properties of \( \#P \), as well as for SpanP, OptP, and a certain version of MidP. In fact, as a technical aside, our proofs establish that for any one of these classes (call it \( \mathcal{F} \)), it holds that \( \mathcal{F} \) has all P-closure properties if and only if \( \mathcal{F} \) has all \( \mathcal{F} \)-closure properties; thus, subtraction and division are in fact hard for the \( \mathcal{F} \)-closure properties of \( \mathcal{F} \).

Our proof proceeds by identifying structural conditions that completely characterize the question of whether a class \( \mathcal{F} \) has all P-closure properties; we then show that this condition is equivalent to closures under subtraction, division, and so on. In particular: \( \#P \) has all P-closure properties if and only if \( PP = UP \); SpanP has all P-closure properties if and only if R-MidP has all P-closure properties if and only if \( P^{PP} = NP \) (equivalently, \( C \cdot NP = NP \)); OptP has all P-closure properties if and only if \( NP = co-NP \).

Just as, in the study of the class \( NP \), the structure of (potentially) incomplete sets has been of great interest [Lad75,CM81,LLR81,AS85,AH87], so also may we be interested in P-closure properties that seem to be neither possessed by a class nor hard for the P-closure properties of the class. For the class \( \#P \), among such closure properties are minimum, maximum, median, and decrement. For such operations, we explore both the implications that would follow were \( \#P \) to have such closure properties, and the relative complexities of the properties. We show, for example, that \( \#P \) is closed under maximum and minimum if and only if \( \#P \) is closed under median.

The paper is organized as follows. Section 2 defines the closure properties and complexity classes that we will be concerned with in this paper. Section 3 establishes the existence of \( \#P \)-hard closure properties and explores the structure of closure properties of seemingly intermediate complexity. Sections 4, 5 and 6 establish the existence of SpanP-hard, OptP-hard, and MidP-hard closure properties, respectively.

2 Preliminaries

Throughout this paper, we use the finite alphabet \( \Sigma = \{0, 1\} \). All the languages we will discuss are subsets of \( \Sigma^* \). For each string \( x \in \Sigma^* \), \( |x| \) denotes the length of \( x \). For each natural number \( n \), \( \Sigma^{=n} \) denotes the set of all strings in \( \Sigma^* \) of length exactly \( n \). For
any set \(L\), \(\overline{L}\) denotes the complement of \(L\), namely \(\Sigma^* - L\). For any class \(K\), \(\text{co}K\) denotes \(\{L \mid \overline{L} \in K\}\). For each set \(S\), \(#S\) denotes the cardinality of \(S\). We assume that the reader is familiar with the basic notions of complexity theory, such as \(P\), \(NP\), and so on [HU79,GJ79].

Our computation model is the nondeterministic Turing machine. We adopt the standard model in which, for every nondeterministic polynomial-time Turing machine \(M\), there exist two polynomials \(p_M\) and \(q_M\) such that for every \(x \in \Sigma^*\), (1) \(M\) on \(x\) eventually halts after executing at most \(p_M(|x|)\) steps, and (2) each computation path of \(M(x)\), has exactly \(q_M(|x|)\) nondeterministic steps. Let \(N\) denote the set of natural numbers. Let \(\langle \cdot, \cdot \rangle_2\) denote some standard pairing function over finite strings that is polynomial-time computable and polynomial-time invertible, and such that for every \(x, x', y, y' \in \Sigma^*\) such that \(|x| = |x'|\) and \(|y| = |y'|\), it holds that \(|(x, y)_2| = |(x', y')_2|\). Let \(\langle y \rangle\) denote \(\langle 1, y \rangle_2\) and, for each \(k \geq 2\), let \(\langle y_1, \ldots, y_k \rangle\) denote \(\langle k, \langle y_1, \langle \ldots, \langle y_{k-1}, y_k \rangle_2 \rangle_2 \rangle_2\). We adopt the standard lexicographical ordering of \(\Sigma^*\): for strings \(x\) and \(y\) in \(\Sigma^*\), \(x\) is lexicographically smaller than \(y\), denoted by \(x < y\), if either (1) \(|x| < |y|\) or (2) \(|x| = |y|\) and there exists \(i\) such that for every \(j, 1 \leq j < i\), \(x_j = y_j\) and \(x_i = 0\) and \(y_i = 1\), where \(x_i\) (respectively, \(y_i\)) denotes the \(i\)th symbol of \(x\) (respectively, \(y\)). For \(x \in \Sigma^*\), \(\text{ord}(x)\) denotes the number of strings \(y \in \Sigma^*\) that are smaller than \(x\). For every string \(x\), identify \(x\) and its order \(\text{ord}(x)\), thus establishing an easily computable 1-1 correspondence between \(\Sigma^*\) and \(N\).

\(\text{PF}\) denotes the class of all polynomial-time computable functions from \(\Sigma^*\) to \(N\). For \(k \in N\), \(C_k\) denotes the constant function that maps every \(x \in \Sigma^*\) to \(k\). For a nondeterministic polynomial-time Turing machine \(M\), \(#\text{acc}_M\) (respectively, \(#\text{rej}_M\), \(#\text{total}_M\)) denotes the function that maps \(x \in \Sigma^*\) to the number of accepting computation paths (respectively, to the number of rejecting computation paths, to the total number of computation paths) of \(M\) on \(x\).

**Definition 2.1** [Val79]
\[\#P = \{\#\text{acc}_M \mid M \text{ is a nondeterministic polynomial-time Turing machine}\}.

We review the definitions of some complexity classes that we will discuss in this paper.

**Definition 2.2**
1. [Coo71,Lev73] \(NP\) is the class of sets \(L\) for which there exists a function \(f \in \#P\) such that for every \(x \in \Sigma^*\), \(x \in L \iff f(x) > 0\).

2. [Val76] \(UP\) is the class of sets \(L\) for which there exists a function \(f \in \#P\) such that for every \(x \in \Sigma^*\), (i) \(x \in L \iff f(x) = 1\), and (ii) \(x \notin L \iff f(x) = 0\).
3. [Sim75,Wag86] C=P is the class of sets $L$ for which there exist functions $f \in \#P$ and $g \in PF$ such that for every $x \in \Sigma^*$, $x \in L \iff f(x) = g(x)$.

4. [Sim75,Gil77] PP is the class of sets $L$ for which there exist functions $f \in \#P$ and $g \in PF$ such that for every $x \in \Sigma^*$, $x \in L \iff f(x) \geq g(x)$.

5. [PZ83,GP86] \(\oplus P\) is the class of sets $L$ for which there exists a function $f \in \#P$ such that for every $x \in \Sigma^*$, $x \in L \iff f(x)$ is odd.

Next, we review the definitions of some operators that we will use in this paper.

**Definition 2.3**

1. For a class $K$, \(\exists K\) is the class defined in the following way: A set $L$ is in \(\exists K\) if there exist a polynomial $p$ and a set $A \in K$ such that for every $x \in \Sigma^*$,

   $$x \in L \iff \#\{y \mid |y| = p(|x|) \text{ and } (x, y) \in A\} \geq 1.$$  

2. For a class $K$, \(\forall K\) is the class defined in the following way: A set $L$ is in \(\forall K\) if there exist a polynomial $p$ and a set $A \in K$ such that for every $x \in \Sigma^*$,

   $$x \in L \iff \#\{y \mid |y| = p(|x|) \text{ and } (x, y) \in A\} = 2^{p(|x|)}.$$  

3. For a class $K$, \(C \cdot K\) is the class defined in the following way: A set $L$ is in \(C \cdot K\) if there exist a polynomial $p$, a function $f \in PF$, and a set $A \in K$ such that for every $x \in \Sigma^*$,

   $$x \in L \iff \#\{y \mid |y| = p(|x|) \text{ and } (x, y) \in A\} \geq f(x).$$  

4. For a class $K$, \(C_\leq K\) is the class defined in the following way: A set $L$ is in \(C_\leq K\) if there exist a polynomial $p$, a function $f \in PF$, and a set $A \in K$ such that for every $x \in \Sigma^*$,

   $$x \in L \iff \#\{y \mid |y| = p(|x|) \text{ and } (x, y) \in A\} = f(x).$$  

5. For a class $K$, \(BP \cdot K\) is the class defined in the following way: A set $L$ is in \(BP \cdot K\) if there exist a polynomial $p$ and a set $A \in K$ such that for every $x \in \Sigma^*$,

   $$x \in L \iff \#\{y \mid |y| = p(|x|) \text{ and } (x, y) \in A\} \geq \frac{3}{4} \cdot 2^{p(|x|)}, \text{ and}$$

   $$x \not\in L \iff \#\{y \mid |y| = p(|x|) \text{ and } (x, y) \in A\} \leq \frac{1}{4} \cdot 2^{p(|x|)}.$$
It is well-known, and not hard to see, that $\exists \cdot P = \text{NP}$, $\forall \cdot P = \text{coNP}$, $C \cdot P = \text{PP}$, and $C_\varepsilon \cdot P = C_\varepsilon \text{P}$.

Below are definitions of two standard hierarchies, the polynomial hierarchy and the counting polynomial hierarchy.

**Definition 2.4** [Sto77] The polynomial hierarchy is the class of sets defined in the following way:

1. $\Sigma_0^P = \Pi_0^P = P$.
2. For $k \geq 1$: $\Sigma_k^P = \exists \cdot \Pi_{k-1}^P$ and $\Pi_k^P = \forall \cdot \Sigma_{k-1}^P$.
3. $\text{PH} = \bigcup_{k \geq 0} \Sigma_k^P = \bigcup_{k \geq 0} \Pi_k^P$.

**Definition 2.5** [Wag86, Tor88a] The counting polynomial hierarchy is the class of sets defined in the following way:

1. $P$ is in the counting polynomial hierarchy.
2. For any class $K$ in the counting polynomial hierarchy, $\exists \cdot K$, $\forall \cdot K$, $C \cdot K$, and $C_\varepsilon \cdot K$ are in the counting polynomial hierarchy.
3. $\text{CH}$, the counting polynomial hierarchy, consists of only the sets described by the previous two items.

The following properties are known to follow immediately from the definitions.

**Proposition 2.6**

1. $\text{UP} \subseteq (\text{NP} \cap \oplus P \cap C_\varepsilon \text{P})$.
2. $\text{coNP} \subseteq C_\varepsilon \text{P}$.
3. $\text{NP} \subseteq \text{PP}$.
4. $\text{PH} \subseteq \text{CH}$.
5. $\oplus P \subseteq P_{\text{PP}} = \text{P}^\text{P}$.
6. For every class $K$, $\exists \cdot K \subseteq \text{NP}^K = \exists \cdot P^K$, $C \cdot K \subseteq \text{PP}^K = C \cdot P^K$, and $C_\varepsilon \cdot K \subseteq C_\varepsilon \cdot P^K = C_\varepsilon \cdot P^K$.

The following results are well-known or easy to see.

**Proposition 2.7**

1. [Tod89] $\text{PH} \subseteq P_{\text{PP}}$.
2. [Rus85, Sim75, Tor88a] $C_\varepsilon \text{P} \subseteq \text{PP} \subseteq C_\varepsilon \cdot C_\varepsilon \text{P}$.
3. [PZ83] $\oplus P = P = \oplus P$.

4. [Sim75] $L$ is in $C_P$ if and only if there exist a nondeterministic polynomial-time Turing machine $M$ and a polynomial $p$ such that for every $x \in \Sigma^*$,
   
   (a) $\#_{\text{total}}_M(x) = 2^p(|x|)+1$,
   
   (b) $x \in L$ if and only if $\#_{\text{acc}}_M(x) = 2^p(|x|)$, and
   
   (c) $x \notin L$ if and only if $0 < \#_{\text{acc}}_M(x) < 2^p(|x|)$.

5. [Tor88a] $C \cdot C \cdot P = C \cdot C \cdot P$ and $C \cdot C \cdot P = C \cdot C \cdot P$.

6. $CH = \bigcup_{k \geq 0} C \cdot \cdots \cdot C \cdot P$. 

7. [KSTT89] $PP^{UP} = PP$ and $C_P^{UP} = C_P$.

8. [TO] $PP^{PH} \subseteq BP \cdot PP$ and $\oplus P^{PH} \subseteq BP \cdot \oplus P$.

9. [Sch87] $BP \cdot PH \subseteq PH$.

10. [FR] $PP$ is closed under truth-table reductions.

11. [GNW90] $C \cdot P$ is closed under positive truth-table reductions.

Next, we define the closure properties that we will consider.

**Definition 2.8** 1. A function class $F$ is closed under division by polynomial-time computable functions (respectively, $F$ functions) if for every $f \in F$ and for every nonzero function $g \in PF$ (respectively, $g \in F$) there is a function $h \in F$ such that for every $x \in \Sigma^*$, it holds that $h(x) = \lfloor f(x)/g(x) \rfloor$.

2. A function class $F$ is closed under division by 2 if for every $f \in F$ there is a function $g \in F$ such that for every $x \in \Sigma^*$, it holds that $h(x) = \lfloor f(x)/2 \rfloor$.

**Definition 2.9** 1. For any two numbers $a, b \in \mathbb{N}$, let $a \oplus b$ denote $\max\{0, a - b\}$.

2. A function class $F$ is closed under subtraction by polynomial-time computable functions (respectively, $F$ functions) if for every $f \in F$ and for every function $g \in PF$ (respectively, $g \in F$) there is a function $h \in F$ such that for every $x \in \Sigma^*$, it holds that $h(x) = f(x) \ominus g(x)$.

---

5The reader is cautioned that there is an exceedingly minor, easily corrected, arithmetic error in the proof found in [Sim75, p. 94].

6This characterization is immediate, since for all classes $K$ in $CH$, (1) $(\forall \cdot K) \cup (\exists \cdot K) \subseteq C \cdot K$ and (2) analogously to [Sim75, p. 94], $C \cdot K \subseteq C \cdot K$. 

8
3. A function class $\mathcal{F}$ is closed under decrement if for every $f \in \mathcal{F}$ there is a function $g \in \mathcal{F}$ such that for every $x \in \Sigma^*$, it holds that $g(x) = f(x) \ominus 1$.

**Definition 2.10** 1. For each natural number $k \geq 2$ and $a_1, \ldots, a_k \in \mathbb{N}$, let $\text{plu}_k(a_1, \ldots, a_k)$ denote the set of the most commonly occurring number(s) amongst $a_1, \ldots, a_k$; that is, $a \in \text{plu}_k(a_1, \ldots, a_k)$ if and only if for every $b \in \{a_1, \ldots, a_k\}$, it holds that $\# \{i \mid a = a_i\} \geq \# \{i \mid b = a_i\}$.

2. For each natural number $k \geq 2$ and $a_1, \ldots, a_k \in \mathbb{N}$, let $\text{plu}_k^*(a_1, \ldots, a_k)$ denote the smallest value in $\text{plu}_k(a_1, \ldots, a_k)$.

3. For each natural number $k \geq 2$, we say that a function class $\mathcal{F}$ is closed under weak plurality of $k$ functions if for every $f_1, \ldots, f_k \in \mathcal{F}$ there exists a function $g \in \mathcal{F}$ such that for every $x \in \Sigma^*$, it holds that $g(x) \in \text{plu}_k(f_1(x), \ldots, f_k(x))$.

4. For each natural number $k \geq 2$, we say that a function class $\mathcal{F}$ is closed under strong plurality of $k$ functions if for every $f_1, \ldots, f_k \in \mathcal{F}$ there exists a function $g \in \mathcal{F}$ such that for every $x \in \Sigma^*$, it holds that $g(x) = \text{plu}_k^*(f_1(x), \ldots, f_k(x))$.

**Definition 2.11** 1. For each natural number $k \geq 2$ and $a_1, \ldots, a_k \in \mathbb{N}$, let $\text{span}_k(a_1, \ldots, a_k)$ denote the number of distinct values in $\{a_1, \ldots, a_k\}$; that is, $\text{span}_k(a_1, \ldots, a_k) = \# \{a_1, \ldots, a_k\}$.

2. For each natural number $k \geq 2$, we say that a function class $\mathcal{F}$ is closed under span of $k$ functions if for every $f_1, \ldots, f_k \in \mathcal{F}$ there exists a function $g \in \mathcal{F}$ such that for every $x \in \Sigma^*$, it holds that $g(x) = \text{span}_k(f_1(x), \ldots, f_k(x))$.

**Definition 2.12** 1. For each odd natural number $k \geq 3$ and $a_1, \ldots, a_k \in \mathbb{N}$, let $\text{median}_k(a_1, \ldots, a_k)$ denote the median of the numbers $a_1, \ldots, a_k$; that is, for a non-decreasing enumeration $b_1, \ldots, b_k$ of values $a_1, \ldots, a_k$, $\text{median}_k(a_1, \ldots, a_k) = b_{(k+1)/2}$.

2. For each odd natural number $k \geq 3$, we say that a function class $\mathcal{F}$ is closed under median of $k$ functions if for every $f_1, \ldots, f_k \in \mathcal{F}$ there is a function $g \in \mathcal{F}$ such that for every $x \in \Sigma^*$, it holds that $g(x) = \text{median}_k(f_1(x), \ldots, f_k(x))$.

**Definition 2.13** 1. A function class $\mathcal{F}$ is closed under minimum if for every two functions $f, g \in \mathcal{F}$ there is a function $h \in \mathcal{F}$ such that for every $x \in \Sigma^*$, it holds that $h(x) = \min \{f(x), g(x)\}$.

2. A function class $\mathcal{F}$ is closed under maximum if for every two functions $f, g \in \mathcal{F}$ there is a function $h \in \mathcal{F}$ such that for every $x \in \Sigma^*$, it holds that $h(x) = \max \{f(x), g(x)\}$.
3  A Complexity Theory for Closure Properties of \(\#P\)

It is well known that \(\#P\) is closed under addition and multiplication [Reg82], and indeed, \(\#P\) has many other closure properties [CGH+89,BGH90].

**Proposition 3.1**

1. [Reg82] \(\#P\) is closed under multiplication; that is, for every pair of nondeterministic polynomial-time Turing machines, \(M_1\) and \(M_2\), there exists a nondeterministic polynomial-time Turing machine \(N\) such that for every \(x \in \Sigma^*\),

\[
\#acc_N(x) = \#acc_{M_1}(x) \times \#acc_{M_2}(x).
\]

2. [Reg82] \(\#P\) is closed under addition; that is, for every pair of nondeterministic polynomial-time Turing machines, \(M_1\) and \(M_2\), there exists a nondeterministic polynomial-time Turing machine \(N\) such that for every \(x \in \Sigma^*\),

\[
\#acc_N(x) = \#acc_{M_1}(x) + \#acc_{M_2}(x).
\]

3. For every nondeterministic polynomial-time Turing machine \(M\), there exists a nondeterministic polynomial-time Turing machine \(N\) such that for every \(x \in \Sigma^*\),

\[
\#total_N(x) = \#total_M(x), \quad \text{and} \quad \#acc_N(x) = \#rej_M(x).
\]

3.1 \(\#P\)-hard Closure Properties

We prove that subtraction, division, weak and strong plurality, and span are \(\#P\)-hard \(P\)-closure properties. We do so by establishing a structural condition—\(PP = UP\)—that completely characterizes both the question of whether \(\#P\) is \(P\)-closed and the question of whether \(\#P\) is \(\#P\)-closed.

**Theorem 3.2** The following statements are equivalent.

1. \(\#P\) is \(P\)-closed. That is, \(\#P\) has every \(P\)-closure property.
2. \(\#P\) is \(\#P\)-closed. That is, \(\#P\) has every \(\#P\)-closure property.
3. \(PP = UP\).
4. \(PH = CH = C_\#P = PP = \oplus P = UP\).
5. \(\#P\) is closed under division by polynomial-time computable functions.
6. \(\#P\) is closed under division by \(\#P\) functions.
7. \#P is closed under subtraction by polynomial-time computable functions.

8. \#P is closed under subtraction by \#P functions.

9. For some \( k \geq 5 \), \#P is closed under weak plurality of \( k \) functions.

10. For every \( k \geq 2 \), \#P is closed under weak plurality of \( k \) functions.

11. For some \( k \geq 3 \), \#P is closed under strong plurality of \( k \) functions.

12. For every \( k \geq 2 \), \#P is closed under strong plurality of \( k \) functions.

13. For some \( k \geq 2 \), \#P is closed under span of \( k \) functions.

14. For every \( k \geq 2 \), \#P is closed under span of \( k \) functions.

Proof First notice the following:

- (3), (4), and \( C=\text{P} = \text{UP} \) are equivalent. This is seen as follows: Obviously, (4) implies (3). Since \( \text{UP} \subseteq C=\text{P} \subseteq \text{PP} \), (3) implies \( C=\text{P} = \text{UP} \). Suppose \( C=\text{P} = \text{UP} \). This gives \( C=\text{PP} = C=\text{C}=\text{P} = C=\text{UP} = \text{PP} \subseteq C=\text{C}=\text{P} = C=\text{UP} = \text{UP} \). Thus, \( \text{CH} = \text{UP} \). Since \( \text{UP} \subseteq (\#\text{P} \cap \text{PH}) \subseteq \text{CH} \), (4) holds. In the following discussions, we use \( C=\text{P} = \text{UP} \) instead of (3).

- (2) implies (1) because \( \text{PF} \subseteq \#\text{P} \).

- (1) implies (5) through (14) because division, subtraction, \( \text{plu}_k, \text{plu}^*_k \), and \( \text{span}_k \) are polynomial-time computable.

- \( C=\text{P} = \text{UP} \) implies (2). This is seen as follows: Suppose \( C=\text{P} = \text{UP} \). Let \( k \geq 2 \) and \( f_1, \ldots, f_k \) and \( g \) be functions in \#P and define \( h(x) = g((f_1(x), \ldots, f_k(x))) \). For every \( i, 1 \leq i \leq k \), define \( A_i = \{(x, n) \mid f_i(x) = n\} \), and define \( A = \{(x, n_1, \ldots, n_k) \mid i, 1 \leq i \leq k, \langle x, n_i \rangle \in A_i\} \). For every \( i, 1 \leq i \leq k \), \( A_i \) is in \( C=\text{P} \), and since \( C=\text{P} \) is closed under positive truth-table reductions, \( A \in C=\text{P} \), and thus, from our supposition, \( A \in \text{UP} \). Let \( D \) be a machine such that \( g = \#\text{acc}_D \) and \( M \) be a machine witnessing that \( A \in \text{UP} \) and define \( N \) to be the machine that, on input \( x \in \Sigma^* \), (i) guesses \( n_1, \ldots, n_k \), (ii) simulates \( M \) on \( z = (x, n_1, \ldots, n_k) \), and (iii) (a) if \( M \) accepts, then simulates \( D \) on \( (n_1, \ldots, n_k) \), and (b) if \( M \) rejects, then rejects \( x \). \( N \) is polynomial-time bounded and \( h = \#\text{acc}_N \). Hence, \#P is \#P-closed.

It suffices to show that each of (5) through (14) implies \( C=\text{P} \subseteq \text{UP} \). But, since (6), (8), (10), (12), and (14) implies (5), (7), (9), (11), and (13), respectively, we have only to show
that each of (5), (7), (9), (11), and (13) implies $C = P = \text{UP}$. In the following, $L_0$ denotes an arbitrary set in $C = P$, $M_0$ and $p_0$ denote a machine and a polynomial that satisfy the properties in Proposition 2.6; that is, for every $x \in \Sigma^*$,

$$x \in L_0 \iff \text{acc}_{M_0}(x) = 2^{p_0(|x|)}; \quad \text{and} \quad x \notin L_0 \iff \text{acc}_{M_0}(x) < 2^{p_0(|x|)}.$$ 

Moreover, let $f_0$ denote $\text{acc}_{M_0}$ and $g_0(x) = 2^{p_0(|x|)}$.

Suppose (5) holds. Define $h(x) = |f_0(x)/g_0(x)|$. By hypothesis, $h \in \#P$, and for every $x \in \Sigma^*$, $h(x) = 1$ if $x \in L_0$ and 0 otherwise. Thus $L_0 \in \text{UP}$.

Suppose (7) holds. Define $h(x) = f_0(x) \Theta (g_0(x) - 1)$. By hypothesis, $h \in \#P$, and for every $x \in \Sigma^*$, $h(x) = 1$ if $x \in L_0$ and 0 otherwise. Thus $L_0 \in \text{UP}$.

Suppose (9) holds for some $k \geq 5$. Define $s(x) = 2f_0(x)$, $t(x) = f_0(x) + g_0(x)$, $u(x) = 2g_0(x)$, and $v(x) = 3g_0(x)$. Clearly, $s, t, u, v$ are in $\#P$, and for every $x \in \Sigma^*$,

- if $x \in L_0$, then $s(x) = t(x) = u(x) = 2g_0(x) < v(x)$, and
- if $x \notin L_0$, then $0 < s(x) < t(x) < u(x) = 2g_0(x) < v(x)$.

Let $d = \lfloor (k - 1)/2 \rfloor$ and define functions $f_1, \ldots, f_k$ in the following way: (i) $f_1 = \cdots = f_d = C_0$, (ii) $f_{d+1} = \cdots = f_{2d-1} = u$, (iii) $f_{2d} = s$, $f_{2d+1} = t$, and (iv) if $k$ is even (that is, if $k = 2d + 2$), then $f_{2d+2} = v$. Clearly, $f_1, \ldots, f_k$ are in $\#P$. It is not hard to see that for every $x \in \Sigma^*$,

- if $x \in L_0$, then $\text{plu}_k(f_1(x), \ldots, f_k(x)) = \{2g_0(x)\}$, and
- if $x \notin L_0$, then $\text{plu}_k(f_1(x), \ldots, f_k(x)) = \{0\}$.

Thus, from our supposition, there exists a function $h \in \#P$ such that for every $x \in \Sigma^*$, $h(x) = 2g_0(x)$ if $x \in L_0$ and 0 otherwise.

Moreover, define $a(x) = 2h(x)$, $b(x) = 4h(x)$, and $c(x) = 5h(x)$. Clearly, $a, b, c$ are in $\#P$, and for every $x \in \Sigma^*$,

- if $x \in L_0$, then $1 < a(x) < b(x) < c(x) = 5h(x)$, and
- if $x \notin L_0$, then $0 = a(x) = b(x) < c(x) = 5h(x)$.

Define $h_1, \ldots, h_k$ in the following way: (i) $h_1 = \cdots = h_d = C_1$, (ii) $h_{d+1} = \cdots = h_{2d-1} = C_0$, (iii) $f_{2d} = s$, $f_{2d+1} = t$, and (iv) if $k$ is even (that is, if $k = 2d + 2$), then $f_{2d+2} = u$. It is not hard to see that for every $x \in \Sigma^*$,
• if \( x \in L_0 \), then \( \text{plu}_k(h_1(x), \cdots, h_k(x)) = \{1\} \), and

• if \( x \notin L_0 \), then \( \text{plu}_k(h_1(x), \cdots, h_k(x)) = \{0\} \).

Thus, from our supposition, there is a function \( r \in \#P \) such that for every \( x \in \Sigma^* \), \( r(x) = 1 \) if \( x \in L_0 \) and 0 otherwise. Hence, \( L_0 \in \text{UP} \); consequently, \( C_\#P \subseteq \text{UP} \).

Suppose that (11) holds for some \( k \geq 3 \). The proof is quite similar to that for weak plurality. Indeed, for the case \( k \geq 5 \), the same functions \( f_1, \cdots, f_k \) and \( h_1, \cdots, h_k \) work properly for this proof also, since \( \text{plu}_k(f_1(x), \cdots, f_k(x)) \) and \( \text{plu}_k(h_1(x), \cdots, h_k(x)) \) always contain exactly one value.

We only have to consider the case \( k \in \{3, 4\} \). Define \( f_1(x) = 0, f_2(x) = 2f_0(x), f_3(x) = f_0(x) + g_0(x), \) and \( f_4(x) = 3g_0(x) \). These functions are in \( \#P \), and for every \( x \in \Sigma^* \),

• if \( x \in L_0 \), then \( 0 = f_1(x) < f_2(x) = f_3(x) = 2g_0(x) < f_4(x) \), and

• if \( x \notin L_0 \), then \( 0 = f_1(x) = f_2(x) < f_3(x) < f_4(x) \).

Define \( h = \text{plu}_k^*(f_1, \cdots, f_k) \). From our supposition, \( h \in \#P \), and for every \( x \in \Sigma^* \), \( h(x) = 2g_0(x) \) if \( x \in L_0 \) and 0 otherwise.

Moreover, define \( h_1(x) = 1, h_2(x) = 2h(x), h_3(x) = 3h(x), \) and \( h_4(x) = 4h(x) \). Clearly, \( h_1, h_2, h_3, \) and \( h_4 \) are in \( \#P \), and for every \( x \in \Sigma^* \),

• if \( x \in L_0 \), then \( 1 = f_1(x) < f_2(x) < f_3(x) < f_4(x) \), and

• if \( x \notin L \), then \( 0 = f_2(x) = f_3(x) = f_4(x) < f_1(x) = 1 \).

Define \( r(x) = \text{plu}_k^*(f_1(x), \cdots, f_k(x)) \). From our supposition, \( r \in \#P \), and for every \( x \in \Sigma^* \), \( r(x) = 1 \) if \( x \in L_0 \) and 0 otherwise. Hence, \( L_0 \in \text{UP} \); consequently, \( C_\#P \subseteq \text{UP} \).

Suppose that (13) holds for some \( k \geq 2 \). Define \( f_1 = \cdots = f_{k-1} = f_0 \) and \( f_k = g_0 \) and define \( h(x) = \text{span}_k(f_1(x), \cdots, f_k(x)) \). From our supposition, \( h \in \#P \), and for every \( x \in \Sigma^* \), \( h(x) = 1 \) if \( x \in L_0 \) and 2 otherwise. Since \( h \in \#P \), for some machine \( D \), it holds that \( h = \#\text{acc}_D \). Let \( p \) be the polynomial such that for every \( x \in \Sigma^* \), each computation path of \( D \) on \( x \) is encoded into a string of length \( p(|x|) \). Define \( N \) to be the machine that, on input \( x \in \Sigma^* \), guesses two paths \( y, z, |y| = |z| = p(|x|) \) of \( D \) on \( x \) with \( y < z \), and accepts \( x \) if and only if \( D \) accepts on both of the paths. It is not hard to see that for every \( x \in \Sigma^* \), if \( x \in L_0 \) then \( \#\text{acc}_N(x) = 0 \) and if \( x \notin L_0 \) then \( \#\text{acc}_N(x) = 1 \). Thus, \( \overline{L_0} \in \text{UP} \). This implies \( C_\#P \subseteq \text{coUP} \). Since \( \text{UP} \subseteq C_\#P \), we have \( C_\#P = \text{UP} \). Q.E.D.

PHCF is the smallest class of functions satisfying the following properties:
• $\#P \subseteq \text{PHCF}$.

• For a polynomial-time nondeterministic Turing machine $M$ and for a function $f \in \text{PHCF}$, define $g(x)$ = the number of accepting computation paths of $M$ on $x$ with function-oracle $f$. Then, $g \in \text{PHCF}$.

PHCF was defined and studied by Wagner [Wag86], who has pointed out to the authors (personal communication, 1991) that each condition in Theorem 3.2 holds if and only if PHCF collapses to $\#P$—a function-based analog of the language collapse $\text{CH} = \text{NP}$.

Corollary 3.3 PHCF = $\#P$ if and only if $\#P$ is $\#P$-closed.

As a final comment, we note that it follows immediately from the previous theorem that there are relativized worlds in which $\#P$ lacks all $\#P$-hard closure properties and, more interestingly, there are relativized worlds (for example, any world in which $P \neq \text{UP} = \text{PSPACE}$, such as in the construction of Rackoff [Rac82]) in which $\#P$ is non-trivial (i.e., $\#P \neq \text{PF}$), yet has all $P$-closure properties. Thus no relativizable proof technique can hope to prove such statements as:

• $\#P$ is not $P$-closed.

• $\#P$ is $P$-closed.

• If $\#P$ is $P$-closed then $P = \text{NP}$.

• If $\#P$ is $P$-closed then $\#P = \text{PF}$.

3.2 Intermediate Closure Properties for $\#P$

The previous subsection established the existence of $\#P$-hard closure properties, including many natural closure properties. This subsection studies closure properties that seem to be neither possessed by $\#P$ nor $\#P$-hard.

First, we consider minimum, maximum, and median.

Theorem 3.4 The following statements are equivalent.

1. $\#P$ is closed under both minimum and maximum.

2. For some odd natural number $k \geq 3$, $\#P$ is closed under median of $k$ functions.

3. For every odd natural number $k \geq 3$, $\#P$ is closed under median of $k$ functions.
Proof [(1) \(\rightarrow\) (2), (1) \(\rightarrow\) (3)] Suppose that \(\#P\) is closed under both minimum and maximum. Let \(k = 2m + 1 \geq 3\) be an odd integer and \(f_1, \cdots, f_k\) be given \(k\) functions in \(\#P\). Define \(g_1, \cdots, g_{2m+1}\) to be an enumeration of all functions that compute \(\max_{m+1}\) of \(m+1\) functions chosen from \(f_1, \cdots, f_{2m+1}\) and \(h(x) = \min\{g_1(x), \cdots, g_{2m+1}(x)\}\). From our supposition, \(h \in \#P\), and it is easy to see that, for every \(x \in \Sigma^*\), \(h(x) = \text{median}_{2m+1}(f_1(x), \cdots, f_{2m+1}(x))\). Thus, for every odd \(k \geq 3\), \(\#P\) is closed under median of \(k\) functions. Thus (1) implies both (2) and (3).

[(2) \(\rightarrow\) (1), (3) \(\rightarrow\) (1)] Suppose that \(\#P\) is closed under median of \(k\) functions for some odd \(k \geq 3\). For simplicity, let \(k = 2m + 1\). Let \(s\) and \(t\) be two functions in \(\#P\). Clearly, there is a function \(u \in \#P\) such that for every \(x \in \Sigma^*\), it holds that \(u(x) > \max\{s(x), t(x)\}\). Define \(f_1, \cdots, f_k\) in the following way:

\[
f_1 = \cdots = f_{m-1} = C_0, \quad f_m = s, \quad f_{m+1} = t, \quad \text{and} \quad f_{m+2} = \cdots f_{2m+1} = u.
\]

For every \(x \in \Sigma^*\),

\[
0 = f_1(x) = \cdots = f_{m-1}(x) \leq \min\{s(x), t(x)\} \leq \max\{s(x), t(x)\} < f_{m+2}(x) = \cdots = f_{2m+1}(x).
\]

This implies that for every \(x \in \Sigma^*\), \(\text{median}_k(f_1(x), \cdots, f_k(x)) = \max\{s(x), t(x)\}\). Thus, \(\#P\) is closed under maximum.

On the other hand, in order to prove that \(\#P\) is closed under minimum, instead of defining \(f_{2m+1}\) to be \(u\), we only have to define \(f_{2m+1} = C_0\). By using a similar argument, for every \(x \in \Sigma^*\), \(\text{median}_k(f_1(x), \cdots, f_k(x)) = \min\{s(x), t(x)\}\). Thus, \(\#P\) is closed under minimum.

The proof for [(3) \(\rightarrow\) (1)] is easy because (3) implies (2).

Q.E.D.

Next, we explore the structural collapses that would occur were \(\#P\) to have the above closure properties. We define explicitly a natural class, exact (counting) polynomial time, that has appeared implicitly in much of the previous work on counting [Tod89, KSTT89].

**Definition 3.5** \(XP\),\(^5\) exact polynomial time, is the class of sets \(L\) for which there exist a function \(f \in \#P\) and a polynomial \(p\) such that for every \(x \in \Sigma^*\),

\[
x \in L \iff f(x) = 2^{p(|x|)} + 1, \quad \text{and} \quad x \notin L \iff f(x) = 2^{p(|x|)}.
\]

\(^5\)As pointed out to the authors by Wagner (personal communication, 1991), \(XP\) and \(SPP\), respectively defined (independently) in [OH90] and [FFK90], are identical classes.
Proposition 3.6  
1. $XP = coXP$.
2. $UP \subseteq XP \subseteq (C= P \cap coC= P \cap \oplus P)$.
3. $C= \cdot XP \subseteq C= P$ and $C \cdot XP \subseteq PP$.

Proof (1) and (2) are immediate from the definition.

Let $L$ be a set in $C= \cdot XP$. There exist a polynomial $p$ and a set $A \in XP$ such that for every $x \in \Sigma^*$,

$$x \in L \iff \# \{ y \mid |y| = p(|x|) \text{ and } (x, y) \in A \} = 2^{p(|x|)-1}.$$ 

Since $A \in XP$, there exist a machine $M$ and a polynomial $q$ such that for every $u \in \Sigma^*$, if $u \in L$ then $f(u) = 2^{q(|u|)} + 1$, and if $u \notin L$ then $f(u) = 2^{q(|u|)}$. Define $N$ to be the machine that, on input $x \in \Sigma^*$, guesses $y, |y| = p(|x|)$, simulates $M$ on $(x, y)$, and accepts $x$ if and only if $M$ accepts $(x, y)$. It is not hard to see that for every $x \in \Sigma^*$,

$$x \in L \iff \# \{ y \mid |y| = p(|x|) \text{ and } (x, y) \in A \} = 2^{p(|x|)+q(m(x))} + 2^{p(|x|)-1},$$

where $m(|x|)$ denotes $|\langle x, y \rangle|$ for $y \in \Sigma^{=p(|x|)}$.\(^6\) Thus, $L \in C= P$.

The proof for $C \cdot XP \subseteq PP$ is quite similar and thus omitted. Q.E.D.

Theorem 3.7  
1. If $\#P$ is closed under either maximum or minimum, then $C= P \subseteq XP$.
2. If $\#P$ is closed under minimum, then $NP = UP$.

Proof Let $L$ be a set in $C= P$. There exist a machine $M$ and a polynomial $p$ such that for every $x \in \Sigma^*$, (i) $\#\text{total}_M(x) = 2^{p(|x|)+1}$, (ii) if $x \in L$, then $\#\text{acc}_M(x) = 2^{p(|x|)}$, and (iii) if $x \notin L$, then $\#\text{acc}_M(x) < 2^{p(|x|)}$. Define $f_1(x) = \#\text{acc}_M(x) + 1$, $f_2(x) = \#\text{rej}_M(x)$, $g_1(x) = 2^{p(|x|)}$, and $g_2(x) = 2^{p(|x|)} + 1$. Clearly, these functions are in $\#P$. Furthermore, define $h_1(x) = \min \{ f_1(x), g_1(x) \}$ and $h_2(x) = \min \{ f_2(x), g_2(x) \}$. Now suppose that $\#P$ is closed under either maximum or minimum. Clearly, either $h_1$ or $h_2$ is in $\#P$. For every $x \in \Sigma^*$,

- if $x \in L$, then $h_1(x) = h_2(x) = 2^{p(|x|)} + 1$, and
- if $x \notin L$, then $h_1(x) = h_2(x) = 2^{p(|x|)}$.

Note that $|\langle x, y \rangle|$ depends only upon $|x|$ and $|y|$.
Thus, $L \in \text{XP}$. Hence, $C=\text{P} \subseteq \text{XP}$.

On the other hand, suppose that $\#\text{P}$ is closed under minimum. Let $L$ be a set in $\text{NP}$ and let $f$ be a function in $\#\text{P}$ witnessing $L \in \text{NP}$. By computing a minimum with $C_1$, the constant function "one," we obtain a new function in $\#\text{P}$ that witnesses $L \in \text{UP}$. This proves the theorem. \[\text{Q.E.D.}\]

From the above theorem, we obtain the following corollary.

**Corollary 3.8** $\text{CH} = \text{PP} = C=\text{P} = \oplus\text{P} = \text{XP}$ and $\text{NP} = \text{UP}$ if one of the following conditions hold:

1. $\#\text{P}$ is closed under both minimum and maximum.

2. For some odd number $k \geq 3$, $\#\text{P}$ is closed under median of $k$ functions.

3. For every odd number $k \geq 3$, $\#\text{P}$ is closed under median of $k$ functions.

**Proof** From Theorem 3.4, the above three conditions are equivalent. By Theorem 3.7, (1) above implies (i) $\text{NP} = \text{UP}$ and (ii) $C=\text{P} \subseteq \text{XP}$. By combining (ii) with (a) $\text{PP} \subseteq C=\text{P} \cdot C=\text{P}$, (b) $C \cdot \text{PP} = C \cdot C=\text{P}$, (c) $C=\text{P} \cdot \text{XP} = C=\text{P}$, and (d) $C \cdot \text{XP} = \text{PP}$, we have $C=\text{P} = \text{PP} = \text{CH} = \text{XP}$. Since $\text{XP} \subseteq \oplus\text{P} \subseteq \text{CH}$, we have $\oplus\text{P} = \text{XP}$. \[\text{Q.E.D.}\]

Next, we consider closure under decrement. The following theorem was pointed out to the authors by Torán (personal communication, 1991).

**Theorem 3.9** If $\#\text{P}$ is closed under decrement, then $\text{NP} \subseteq \text{XP}$.

**Proof** Suppose that $\#\text{P}$ is closed under decrement and let $L$ be a set in $\text{NP}$. There is a function $f \in \#\text{P}$ such that for every $x \in \Sigma^*$, it holds that $x \in L$ if and only if $f(x) > 0$. Define $g(x) = 2f(x)$ and $h(x) = g(x) \oplus 1$. $g$ is in $\#\text{P}$, and since we are assuming that $\#\text{P}$ is closed under decrement, $h$ is in $\#\text{P}$, too. Let $M$ and $N$ be polynomial-time machines such that $g = \#\text{acc}_M$ and $h = \#\text{acc}_N$. Let $p$ be the polynomial such that $\#\text{total}_N(x) = 2^{p(|x|)}$ for every $x \in \Sigma^*$. Define $u = \#\text{acc}_M + \#\text{rej}_N$. Obviously, $u \in \#\text{P}$ and for every $x \in \Sigma^*$, (i) if $x \in L$, then $u(x) = 2^{p(|x|)} + 1$, and (ii) if $x \notin L$, then $u(x) = 2^{p(|x|)}$. Thus, $L \in \text{XP}$. \[\text{Q.E.D.}\]

From Theorem 3.9, $\text{XP} \subseteq \oplus\text{P}$, and Torán's [Tor88b] relativized world in which $\text{NP}$ is not contained in $\oplus\text{P}$, we immediately obtain the following result.

**Corollary 3.10** There is a relativized world in which $\#\text{P}$ is not closed under decrement.
In fact, we can say a bit more. Let us say that a function $f : \Sigma^* \rightarrow \mathbb{N}$ is bounded by $k$ if, for every $x \in \Sigma^*$, it holds that $f(x) \leq k$. On one hand, it holds (absolutely, and also in every relativized world) that if $f$ is a $\#P$ function bounded by 2, then $f(x) \oplus 1$ is a $\#P$ function. On the other hand, via a combinatorial diagonalization argument extending those used by Cai et al. [CGH'89] to study the counting hierarchy, one can show that there are relativized worlds in which some $\#P$ function $f$ is bounded by 3, yet $f(x) \oplus 1$ is not in $\#P$.

Next, we consider conditions sufficient to ensure the closure of $\#P$ under decrement.

**Theorem 3.11** If $\coNP \subseteq \UP$, then $\#P$ is closed under decrement.

**Proof** Suppose that $\coNP \subseteq \UP$ and let $M$ be any nondeterministic polynomial-time Turing machine. Since $M$ runs in polynomial time, there exists a polynomial $p$ such that for every $x$, each computation path of $M$ on $x$ is encoded into a string of length $p(|x|)$. Let $A$ be the set $\{(x, y) \mid |y| = p(|x|) \text{ and } y \text{ is lexicographically the smallest accepting path of } M \text{ on } x\}$. Clearly, $A \in \coNP$, and thus, from our supposition, $A \in \UP$; that is, there exists a machine $D$ such that for every $(x, y)$, $|y| = p(|x|)$, it holds that $\#\text{acc}_D((x, y)) = 1$ if $(x, y) \in A$ and 0 otherwise.

Now define $N$ to be the machine that, on input $x \in \Sigma^*$, (i) guesses $y, z, |y| = |x| = p(|x|)$, with $y \neq z$, (ii) simultaneously simulates (a) $D$ on $(x, y)$ and (b) $M$ on $x$ along path $z$, and (iii) accepts $x$ if and only if both $D$ and $M$ accept. $N$ runs in time polynomial in $|x|$, and for every $x \in \Sigma^*$, it holds that $\#\text{acc}_N(x) = \#\text{acc}_M(x) - 1$ if $\#\text{acc}_M(x) > 0$, and 0 otherwise. Thus, $N$ witnesses that $\#P$ is closed under decrement. Q.E.D.

Next, we consider closure under division by 2.

**Theorem 3.12** If $\#P$ is closed under division by 2, then $\oplus P = \XP$.

**Proof** Suppose that $\#P$ is closed under division by 2 and let $L$ be a set in $\oplus P$. There is a function $f \in \#P$ such that for every $x \in \Sigma^*$, it holds that $x \in L \iff f(x)$ is odd. Since $\#P$ is closed under division by 2, there is a function $g \in \#P$ such that for every $x \in \Sigma^*$, $g(x) = 2\lfloor f(x)/2 \rfloor$. Clearly, for every $x \in \Sigma^*$, it holds that:

$$
x \in L \iff f(x) - g(x) = 1, \quad \text{and}
$$

$$
x \notin L \iff f(x) - g(x) = 0.
$$

Let $M$ be a machine such that $g = \#\text{acc}_M$. Without loss of generality, we may assume that for some polynomial $p$, $\# \text{total}_M(x) = 2^p(|x|)$ for every $x \in \Sigma^*$. Define $h(x) = f(x) + \ldots$
The complexity-theoretic theory of lowness was developed by Schöning to study the
types of information that increase the computational power of complexity classes ([Sch83],
see also [KSTT89,AH]). We'll say that a class K is $C = P$-low if $C = P^K = C = P$, and that a
class $K$ is $PP$-low if $PP^K = PP$. From the above theorem, we obtain the following corollary.

**Corollary 3.13** If $\#P$ is closed under division by 2, then $\oplus P$ is $C = P$-low and $PP$-low.

**Proof** Since $C = P^{\oplus P} \subseteq C = P^{P^{\oplus P}} \subseteq C = P^{\oplus P}$ and $PP^{\oplus P} \subseteq C = P^{\oplus P} \subseteq C = P$ implies both $C = P^{\oplus P} \subseteq C = P$ and $PP^{\oplus P} \subseteq PP$.

Toda [Tod89] has shown that $PH \subseteq PP$; it is not known whether $PH \subseteq PP$
(see [BHW91,Bei89]). However, if we assume that $\#P$ is closed under division by two,
this inclusion indeed holds.

**Corollary 3.14** If $\#P$ is closed under division by 2, then $PH \subseteq PP$.

**Proof** Assume that $\#P$ is closed under division by 2. Notice that $PH \subseteq \oplus P^{PH} \subseteq\langle BP \cdot \oplus P \right \rangle^{PP}$. Combining this with Corollary 3.13, it follows that $PH \subseteq PP$.

4 A Complexity Theory for Closure Properties of SpanP

In this section, we apply the techniques of Section 3 to SpanP, the class of functions
measuring the number of distinct outputs of NP machines.

**Definition 4.1** ([KST89], see also [K89,Sch90]) For a polynomial-time nondeterministic
Turing transducer $M$, we say that $y$ is an output string of $M$ if, on some accepting computation
path, $M$ halts with $y$ on its work tape. $out_N(x)$ denotes the set of all output strings of
$M$ on $x$. For a polynomial-time nondeterministic Turing transducer $M$, $span_M$ is the mapping
from $\Sigma^*$ to $N$ such that for every $x \in \Sigma^*$, $span_M(x) = \#out_N(x)$. SpanP = \{span_M | M
is a polynomial-time nondeterministic Turing transducer\}.

We define two complexity classes that we will be concerned with in this section, and
immediately prove that these classes characterize levels of the counting polynomial hierarchy
(see [Ogi91] for a study of characterizations of certain other levels of the counting polynomial
hierarchy).
Definition 4.2  
1. CSpanP is the class of sets $L$ for which there exist a function $f \in \text{SpanP}$ and a function $g \in \text{PF}$ such that for every $x \in \Sigma^*$, it holds that:

$$x \in L \iff f(x) \geq g(x).$$

2. $C=\text{SpanP}$ is the class of sets $L$ for which there exist a function $f \in \text{SpanP}$ and a function $g \in \text{PF}$ such that for every $x \in \Sigma^*$, it holds that:

$$x \in L \iff f(x) = g(x).$$

Proposition 4.3  
1. [KST89] SpanP $\supseteq \#P$.

2. Every closure property $\#P$ is currently known to have (see [Reg82, CGH+89, BGH90]) is also possessed by SpanP. For example, SpanP is closed under addition, multiplication and binomial coefficients.

3. A function $f$ is in SpanP if and only if there exist a polynomial $p$ and a set $A \in \text{NP}$ such that for every $x \in \Sigma^*$, it holds that:

$$f(x) = \#\{y \mid |y| = p(|x|) \text{ and } (x,y) \in A\}.$$

4. CSpanP = $C \cdot \text{NP}$ and $C=\text{SpanP} = C= \cdot \text{NP}$.

5. CSpanP = NP if and only if $C=\text{SpanP} = \text{NP}$.

**Proof**  
(1) For a given polynomial-time nondeterministic Turing acceptor $M$, consider a transducer $N$ that guesses a computation path $y$ of $M$ (named by its nondeterministic guess bits) and outputs $y$ if $M$ accepts along path $y$. Clearly, $\text{span}_N = \#\text{acc}_M$.

(2) All known $\#P$ closure properties are proven by direct simulations of relatively simply form. The simulations can be modified from, in the $\#P$ case, guessing paths and checking properties of the guessed paths (such as acceptance behavior) to, in the SpanP case, guessing paths and outputs and checking relations between the guessed objects.

To show how this works, we give three examples: addition, multiplication and binomial coefficients. Let $\text{span}_M$ and $\text{span}_N$ be two given SpanP functions.

Define $D$ to be the machine that, on input $x \in \Sigma^*$, guesses $b \in \{0,1\}$ and $y$, nondeterministically checks that $y \in \text{out}_M(x)$ if $b = 0$ and $y \in \text{out}_N(x)$ if $b = 1$, and outputs $by$ along paths on which the check succeeded. For every $x \in \Sigma^*$, $\text{out}_D(x) = \{0y \mid y \in \text{out}_M(x)\} \cup \{1y \mid y \in \text{out}_N(x)\}$. Thus, for every $x \in \Sigma^*$, $\text{span}_D(x) = \text{span}_M(x) + \text{span}_N(x)$.  

20
Define $E$ to be the machine that, on input $x \in \Sigma^*$, (i) guesses $y$ and $z$, (ii) nondeterministically checks that $y \in \text{out}_M(x)$ and $z \in \text{out}_N(x)$ and (iii) outputs $(y, z)$ along paths on which the check succeeded. For every $x \in \Sigma^*$, $\text{out}_E(x) = \{(y, z) \mid y \in \text{out}_M(x) \text{ and } z \in \text{out}_N(x)\}$. Thus, for every $x \in \Sigma^*$, $\text{span}_E(x) = \text{span}_M(x) \cdot \text{span}_N(x)$.

Finally, let $k \in \mathbb{N}$ and define $T$ to be a machine that, on input $x \in \Sigma^*$, guesses $k$ distinct strings, $y_1, \ldots, y_k$ in lexicographically increasing order, nondeterministically checks that $y_1, \ldots, y_k \in \text{out}_M(x)$ and, if so, outputs $(y_1, \ldots, y_k)_k$ along paths on which the check succeeded. For every $x \in \Sigma^*$, $\text{span}_T(x) = \binom{\text{span}_M(x)}{k}$.

Hence, $\text{SpanP}$ is closed under addition, multiplication, and binomial coefficients.

(3) Let $f = \text{span}_M$ for some polynomial-time nondeterministic Turing machine $M$. Without loss of generality, we may assume that for every $x \in \Sigma^*$, each output of $M$ on $x$ has length $p(|x|)$, where $p$ is a fixed polynomial. Define $A = \{(x, y) \mid M$ on $x$ outputs $y\}$. Clearly, $A \in \text{NP}$, and for every $x \in \Sigma^*$, it holds that $\text{span}_M = \#\{y \mid |y| = p(|x|) \text{ and } (x, y) \in A\}$.

Conversely, given a set $A \in \text{NP}$ and a polynomial $p$, consider a polynomial-time nondeterministic machine $N$ that, on input $x \in \Sigma^*$, (i) guesses $y$ of length $p(|x|)$, (ii) nondeterministically tests whether $(x, y) \in A$ or not, (iii) outputs $y$ along paths on which the check succeeded. For every $x \in \Sigma^*$, $\text{span}_N = \#\{y \mid |y| = p(|x|) \text{ and } (x, y) \in A\}$.

(4) From (3) of this proposition, $\text{CSpanP} = \text{C} \cdot \text{NP}$ and $\text{C} = \text{SpanP} = \text{C} \cdot \text{NP}$.

(5) First note that (i) $\text{C} = \text{NP} \subseteq \text{C} \cdot \text{NP}$ and (ii) $\text{C} \cdot \text{NP} \subseteq \exists \cdot \text{C} = \text{NP}$. Suppose $\text{CSpanP} = \text{NP}$. From (i) we have $\text{C} \cdot \text{NP} \subseteq \text{NP}$. Also, trivially, $\text{NP} \subseteq \text{C} \cdot \text{NP}$ holds. So $\text{C} = \text{SpanP} = \text{NP}$ follows from our supposition. On the other hand, suppose $\text{C} = \text{SpanP} = \text{NP}$. Thus, from (ii), we have $\text{C} \cdot \text{NP} \subseteq \exists \cdot \text{NP} = \text{NP}$. Also, trivially, $\text{NP} \subseteq \text{C} \cdot \text{NP}$ holds. So $\text{CSpanP} = \text{NP}$ follows from our supposition. Q.E.D.

Now we consider the question of $\text{SpanP}$'s closure properties. The theorem below should be compared and contrasted with this paper's results on $\#P$: the possibilities that $\#P$ and $\text{SpanP}$ are $\text{P}$-closed are characterized, respectively, by the complexity class collapses $\text{PP} = \text{UP}$ (equivalently, $\text{C} = \text{P} = \text{UP}$) and $\text{C} \cdot \text{NP} = \text{NP}$.

**Theorem 4.4** The following statements are equivalent.

1. \text{SpanP} is \text{P}-closed. That is, \text{SpanP} has every \text{P}-closure property.

2. \text{SpanP} is \text{SpanP}-closed. That is, \text{SpanP} has every \text{SpanP}-closure property.

3. \text{CSpanP} = \text{NP} (equivalently, $\text{C} \cdot \text{NP} = \text{NP}$).

21
4. $PH = CH = C_{=}P = PP = NP$.

5. SpanP is closed under division by polynomial-time computable functions.

6. SpanP is closed under division by SpanP functions.

7. SpanP is closed under subtraction by polynomial-time computable functions.

8. SpanP is closed under subtraction by SpanP functions.

9. For some $k \geq 5$, SpanP is closed under weak plurality of $k$ functions.

10. For every $k \geq 2$, SpanP is closed under weak plurality of $k$ functions.

11. For some $k \geq 3$, SpanP is closed under strong plurality of $k$ functions.

12. For every $k \geq 2$, SpanP is closed under strong plurality of $k$ functions.

13. For some $k \geq 2$, SpanP is closed under span of $k$ functions.

14. For every $k \geq 2$, SpanP is closed under span of $k$ functions.

Proof The proof is quite similar to that of Theorem 3.2. First notice the following:

- (2) implies (1) because SpanP $\subseteq$ PF.

- (1) implies (5) through (14) because division, subtraction, weak plurality, strong plurality, and span are computable within polynomial time.

- Trivially, (4) implies (3).

- (3) implies (4) because under the assumption that $C \cdot NP = NP$, we have (a) $\Pi_P^P \subseteq \forall \cdot NP \subseteq C \cdot NP \subseteq NP$, (b) $C \cdot C \cdot P \subseteq C \cdot C \cdot NP \subseteq C \cdot NP \subseteq NP$, and (c) $C_{=} \cdot NP = NP$.

- (3) implies (2). This can be seen as follows: (3) is equivalent to $C_{=}\text{SpanP} = NP$. From $C_{=}\text{SpanP} = NP$, one can easily show that for any polynomial-time nondeterministic Turing machine $M$, the set $L_M = \{(x,n) | \text{span}_M(x) = n\}$ is in NP. Given SpanP functions $f_1, \ldots, f_k$ (computed by $M_1, \ldots, M_k$) and $g$, in order to compute $g((f_1(x), \ldots, f_k(x)))$, we have only to (i) guess $n_1, \ldots, n_k$, (ii) guess certificates that $(x, n_i) \in L_M$ for each $1 \leq i \leq k$, and (iii) generate $g((n_1, \ldots, n_k))$ distinct outputs by simulating the machine witnessing $g \in \text{SpanP}$.

It suffices to show that each of (5) through (14) implies (3). But, as (6), (8), (10), (12), and (14) imply (5), (7), (9), (11), and (13), respectively, and $C_{=}\text{SpanP} = NP$ if and only
if $C = \text{SpanP} = \text{NP}$, we have only to show that each of $(5)$, $(7)$, $(9)$, $(11)$, and $(13)$ implies $C = \text{SpanP} = \text{NP}$.

In the following discussions, let $L_0$ be a set in $C = \text{SpanP}$ and $f_0 \in \text{SpanP}$ and $g_0 \in \text{PF}$ be functions such that for every $x \in \Sigma^*$, $x \in L_0$ if and only if $f_0(x) = g_0(x)$. Since $\text{SpanP}$ is closed under addition, without loss of generality we may assume that there exists a polynomial $p$ such that for every $x \in \Sigma^*$, $0 < f_0(x) < 2^{p(|x|)+1}$ and $g_0(x) = 2^{p(|x|)}$.

Suppose $(5)$ holds. Define $h(x) = \lceil f_0(x)/g_0(x) \rceil$. For every $x \in \Sigma^*$, $x \in L_0$ if and only if $h(x) \geq 1$. From our supposition, $h \in \text{SpanP}$; that is, for some polynomial-time nondeterministic Turing machine $M$, it holds that $h = \text{span}_M$. So for every $x \in \Sigma^*$, $x \in L_0$ if and only if $M$ on $x$ has at least one output. This implies $L_0 \in \text{NP}$.

Suppose $(7)$ holds. Define $h(x) = f_0(x) \oplus (g_0(x) - 1)$. From our supposition, $h$ is in $\text{SpanP}$, and for every $x \in \Sigma^*$, $x \in L_0$ if and only if $h(x) \geq 1$. Thus, there is a polynomial-time machine $M$ such that for every $x \in \Sigma^*$, $x \in L_0$ if and only if $M$ on $x$ has at least one output. Hence, $L_0 \in \text{NP}$.

Suppose $(9)$ holds for some $k \geq 5$. Define $s(x) = 2f_0(x)$, $t(x) = f_0(x) + g_0(x)$, $u(x) = 2g_0(x)$ and $v(x) = 5g_0(x)$. Then for every $x \in \Sigma^*$, (i) if $x \in L_0$, then $s(x) = t(x) = u(x) < v(x)$, and (ii) if $x \notin L_0$, then $s(x), t(x), u(x)$, and $v(x)$ are all distinct and larger than zero. Now define $d = \lfloor (k-1)/2 \rfloor$ and $f_1 = \cdots = f_d = 0$, $f_{d+1} = \cdots = f_{2d-1} = u$, $f_{2d} = s$ and $f_{2d+1} = t$. Furthermore, if $k$ is even, define $f_{2d+2} = v$. Now define $h(x) = \text{plu}_k(f_1(x), \cdots, f_k(x))$. From our supposition, $h \in \text{SpanP}$. It is not hard to see that (a) if $x \in L_0$, then $\text{plu}_k(f_1(x), \cdots, f_k(x)) = \{u(x)\} = \{2g_0(x)\}$, and (b) if $x \notin L_0$, then $\text{plu}_k(f_1(x), \cdots, f_k(x)) = 0$. Thus, from our supposition, there is a polynomial-time machine $M$ such that for every $x \in \Sigma^*$, (i) $x \in L_0$ if and only if $\text{span}_M(x) = 2g_0(x)$, and (ii) $x \notin L_0$ if and only if $\text{span}_M(x) = 0$. So for every $x \in \Sigma^*$, $x \in L_0$ if and only if $M$ on $x$ has at least one output. Therefore, $L_0 \in \text{NP}$.

Suppose $(11)$ holds for some $k \geq 3$. If $k \geq 5$, the same construction as in the previous paragraph establishes $C = \text{SpanP} = \text{NP}$. So suppose that $k \in \{3,4\}$ and define $f_1(x) = 0$, $f_2(x) = 2f_0(x)$, $f_3(x) = f_0(x) + g_0(x)$, and $f_4(x) = 5g_0(x)$. It is not hard to see that for any $k \in \{3,4\}$ and for every $x \in \Sigma^*$, $x \in L_0$ if and only if $\text{plu}_k(f_1(x), \cdots, f_k(x)) > 0$. From our supposition, there is a polynomial-time machine $M$ such that for every $x \in \Sigma^*$, it holds that $x \in L_0$ if and only if $M$ on $x$ has at least one output. Thus, $L_0 \in \text{NP}$.

Finally, suppose that $(13)$ holds for some $k \geq 2$. Define $f_1(x) = \cdots = f_{k-1}(x) = f_0(x)$ and $f_k(x) = g_0(x)$. Then, clearly, for every $x \in \Sigma^*$, it holds that (i) if $x \in L_0$, then
span_k(f_1(x), \cdots, f_k(x)) = 1, and (ii) if x \notin L_0, then span_k(f_1(x), \cdots, f_k(x)) = 2. From our supposition, there exists a machine M such that for every x \in \Sigma^*, it holds that (a) if x \in L_0, then span_M(x) = 1 and (b) if x \notin L_0, then span_M(x) = 2. Define N to be a machine that, on input x \in \Sigma^*, guesses two distinct strings y and z and accepts x if and only if y, z \in out_N(x). N witnesses that L_0 \in co-NP. Since NP \subseteq C_{=\text{SpanP}}, we have C_{=\text{SpanP}} = NP.

Q.E.D.

Next we consider relationships between closure properties of \#P and those of SpanP. The following theorem may be viewed, informally, as a translation result—in particular, a downward separation result—for closure.

**Theorem 4.5** If \#P is P-closed, then SpanP is P-closed.

**Proof** Suppose that \#P is \#P-closed. From Theorem 3.2, we have CH = NP. This implies C_{SpanP} = NP. Q.E.D.

 Nonetheless, the following theorem indicates that the converse of Theorem 4.5 may be hard to establish.

**Theorem 4.6** There is a relativized world in which SpanP is P-closed yet \#P is not P-closed.

The above result follows from Lemma 4.7. The proof of the lemma allows certain diagonalization requirements to be injured; however, in such cases, the requirements are promptly (and permanently) re-satisfied.

**Lemma 4.7** There is a recursive oracle A such that \text{UP}^A \neq \text{NP}^A = \text{PSPACE}^A.

**Proof of Theorem 4.6** The theorem follows immediately from Lemma 4.7 because NP = PSPACE implies C_{SpanP} = NP and UP \neq NP implies UP \neq PP. Q.E.D.

**Proof of Lemma 4.7** Let N_1, N_2, \cdots be an enumeration of standard clocked polynomial-time nondeterministic oracle Turing machines. Without loss of generality, assume that for every i \geq 1, N_i runs in time p_i(n) = n^{log^* n}. For any set A, define:

\[ U(A) = \{ N_i \# 1^k \# z \mid N_i^A(x) \text{ accepts, having used space at most } k \}, \]

where u_1 \# \cdots \# u_m denotes a standard (easily computable and invertible) 1-1 encoding of m strings into one string such that |u_1 \# \cdots \# u_m| = 2(m - 1 + \sum_{i=1}^{m} |u_i|). Note that (i) U(A)
is \( \leq_{m}^{P} \)-complete for \( \text{PSPACE}^{A} \) and (ii) for every \( z \) of the form \( N_{i} \# x \# z, N_{i}^{A}(z) \) queries only strings of length less than \( |x| \).

Furthermore, define:

\[
L(A) = \{ 1^n \mid (\exists y)(|y| = n \text{ and } 0y \notin A) \}.
\]

Note that \( L(A) \in \text{NP}^{A} \).

We will construct a set \( A \) so that \( L(A) \not\in \text{UP}^{A} \) and \( U(A) \in \text{NP}^{A} \). \( A \) is constructed in stages. We use two sequences of sets \( \{ A_{\ell} \}_{\ell=0}^{\infty} \) and \( \{ B_{\ell} \}_{\ell=0}^{\infty} \). Initially, \( A_{0} = B_{0} = \emptyset \). At stage \( \ell \geq 1 \), \( A_{\ell} \) is extended from \( A_{\ell-1} \) either by adding at most two strings of the form \( 0y, |y| = \ell \) or by adding strings of the form \( 1\# x \# y, |y| = |x| = \ell \), and \( B_{\ell} \) is extended from \( B_{\ell-1} \) by adding at most \( p_{\ell}(\ell) \) strings, where consistently \( A_{\ell} \cap B_{\ell} = \emptyset \) is preserved. \( A \) is defined to be \( \bigcup_{\ell=0}^{\infty} A_{\ell} \). Note that (i) when entering stage \( \ell \), \( B_{\ell} \) contains fewer than \( \ell \cdot p_{\ell}(\ell) = \ell^{1 + \log^* \ell} \) elements and (ii) after finishing stage \( \ell \), for every \( x, |x| \leq \ell \), the membership of \( x \) in \( U(A) \) is never touched.

**Construction of \( A \)**

**Stage \( \ell = 2i + 1 \):** If \( i \) is of the form \( 2^{2^c} \) with \( c = j + 100 \), then we will attempt to diagonalize against machine \( N_{j}^{A_{\ell}}(1^{\ell}) \).

**Case 1 \([N_{j}^{A_{\ell}}(1^{\ell}) \text{ accepts.}]\)** Pick one accepting path (say \( \pi \)) and put into \( B_{\ell} \) all negative queries along path \( \pi \). \( N_{j}^{A} \) will not recognize \( L(A) \).

**Case 2 \([N_{j}^{A_{\ell}}(1^{\ell}) \text{ rejects.}]\)** Let \( D_{\ell} \) denote the set of all strings \( y \) of length \( \ell \) such that \( 0y \not\in B_{\ell} \). Note that \( \# D_{\ell} \geq 2^{\ell} - \ell \cdot p_{\ell}(\ell) \).

**Subcase 2a \([\text{For every } y \in D_{\ell}, N_{j}^{A_{\ell} \cup \{0y\}}(1^{\ell}) \text{ accepts.}]\)** From the Combinatorial Lemma of Cai et al. [CGH+89, page 104] there exist two strings \( y_{1}, y_{2} \in D_{\ell} \) such that \( N_{j}^{A_{\ell} \cup \{0y_{1}, 0y_{2}\}}(1^{\ell}) \) has at least two accepting paths. Pick one such pair of strings, \( y_{1} \) and \( y_{2} \), pick two accepting paths (say \( \pi_{1} \) and \( \pi_{2} \)), put \( 0y_{1} \) and \( 0y_{2} \) into \( A_{\ell} \) and put into \( B_{\ell} \) all negative queries along paths \( \pi_{1} \) and \( \pi_{2} \). Thus, \( N_{j}^{A} \) will not be categorical relative to \( A \).

**Subcase 2b \([\text{For some } y \in D_{\ell}, N_{j}^{A_{\ell} \cup \{0y\}}(1^{\ell}) \text{ rejects.}]\)** Pick one such \( y \) and put \( 0y \) into \( A_{\ell} \). Mark this \( N_{j}^{A}(1^{\ell}) \) as our "vulnerable diagonalizing situation" (for short, v.d.s.).

\(^{7}\)A machine \( M \) is **categorical** relative to \( A \) if for every input \( x \), it holds that \( M^{A}(x) \) has at most one accepting path; thus, a set is in \( \text{UP}^{A} \) if and only if it is accepted by some polynomial-time nondeterministic Turing machine that is categorical relative to \( A \) (see [Val76]).

25
Though it holds that \( N_j^{A_\ell}(1^\ell) \) rejects and \( 1^\ell \in L(A) \) at this stage, at some stage \( m > \ell \), adding some strings to \( A \) might make \( N_j^{A_\ell}(1^\ell) \) accept. Note that after finishing stage \( p_\ell(\ell) \), we do not have to pay attention to the \( \text{v.d.s.} \) because every string added to \( A \) at stage \( m \) has length greater than \( m \). Since this marking is done only when we are at stage \( 2i + 1 \) with \( i = \frac{2^j}{2^j - 2} \), at any stage at most one \( \text{v.d.s.} \) is marked.

Stage \( \ell = 2i \): At this stage, we will continue to ensure that \( U(A) \) is in \( \text{NP}^A \). For each \( x \) of length \( \ell \), in lexicographically increasing order, we will establish that:

(\(*\) \( x \in U(A) \) if and only if for some \( y, |y| = |x| \), it holds that \( 1^\#x\#y \in A \).

Case I [Either \( x \notin U(A_\ell) \) or \( x \in U(A_\ell) \) but the \( \text{v.d.s.} \), if any, does not query any string of the form \( 1^\#x\#y \) with \( |y| = |x| \).] If \( x \notin U(A) \), we do nothing because \( \(*\) \) is already satisfied. If \( x \in U(A) \), we pick one \( 1^\#x\#y \notin B_\ell \) with \( |y| = |x| \) and put it into \( A_\ell \).

Thus, the condition that the \( \text{v.d.s.} \) rejects is preserved and \( \(*\) \) is satisfied.

Case II [\( x \in U(A_\ell) \), a \( \text{v.d.s.} \) exists (say \( N_j^{A_\ell}(1^\ell) \)) and \( N_j^{A_\ell}(1^\ell) \) queries some \( 1^\#x\#y, |y| = |x| \).] Call a string \( y \) with \( |y| = |x| \) \( \text{troublesome} \) if \( N_j^{A_\ell \cup \{1^\#x\#y\}}(1^\ell) \) accepts. Let \( D_\ell \) denote the set of all strings \( y \) such that \( |y| = |x| \) and \( 1^\#x\#y \notin A_\ell \cup B_\ell \). Note that \#\( D_\ell \geq 2^\ell - \ell \cdot p_\ell(\ell) \).

Subcase IIa [Every string in \( D_\ell \) is \( \text{troublesome.} \) Again, by the Combinatorial Lemma [CGH+89, page 104], there exist two strings \( y_1, y_2 \in D_\ell \) such that \( N_j^{A_\ell \cup \{1^\#x\#y_1, 1^\#x\#y_2\}}(1^\ell) \) has at least two accepting paths. Pick such \( y_1 \) and \( y_2 \), pick two accepting paths (say \( \pi_1 \) and \( \pi_2 \)), put \( 1^\#x\#y_1 \) and \( 1^\#x\#y_2 \) into \( A_\ell \), and put into \( B_\ell \) all negative queries along \( \pi_1 \) and \( \pi_2 \). Thus, \( \(*\) \) is satisfied and we have (permanently) completed the diagonalization against the machine involved in the \( \text{v.d.s.} \), as that machine will not be categorical relative to \( A \). The \( \text{v.d.s.} \) is now removed.

Subcase IIb [There exists some \( y \in D_\ell \) that is not \( \text{troublesome.} \) Pick one such \( y \) and put \( 1^\#x\#y \) into \( A_\ell \). \( \(*\) \) has been satisfied, though the \( \text{v.d.s.} \) remains active.

End of Construction of \( A \)

From the above construction, we have \( L(A) \notin \text{UP}^A \) and \( U(A) \in \text{NP}^A \). Hence the proof is completed. Q.E.D.
5 A Complexity Theory for Closure Properties of OptP

In this section, we study potential closure properties of OptP, the class of functions computing the maximum output of NP machines.

First we define the class OptP.

**Definition 5.1 [Kre88]** For a polynomial-time nondeterministic Turing transducer $M$, $\text{opt}_N(x)$ is the mapping from $\Sigma^*$ to $\mathbb{N}$ such that for every $x \in \Sigma^*$, $\text{opt}_M(x) = \max\{n \in \text{out}_N(x)\}$ if $\text{out}_M(x) \neq \emptyset$ and 0 otherwise. And $\text{OptP} = \{\text{opt}_M | M$ is a polynomial-time nondeterministic Turing transducer\}.

We give several properties of OptP.

**Proposition 5.2** 1. [Kre88] For any function $f \in \text{OptP}$, the set $\{(x, k) | f(x) = k\}$ is in $P^{NP}$.

2. There is a function $f \in \text{OptP}$ such that for every $x \in \Sigma^*$, $f(x) > 0$ and the set $\{x | f(x)$ is odd\} is $P^P$-complete for $P^{NP}$.

Now we consider closure properties of OptP.

**Theorem 5.3** The following statements are equivalent.

1. OptP is P-closed. That is, OptP has every P-closure property.

2. OptP is OptP-closed. That is, OptP has every OptP-closure property.

3. NP = co-NP.

4. PH = NP.

5. OptP is closed under division by OptP functions.

6. OptP is closed under subtraction by OptP functions.

7. For some $k \geq 2$, OptP is closed under span of $k$ functions.

**Proof** First notice the following:

- (2) implies (1) because OptP $\supseteq$ PF.

---

*In [Kre88], a function that satisfies only completeness is given. However, with a slight modification, one can show the existence of a function satisfying the nonnegative property.*
• (1) implies (5) through (7) because division, subtraction, and span are computable in polynomial time.
• Trivially, (4) is equivalent to (3).
• (3) implies (2). This can be seen as follows: Let \( f_1, \ldots, f_k \) and \( g \) be \( \text{OptP} \) functions with \( k \geq 1 \) and define \( h(x) = g((f_1(x), \ldots, f_k(x))) \). From Proposition 5.2, the set \( A = \{(x, n_1, \ldots, n_k) | \text{for every } i, 1 \leq i \leq k, f_i(x) = n_i \} \) is in \( \text{P}^{\text{NP}} \). Assume \( \text{NP} = \text{co-NP} \).

It suffices to show that each of (5) through (7) implies (3). Let \( f \) be an \( \text{OptP} \) function in Proposition 5.2, part (2), and let \( M \) be a polynomial-time machine such that \( f = \text{opt}_M \). Define \( A = \{ x | f(x) \text{ is odd} \} \). \( A \) is \( \leq_{\text{m}}^\text{P} \)-complete for \( \text{P}^{\text{NP}} \). Define \( N \) to be the machine that, on input \( x \in \Sigma^* \), guesses \( n_1, \ldots, n_k \), nondeterministically certifies that \( z = (x, n_1, \ldots, n_k) \in A \), and simulates \( M \) on \( (n_1, \ldots, n_k) \) if \( z \in A \) is certified. Obviously, \( \text{opt}_N(x) = h(x) \) for every \( x \in \Sigma^* \).

Suppose (5) holds. Define \( h(x) = [\text{opt}_N(x)/\text{opt}_M(x)] \). There is a machine \( D \) such that \( h = \text{opt}_D \). For every \( x \in \Sigma^* \), \( h(x) = 0 \) if \( f(x) \) is odd and 1 otherwise. Thus, for every \( x \in \Sigma^* \), \( h \in A \) if and only if \( 1 \notin \text{out}_D(x) \). Therefore, \( A \in \text{co-NP}, \) and hence, \( \text{P}^{\text{NP}} = \text{NP} \).

Suppose (6) holds. Define \( h(x) = \text{opt}_M(x) \Theta \text{opt}_N(x) \). There is a machine \( D \) such that \( h = \text{opt}_D \). For every \( x \in \Sigma^* \), \( h(x) = 1 \) if \( f(x) \) is odd and 0 otherwise. Thus, for every \( x \in \Sigma^* \), \( h \in A \) if and only if \( 1 \in \text{out}_D(x) \). Therefore, \( A \in \text{NP}, \) and hence, \( \text{P}^{\text{NP}} = \text{NP} \).

Suppose (7) holds for some \( k \geq 2 \). Define \( f_1 = \cdots = f_{k-1} = \text{opt}_M \) and \( f_k = \text{opt}_N \) and define \( h(x) = \text{span}_k((f_1(x), \ldots, f_k(x))) \). There is a machine \( D \) such that \( h = \text{opt}_D \). For every \( x \in \Sigma^* \), \( h(x) = 2 \) if \( f(x) \) is odd and 1 otherwise. Thus, for every \( x \in \Sigma^* \), \( h \in A \) if and only if \( 2 \in \text{out}_D(x) \). Therefore, \( A \in \text{NP} \), and hence, \( \text{P}^{\text{NP}} = \text{NP} \).

Q.E.D.

6 A Complexity Theory for Closure Properties of MidP

In this section, we study potential closure properties of MidP, the class of functions computing the median value amongst outputs of \( \text{NP} \) machines.
Definition 6.1 For a set $S \subseteq \mathbb{N}$, $\text{left-mid}(S)$ (respectively, $\text{right-mid}(S)$) denotes the $\lceil \frac{\#S}{2} \rceil$-th smallest (respectively, largest) value in $S$ if $S \neq \emptyset$ and 0 otherwise. For a polynomial-time nondeterministic Turing transducer $M$, $\text{l-mid}_M$ (respectively, $\text{r-mid}_M$) is the mapping from $\Sigma^*$ to $\mathbb{N}$ such that for every $x \in \Sigma^*$, $\text{l-mid}_M(x) = \text{left-mid}(\text{out}_N(x))$ (respectively, $\text{r-mid}_M(x) = \text{right-mid}(\text{out}_N(x))$). And $\text{L-MidP} = \{\text{l-mid}_M \mid M$ is a polynomial-time nondeterministic Turing transducer} and $\text{R-MidP} = \{\text{r-mid}_M \mid M$ is a polynomial-time nondeterministic Turing transducer}.

Toda introduced and studied $\text{L-MidP}$ as $\text{MidP}$ in [Tod90a]. The results he proved are indeed robust for the definition of $\text{MidP}$; that is, they hold no matter which definition is used. We are interested in the difference between $\text{L-MidP}$ and $\text{R-MidP}$, and as we shall see later in this section, these classes are slightly different in terms of closure properties.

Here we give several properties of $\text{MidP}$.

Proposition 6.2 1. $\text{R-MidP} \supseteq \text{L-MidP}$.

2. [Tod90a] There exist a function $f \in \text{L-MidP} \cap \text{R-MidP}$ and a polynomial-time nondeterministic Turing transducer $M$ that satisfy the following conditions:

(a) $f = \text{l-mid}_M = \text{r-mid}_M$,
(b) $f(x) > 0$ for every $x \in \Sigma^*$,
(c) the set $\{x \mid f(x) \text{ is odd}\}$ is $\leq_p$-complete for $\text{P}^\text{PP}$,
(d) the set $\{(x, k) \mid f(x) = k\}$ is in $\text{P}^\text{PP}$, and
(e) for every function $g \in \text{L-MidP}$, there are functions $r_1$ and $r_2$ in $\text{PF}$ such that for every $x \in \Sigma^*$, $g(x) = r_2((x, f(r_1(x))))$.

Proof We only show (1). For a set $S \subseteq \mathbb{N}$ and for every $k$ larger than $\max\{n \in S\}$, $\text{left-mid}(S \cup \{0, k\}) = \text{right-mid}(S)$. Given a polynomial-time machine $M$ and a polynomial $p$ bounding the runtime of $M$, define $N$ to be the machine that, on input $x \in \Sigma^*$, either (i) outputs 0, (ii) outputs $2^p(|x|)$, or (iii) simulates $M$ on $x$. Clearly, for every $x \in \Sigma^*$, $\text{left-mid}(\text{out}_N(x)) = \text{right-mid}(\text{out}_M(x))$. Thus, $\text{R-MidP} \subseteq \text{L-MidP}$. Q.E.D.

Now we consider closure properties of $\text{L-MidP}$. Unfortunately, these results fall short of completely characterizing the $\text{P}$-closure of $\text{L-MidP}$. Nonetheless, we are able to provide sufficient and necessary conditions that are relatively tight.

\hspace{1em} In [Tod90a], the function given satisfies only (2c), (2d), and (2e). However, one can easily modify that function so that (2a) and (2b) are also satisfied.
NP(3), a level of the Boolean hierarchy over NP [CGH+88,CGH+89], is the class of sets $L$ for which there are sets $L_1$, $L_2$, and $L_3$ in NP such that $L = (L_1 \cap L_2) \cup L_3$.

**Theorem 6.3** If $L$-MidP is P-closed, then $PP = NP(3)$, and hence $CH = PH = PP = NP(3)$.

**Proof** Suppose that $L$-MidP is P-closed. Let $f$ and $M$ be a function and a machine as in Proposition 6.2, part (2). Define $A = \{x \mid f(x)$ is odd$\}$. Define $U$ (respectively, $V$) to be a machine that, on input $x \in \Sigma^*$, simulates $M$ on $x$, obtains an output $n$ of $M$, and outputs $2n$ (respectively, $n + 2|n/2|$). It is easy to see that for every $x \in \Sigma^*$,

- if $f(x)$ is odd, then $l$-mid$_U(x) = l$-mid$_V(x) + 1$ and
- if $f(x)$ is even, then $l$-mid$_U(x) = l$-mid$_V(x)$.

Define $g(x) = l$-mid$_U(x) \oplus l$-mid$_V(x)$. For every $x \in \Sigma^*$, it holds that:

- if $x \in A$, then $g(x) = 1$ and
- if $x \notin A$, then $g(x) = 0$.

From the assumption, $g \in l$-mid$_N$ for some machine $N$. We have for every $x \in \Sigma^*$,

$$x \in A \iff out_N(x) \in \{\{1\}, \{1, k\}, \{0, 1, k\}, \{0, 1, k, \ell\}\},$$

where $\ell > k > 1$. Now define $L_1 = \{x \mid 1 \in out_N(x)\}$, $L_2 = \{x \mid \#out_N(x) \geq 2\}$, and $L_3 = \{x \mid (\exists k \geq 2)(k \in out_N(x))\}$. Trivially, $L_1, L_2$, and $L_3$ are in NP and $A = L_1 \cap (L_2 \cup L_3)$. Thus, $A \in NP(3)$. Since $A$ is $\leq_m$-complete for $PP$ and $PP \supseteq NP(3)$, it holds that $PP = NP(3)$.

Furthermore, since $PP^{PH} \subseteq BP \cdot PP$, $BP \cdot PH \subseteq PH$, and $PH \subseteq PP$, assuming $PP = NP(3)$, it holds that $PP^{PH} \subseteq BP \cdot NP(3) \subseteq PH \subseteq PP \subseteq NP(3)$. Since PP is closed under truth-table reductions, we have $CH = PH = PP = NP(3)$. Q.E.D.

**Theorem 6.4** If $PP = NP$, then $L$-MidP is P-closed.

**Proof** Suppose $PP = NP$. Let $g$ be any $L$-MidP function and $h$ be any PF function and define $t(x) = h(g(x))$. It suffices to show that $t \in L$-MidP. Let $f$ be a function and $M$ be a machine as in Proposition 6.2, part (2). From Proposition 6.2, part (2e), there are functions $r_1$ and $r_2$ in PF such that for every $x \in \Sigma^*$, $g(x) = r_2((x, f(r_1(x))))$. Moreover, from (2d), the set $A = \{(x, k) \mid f(x) = k\}$ is in $PP$, and thus, from our supposition, $A \in NP$. Define $N$
to be the machine that, on input $x \in \Sigma^*$, guesses $k \geq 0$, nondeterministically certifies that $(r_1(x), k) \in A$, and outputs $h(r_2((x, k)))$. Obviously, for every $x \in \Sigma^*$, $out_N(x) = \{t(x)\}$. Therefore, $l\text{-mid}_N = t$.

On the other hand, we obtain a complete characterization of when $R$-$\text{MidP}$ is $P$-closed.

**Theorem 6.5** The following statements are equivalent.

1. $R$-$\text{MidP}$ is $P$-closed. That is, $R$-$\text{MidP}$ has every $P$-closure property.
2. $R$-$\text{MidP}$ is $R$-$\text{MidP}$-closed. That is, $R$-$\text{MidP}$ has every $R$-$\text{MidP}$-closure property.
3. $P^{PP} = NP$.
4. $PH = CH = PP = NP$.
5. $R$-$\text{MidP}$ is closed under division by $R$-$\text{MidP}$ functions.
6. $R$-$\text{MidP}$ is closed under subtraction by $R$-$\text{MidP}$ functions.

**Proof** First notice the following:

- (2) implies (1) because $R$-$\text{MidP} \supseteq PF$.
- (1) implies (5) and (6) because division and subtraction are computable within polynomial time.
- Trivially, (4) implies (3).
- (3) implies (4) because under the assumption that $P^{PP} = NP$, we have $P^{PP} \subseteq PP^{NP} \subseteq BP \cdot PP \subseteq BP \cdot NP \subseteq PH \subseteq P^{PP} \subseteq NP$.
- The proof of Theorem 6.4 also establishes that (3) implies (2).

It suffices to show that each of (5) and (6) implies (3). Let $f$ and $M$ be a function and a machine in Proposition 6.2, part (2). Define $A = \{x \mid f(x) \text{ is odd}\}$. From Proposition 6.2, part (2c), $A$ is $\leq^p_m$-complete for $P^{PP}$. Define $U$ (respectively, $V$) to be the machine that, on input $x \in \Sigma^*$, simulates $M$ on $x$, obtains an output $n$ of $M$, and outputs $2n$ (respectively, $n + 2\lfloor n/2 \rfloor$). It is not hard to see that for every $x \in \Sigma^*$,

- if $x \in A$, then $r\text{-mid}_U(x) = r\text{-mid}_V(x) + 1$, and
- if $x \notin A$, then $r\text{-mid}_U(x) = r\text{-mid}_V(x)$.
Suppose (5) holds. Define \( g(x) = \lfloor r \cdot \text{mid}_N(x) / r \cdot \text{mid}_U(x) \rfloor \). There is a machine \( N \) such that \( g = r \cdot \text{mid}_N \). For every \( x \in \Sigma^* \), it holds that if \( x \in A \), then \( r \cdot \text{mid}_N(x) = 0 \) and if \( x \not\in A \), then \( r \cdot \text{mid}_N(x) = 1 \). By definition, if \( r \cdot \text{mid}_N(x) = 0 \), then \( \text{out}_N(x) \) is either \( \emptyset \) or \( \{0\} \), and if \( r \cdot \text{mid}_N(x) = 1 \), then \( 1 \in \text{out}_N(x) \). Thus, for every \( x \in \Sigma^* \), if \( x \not\in A \) if and only if \( 1 \in \text{out}_N(x) \). Therefore, \( \overline{A} \in \text{NP} \). Hence, \( \text{P}^{\text{PP}} = \text{NP} \).

Suppose (6) holds. Define \( g(x) = r \cdot \text{mid}_U(x) \cap r \cdot \text{mid}_V(x) \). There is a machine \( N \) such that \( g = r \cdot \text{mid}_N \). For every \( x \in \Sigma^* \), it holds that if \( x \in A \), then \( r \cdot \text{mid}_N(x) = 1 \), and if \( x \not\in A \), then \( r \cdot \text{mid}_N(x) = 0 \). So for every \( x \in \Sigma^* \), if \( x \in A \) if and only if \( 1 \in \text{out}_N(x) \). Thus, \( A \in \text{NP} \). This gives \( \text{P}^{\text{PP}} = \text{NP} \). Q.E.D.

7 Conclusions

The paper proposes and develops a complexity theory of feasible closure properties. For each of the classes \( \text{NP} \), \( \text{SpanP} \), \( \text{OptP} \), and \( \text{R-MidP} \), we've proven complete characterizations—in terms of complexity class collapses—of the conditions under which the class has all feasible closure properties. In particular, \( \text{NP} \) is \( \text{P} \)-closed if and only if \( \text{P} = \text{UP} \); \( \text{SpanP} \) is \( \text{P} \)-closed if and only if \( \text{R-MidP} \) is \( \text{P} \)-closed if and only if \( \text{P}^{\text{PP}} = \text{NP} \); \( \text{OptP} \) is \( \text{P} \)-closed if and only if \( \text{NP} = \text{co-NP} \). Furthermore, for each of these classes, we've proven natural operations—such as subtraction and division—to be "hard" closure properties, in the sense that if a class is closed under one of these, then it has all feasible closure properties.

We also studied potentially intermediate closure properties for \( \text{NP} \). These properties—maximum, minimum, median, and decrement—seem neither to be possessed by \( \text{NP} \) nor to be \( \text{NP} \)-hard. We proved interrelationships among them, and structural collapses that would follow were \( \text{NP} \) to possess such closures.

Though the results of this paper provide strong evidence that the classes discussed are not closed under, for example, subtraction, we caution that the classes are not far from being closed under, for example, subtraction; given the flexibility of polynomial-time pre- and post-computation, subtraction and all other closure properties can be performed. More formally, for \( \mathcal{F} \in \{ \text{NP}, \text{SpanP}, \text{OptP}, \text{R-MidP} \} \), it holds that \( \text{PF}^\mathcal{F}_{TM} = \text{PF}^\mathcal{F}_{1-T} \), where \( \text{PF}^\mathcal{F}_{TM} \) is the class of functions computed by polynomial-time deterministic Turing machines making parallel queries to \( \mathcal{F} \), and \( \text{PF}^\mathcal{F}_{1-T} \) is the class of functions computed by polynomial-time deterministic Turing machines that making one query to \( \mathcal{F} \).\(^{10}\) This observation highlights

\(^{10}\)The proofs are not hard; for the \( \text{NP} \) case this is implicit in the work of [PZ83, Wag86, CH90], and the other cases can be shown via similar tricks (see also [Tod90a]).
the fact that the closure properties of a class are sensitive to the syntactic framing of the class.

Acknowledgments

We thank Gerhard Buntrock, William Gasarch, Ulrich Hertrampf, Albrecht Hoene, Sudhir Jha, Uwe Schöning, Jacobo Torán, and Jozef Vyskoč for enjoyable discussions. We are grateful to Yenjo Han and Klaus Wagner for helpful suggestions, and to Richard Beigel, Lance Fortnow, and Kenneth Regan for making available copies of their manuscripts and papers. We are particularly grateful to Osamu Watanabe, who organized the workshop at which this research was performed and who contributed generously and substantively to the paper, and to Alan Selman for sharing his knowledge of the history and context of nonconstructive completeness.

References


