Gap-Definable Counting Classes

Stephen A. Fenner*
Computer Science Department
University of Southern Maine
96 Falmouth Street
Portland, Maine 04103

Lance J. Fortnow†
Stuart A. Kurtz
Computer Science Department
University of Chicago
1100 East Fifty-eighth Street
Chicago, Illinois 60637

July 12, 1992

*Work done while the first author was a graduate student at the University of Chicago Computer Science Department, supported in part by a University of Chicago Fellowship.
†Supported by NSF Grant CCR-9009936

Running Head: Gap-Definable Counting Classes

Send proofs to:

Stephen Fenner
Computer Science Department
University of Southern Maine
96 Falmouth Street
Portland, Maine 04103

Abstract

The function class #**P** lacks an important closure property: it is not closed under subtraction. To remedy this problem, we introduce the function class Gap**P** as a natural alternative to #**P**. Gap**P** is the closure of #**P** under subtraction, and has all the other useful closure properties of #**P** as well. We show that most previously studied counting classes, including PP, $C_{=}P$, and $\mathrm{Mod}_k P$, are "gap-definable," i.e., definable using the values of Gap**P** functions alone. We show that there is a smallest gap-definable class, SPP, which is still large enough to contain **Few**. We also show that SPP consists of exactly those languages low for Gap**P**, and thus SPP languages are low for any gap-definable class. These results unify and improve earlier disparate results of Cai & Hemachandra [7] and Köbler, Schöning, Toda, & Torán [15]. We show further that any countable collection of languages is contained in a unique minimum gap-definable class, which implies that the gap-definable classes form a lattice under inclusion. Subtraction seems necessary for this result, since nothing similar is known for the #**P**-definable classes.

1 Introduction

In 1979, Valiant [29] defined the class #**P**, the class of functions definable as the number of accepting computations of some polynomial-time nondeterministic Turing machine. Valiant showed many natural problems complete for this class, including the permanent of a zero-one matrix. Toda [27] showed that these functions have more power than previously believed; he showed how to reduce any problem in the polynomial-time hierarchy to a single value of a #**P** function.

The class #**P** has its shortcomings, however. In particular, #**P** functions cannot take on negative values and thus #**P** is not closed under subtraction. Also, one cannot express as a #**P** function the permanent of a matrix with arbitrary (possibly negative) integer entries, or even a simple polynomial-time function which outputs negative values.

In this paper, we analyze Gap**P**, a function class consisting of differences—"gaps"—between the number of accepting and rejecting paths of NP Turing machines. This class, introduced in section 3, is exactly the closure of #**P** under subtraction. Gap**P** also has all the other nice closure properties of #**P**, such as addition, multiplication, and binomial coefficients. Beigel, Reingold, & Spielman first used gaps to great advantage in [6] to show that PP is closed under intersection. Toda and Ogiwara have also formulated their results in [28] using Gap**P** instead of #**P** (see section 3). We will argue that Gap**P** is the right way to think about #**P**-like functions.

Many complexity classes, such as NP, UP, BPP, PP, $C_{=}P$, and $\oplus P$, have definitions based on the number of accepting paths of an NP machine. In section 4 we will look at complexity classes defined in terms of the gap of an NP machine. Some classes such as PP, $C_{=}P$, and $\oplus P$ have very simple characterizations in this manner. In particular, in section 5 we study a class SPP, alluded to but not specifically named in [15]. This class has also been studied independently by Ogiwara & Hemachandra [19] under the name XP, and by Gupta [12] under the name ZUP. We show that SPP, the gap analog of UP, is the smallest of all reasonable gap-definable classes. SPP languages are exactly the low sets for Gap**P** (that is, $L \in SPP$ if and only if $\text{Gap}\mathbf{P}^L = \text{Gap}\mathbf{P}$), and thus are low for any gap-definable class. We also show that SPP equals the gap analog of **Few**, and this gives us an alternate proof that **Few** is contained in $\oplus P$ ([7, 5, 4, 15]). From containment and lowness considerations, we further conclude that P, UP, NP, and BPP are unlikely to be gap-definable.

In section 6 we address the question, first asked in [28], of whether the polynomial hierarchy (PH) is randomly reducible to SPP. We show that this question cannot be answered by relativizable techniques, that is, we show that there is an oracle relative to which NP is not randomly reducible to SPP (proposition 6.1), but with respect to a random oracle, PH is low for SPP.

In section 7, we consider the possibility that Gap**P** is closed under certain operations stronger than those discussed in section 3. We show that such closure is equivalent to certain unlikely complexity theoretic collapses. Similar, more extensive results were obtained independently for Gap**P** and #**P** in [19, 12].

In section 8, we determine structural properties of the collection of all gap-definable classes. We define GapCl, a simple, albeit nonconstructive, closure operation on sets (the 'gap-closure'). From this we show that any countable set of languages $\mathcal{C}$ has a unique minimum gap-definable class $\text{GapCl}(\mathcal{C})$ containing it. We then show that the collection of all gap-definable classes is

closed under intersection and forms a lattice under inclusion. We also show that some classes which are not obviously gap-definable in fact have this property.

Finally, we look at alternatives to the notion of gap-definability in section 9. Narrower notions of gap-definability can be advantageous, especially in light of the results in [28]. We define *nice* gap-definable classes—those for which the proofs in [28] go through. Nice classes have several other desirable properties, and most of the usual gap-definable classes are nice. On the negative side, we show that the structural results of section 8 probably do not hold for nice classes: the intersection of two nice classes is almost always as small as possible—SPP.

We pose questions for further research in section 10.

2 Notation and Definitions

We let $\Sigma = \{0, 1\}$ and let Σ^* denote the set of binary strings, which we identify with the natural numbers via the usual binary representation. We let Z denote the set of integers. In this section only, we will once or twice wish to emphasize that $\Sigma^* \subseteq Z$, so we will then write Z^+ in place of Σ^*, and reserve Σ^* to refer to the set of inputs to machines. For purposes of computation, we will also have occasion to identify Z with Σ^* in some standard way, e.g., via the usual binary representation together with an extra sign bit. For $x \in \Sigma^*$ we write $|x|$ for the length of x. A *language* is a subset of Σ^*, and unless stated otherwise, all functions have domain Σ^*. Following custom, we sometimes identify a language L with its characteristic function χ_L, so we have, for all $x \in \Sigma^*$,

$$L(x) \stackrel{df}{=} \chi_L(x) \stackrel{df}{=} \begin{cases} 1 & \text{if } x \in L, \\ 0 & \text{if } x \notin L. \end{cases}$$

If A and B are sets, we let $B - A$ denote the relative complement of A in B. If $A \subseteq \Sigma^*$, we usually use $\overline{A}$ as shorthand for $\Sigma^* - A$. If $A, B \subseteq \Sigma^*$, we use $A \oplus B$ to denote the *join* of A and B:

$$A \oplus B \stackrel{df}{=} \{2x \mid x \in A\} \cup \{2x + 1 \mid x \in B\}.$$

We assume the reader is familiar with the basic concepts of computational complexity theory, including Turing machines, complexity classes, polynomial-time reductions (m-reductions and Turing reductions, and to a lesser extent, tt-reductions), complete sets, nondeterminism, relativization, etc. We also assume that the reader has basic knowledge of computable functions and recursively enumerable (r.e.) sets. There are a number of good textbooks covering these subjects, including [13]. We use P and FP to denote the classes of all polynomial-time computable languages and functions respectively. We use NP to denote the class of all languages computable in nondeterministic polynomial time, and PH denotes the polynomial hierarchy (see [25]).

We say informally that a class of languages or functions is *relativizable* if its definition refers—explicitly or implicitly—to computation and/or computing machines. If $\mathcal{C}$ is a relativizable class and $L \subseteq \Sigma^*$, we follow custom and define $\mathcal{C}^L$ by replacing each machine directly or indirectly referenced in the definition of $\mathcal{C}$ with an oracle machine with similar properties except that the new machine has access to L as an oracle. We write $\mathcal{C}^\emptyset$ simply as $\mathcal{C}$ as is customary. If $\mathcal{D}$ is a language class, we write $\mathcal{C}^\mathcal{D}$ for the set $\bigcup_{L \in \mathcal{D}} \{\mathcal{C}^L\}$ as usual. We say L is *low* for $\mathcal{C}$ if $\mathcal{C}^L = \mathcal{C}$.

A class of languages is *low* for $\mathcal{C}$ if every language in the class is low for $\mathcal{C}$. This notion was borrowed from recursion theory and was applied to complexity classes in, for example, [15] (see [15] for further references).

We now define the machines we will be considering.

Definition 2.1 *A counting machine (CM) is a nondeterministic Turing machine running in polynomial time with two halting states: accepting and rejecting, and every computation path must end in one of these states. An oracle machine having the above properties and running in polynomial time uniformly for all oracles is called an oracle counting machine (OCM).*

A counting machine is simply an NP machine. We use this alternate terminology to emphasize that the machine's acceptance criterion is based on the *number* of accepting and/or rejecting paths. The following notions all pertain to CM's.

Definition 2.2 *Let M be a CM. We define the function $\#M\colon \Sigma^* \to Z^+$ to be such that for all $x \in \Sigma^*$, $\#M(x)$ is the number of accepting computation paths of M on input x. Similarly, $Total_M\colon \Sigma^* \to Z^+$ is the total number of computation paths of M on input x. The CM $\overline{M}$ is the machine identical to M but with the accepting and rejecting states interchanged (thus $\overline{M}$ rejects whenever M accepts and vice versa).*

Notice that for all $x \in \Sigma^*$,

$$\#M(x) + \#\overline{M}(x) = \mathrm{Total}_M(x) = \mathrm{Total}_{\overline{M}}(x),$$

and $\#\overline{M}(x)$ is the number of rejecting paths of M on input x.

If M is a CM, we define the *nondeterministic branching degree* of M to be the maximum number of possible successors to any instantaneous description (ID) of M. For any computation path p of M, we define $rank(p)$ (the *rank* of p) to be the number of nondeterministic moves made along p, that is, $rank(p)$ is the number of ID's along p with more than one successor (the halting ID's have no successors). A CM M is in *normal form* if it has nondeterministic branching degree at most two, and the rank of any computation path of M is always equal to a fixed positive polynomial in the length of the input. Thus if M is in normal form, then $\mathrm{Total}_M(x) = 2^{q(|x|)}$ for some positive polynomial q. ¿From now on, all machines will be denoted with the capital letters M and N, possibly with primes or subscripts, and will be CM's unless stated otherwise.

We now define some of the usual counting classes. These are not always the original definitions, but can easily be shown to be equivalent to them. See [4] for more details.

Definition 2.3

- *(Valiant [29])* $\#\boldsymbol{P} \stackrel{df}{=} \{\#M \mid M \text{ is a CM}\}$.
- *(Gill [9]) PP is the class of all languages L such that there exists M and an FP function f such that, for all x,*

  $$x \in L \iff \#M(x) > f(x).$$

  The function f is the threshold of M.

- *(Wagner [30]) $C_=P$ is the class of all languages L such that there exists M and an FP function f such that, for all x,*

$$x \in L \iff \#M(x) = f(x).$$

- *(Beigel, Gill, Hertrampf [5]) For $k \geq 2$, define Mod_kP to be the class of all languages L such that there exists M such that, for all x,*

$$x \in L \iff \#M(x) \not\equiv 0 \bmod k.$$

The class Mod_2P is also called $\oplus P$ ('Parity P'). This class was defined by Papadimitriou & Zachos [20] and by Goldschlager & Parberry [10] (see [4] for details). The following two classes will also be of interest to us:

Definition 2.4

- *(Allender [1]) For any language L, $L \in$ **FewP** if and only if there exist a CM M and a polynomial p such that for all $x \in \Sigma^*$, $\#M(x) \leq p(|x|)$ and*

$$x \in L \iff \#M(x) > 0.$$

- *(Cai & Hemachandra [7]) For any language L, $L \in$ **Few** if and only if there exist a CM M, a polynomial p, and a polynomial-time computable predicate $A(x, y)$ such that for all $x \in \Sigma^*$, $\#M(x) \leq p(|x|)$ and*

$$x \in L \iff A(x, \#M(x)).$$

Clearly, $\mathbf{FewP} \subseteq NP$. This is not known for **Few**, but it is well-known that $\mathbf{Few} \subseteq P^{NP[\log]}$, and in fact, $\mathbf{Few} \subseteq P^{\mathbf{FewP}}$ [14].

3 Gaps

Definition 3.1 *If M is a CM, define the function $gap_M : \Sigma^* \to Z$ as follows:*

$$gap_M \stackrel{df}{=} \#M - \#\overline{M}.$$

The function gap_M represents the "gap" between the number of accepting and the number of rejecting paths of M. We define the natural gap analog of the function class $\#\mathbf{P}$:

Definition 3.2

$$Gap\boldsymbol{P} \stackrel{df}{=} \{gap_M \mid M \text{ is a CM}\}.$$

This class was defined independently in [12] and named $\mathbf{Z}\#\mathbf{P}$. From now on in this chapter, we follow the spirit of [6] and work almost exclusively with gaps. The advantages are that gap functions can take on positive and negative values, and we can subtract gaps without introducing the large offsets that we get when we are counting accepting paths only. We can add and multiply gaps as well, thus $\mathrm{Gap}\mathbf{P}$ has a canonical ring structure.

Lemma 3.3 *For every CM M, there is a CM N such that $gap_N = \#M$. (That is, $\#\boldsymbol{P} \subseteq Gap\boldsymbol{P}$.)*

Proof: Given an input x, the machine N guesses a path p of $M(x)$. If p is accepting, N accepts. Otherwise, N branches once, accepting on one branch and rejecting on the other. We have, for all x,

$$\begin{aligned} \mathrm{gap}_N(x) &= \#N(x) - \#\overline{N}(x) \\ &= \#N(x) - \#\overline{M}(x) \\ &= \#M(x) + \#\overline{M}(x) - \#\overline{M}(x) \\ &= \#M(x). \end{aligned}$$

□

It is clear that gaps are no harder to compute than numbers of accepting paths. Proposition 3.5 gives (perhaps) the strongest statement of this fact.

Definition 3.4 *If $\mathcal{C}$ and $\mathcal{D}$ are two function classes, define*

$$\mathcal{C} \diamond \mathcal{D} \stackrel{df}{=} \{f \diamond g \mid f \in \mathcal{C} \ \& \ g \in \mathcal{D}\},$$

where $\diamond$ is some appropriate binary operation, i.e., addition, subtraction, composition, etc.

Proposition 3.5

$$Gap\mathbf{P} = \#\mathbf{P} - \#\mathbf{P} = \#\mathbf{P} - FP = FP - \#\mathbf{P}.$$

(Note that here the minus sign refers to elementwise subtraction, not to set theoretic complement.)

Proof: For any M we have

$$\mathrm{gap}_M = \#M - \#\overline{M}$$

by definition, so Gap$\mathbf{P} \subseteq \#\mathbf{P} - \#\mathbf{P}$. To show that $\#\mathbf{P} - \#\mathbf{P} \subseteq \#\mathbf{P} - FP$, let f and g be $\#\mathbf{P}$ functions. We can assume that $f = \#M$ and $g = \#N$, where M and N are CM's, and N is in normal form with polynomial q (just pad N with extra rejecting paths so that all paths have rank q; the result is a normal form machine with the same number of accepting paths). Let M' be the machine which first branches once, then simulates M on one branch and $\overline{N}$ on the other. We have, for any x,

$$\begin{aligned} f(x) - g(x) &= \#M(x) - \#N(x) \\ &= \#M(x) + \#\overline{N}(x) - 2^{q(|x|)} \\ &= \#M'(x) - 2^{q(|x|)}. \end{aligned}$$

Therefore $f - g \in \#\mathbf{P} - FP$, and the inclusion holds. To show that $\#\mathbf{P} - FP \subseteq$ Gap$\mathbf{P}$, let f be a $\#\mathbf{P}$ function and let $g \in FP$. By lemma 3.3 there is an M such that $f = \mathrm{gap}_M$. Let N be such that for all $x \in \Sigma^*$, $N(x)$ resembles $M(x)$ padded with $g(x)$ rejecting paths. Clearly, $\mathrm{gap}_N = f - g$. It now follows that the first two equalities hold. The last equality holds since $\#\mathbf{P} - \#\mathbf{P}$ is closed under negation. □

We might just as well have taken the first equality in proposition 3.5 as the definition of Gap$\mathbf{P}$, and altered the proofs below accordingly. This route was indeed taken in [28]. We nonetheless prefer to use our original definition in this chapter, if only for the conceptual ease of associating a single machine to every Gap$\mathbf{P}$ function.

We now list the closure properties of Gap$\mathbf{P}$, deferring the proofs until afterwards. It is well known that properties 1, 3, 4, and 5 below are also shared by $\#\mathbf{P}$. Property 2 clearly is not shared by $\#\mathbf{P}$. It is this property that gives Gap$\mathbf{P}$ its power. Property 6 seems to depend heavily on property 2, so we don't believe *it* is shared with $\#\mathbf{P}$ either. From these properties, it is easy to see that the permanent of an arbitrary integer matrix can be computed in Gap$\mathbf{P}$, although it cannot be computed in $\#\mathbf{P}$.

Closure Property 1 *Gap$\boldsymbol{P} \circ FP =$ Gap$\boldsymbol{P}$ and $FP \subseteq$ Gap$\boldsymbol{P}$.*

Closure Property 2 *If $f \in$ Gap$\boldsymbol{P}$ then $-f \in$ Gap$\boldsymbol{P}$.*

Closure Property 3 *If $f \in$ Gap$\boldsymbol{P}$ and q is a polynomial, then the function*

$$g(x) \stackrel{df}{=} \sum_{|y| \leq q(|x|)} f(\langle x, y \rangle)$$

is in Gap$\boldsymbol{P}$.

Closure Property 4 *If $f \in$ Gap$\boldsymbol{P}$ and q is a polynomial, then the function*

$$g(x) \stackrel{df}{=} \prod_{0 \leq y \leq q(|x|)} f(\langle x, y \rangle)$$

is in Gap$\boldsymbol{P}$.

Closure Property 5 *If $f \in$ Gap$\boldsymbol{P}$, $k \in FP$, and $k(x)$ is bounded by a polynomial in $|x|$, then the function*

$$g(x) \stackrel{df}{=} \binom{f(x)}{k(x)}$$

is in Gap$\boldsymbol{P}$.

Closure Property 6 *If $f, g \in$ Gap$\boldsymbol{P}$ and $0 \leq g(x) \leq q(|x|)$ for some polynomial q, then the function*

$$h(x) \stackrel{df}{=} f(\langle x, g(x) \rangle)$$

is in Gap$\boldsymbol{P}$.

Corollary 3.6 *Gap$\boldsymbol{P}$ is closed under addition, subtraction, and multiplication.*

Proof: Let f_1 and f_2 be in Gap$\mathbf{P}$. Let N be a CM such that for all x, $\mathrm{gap}_N(\langle x, 0 \rangle) = f_1(x)$, $\mathrm{gap}_N(\langle x, 1 \rangle) = f_2(x)$, and $\mathrm{gap}_N(\langle x, i \rangle) = 0$ for $i \geq 2$. For addition and multiplication, apply closure properties 3 and 4, respectively, with $q(x) \geq 2$ arbitrary. Subtraction follows from addition and closure property 2. $\square$

Proof of Closure Property 1: Given a CM M and $g \in FP$. Let N be such that $N(x)$ simulates $M(g(x))$ for all $x \in \Sigma^*$. Clearly, N is a CM and $\mathrm{gap}_N = \mathrm{gap}_M \circ g$. The second statement follows immediately from lemma 3.3. □

Proof of Closure Property 2: Immediate from proposition 3.5. □

Proof of Closure Property 3: If $f = \mathrm{gap}_M$ for some CM M, then there is a CM N which first guesses a y of length not greater than $q(|x|)$, then simulates M on input $\langle x, y\rangle$ for each y guessed. Clearly, $g = \mathrm{gap}_N$. □

Proof of Closure Property 4: Given $f = \mathrm{gap}_M$, the machine N guesses, in sequence, computation paths of M on the inputs $\langle x, 0\rangle$, $\langle x, 1\rangle$, $\langle x, 2\rangle$, and so on through $\langle x, q(|x|)\rangle$. N accepts if an even number of these paths are rejecting, and N rejects if an odd number of these paths are rejecting. N is clearly a CM. The fact that $g = \mathrm{gap}_N$ can be shown by induction on the value $n = q(|x|)$ as follows: for $n = 0$, we have $\mathrm{gap}_N(x) = f(\langle x, 0\rangle) = g(x)$ because $N(x)$ behaves just as $M(\langle x, 0\rangle)$ does. If $n > 0$, assume true for $n - 1$, and let N' be a machine that acts the same as N except that N' only guesses paths of M on inputs $\langle x, 0\rangle, \ldots, \langle x, n-1\rangle$. For convenience, let $a_{N'} \stackrel{df}{=} \#N'(x)$, $r_{N'} \stackrel{df}{=} \#\overline{N'}(x)$, $a_M \stackrel{df}{=} \#M(\langle x, n\rangle)$, and $r_M \stackrel{df}{=} \#\overline{M}(\langle x, n\rangle)$. By the inductive hypothesis, we have

$$\begin{aligned} g(x) &= \mathrm{gap}_{N'}(x) f(\langle x, n\rangle) \\ &= \mathrm{gap}_{N'}(x) \mathrm{gap}_M(\langle x, n\rangle) \\ &= (a_{N'} - r_{N'})(a_M - r_M) \\ &= (a_{N'} a_M + r_{N'} r_M) - (a_{N'} r_M + r_{N'} a_M). \end{aligned}$$

Now $N(x)$ accepts whenever it guesses an even number of rejecting paths. This happens either when there are an even number of rejections through $\langle x, n-1\rangle$ and the last path is accepting, or when there are an odd number of rejections through $\langle x, n-1\rangle$ and the last path is rejecting. Thus by the definition of N', the total number of sequences accepted by $N(x)$ is exactly $a_{N'} a_M + r_{N'} r_M$. Likewise, the total number of sequences rejected by $N(x)$ is $a_{N'} r_M + r_{N'} a_M$. Therefore $g(x) = \mathrm{gap}_N(x)$. □

Of all the closure properties, property 5 is perhaps the most useful and least obvious. It states that, like #**P**, Gap**P** is closed under binomial coefficients. To prove this closure property, we will need a combinatorial lemma (lemma 3.7). We define the binomial coefficient as follows:

$$\binom{x}{y} \stackrel{df}{=} \frac{x(x-1)(x-2)\cdots(x-y+1)}{y!},$$

which makes sense for all real numbers x and all nonnegative integers y. (If $y = 0$ then $\binom{x}{y} \stackrel{df}{=} 1$ by convention.) Lemma 3.7 is proved using Vandermonde's convolution [11, page 174], which states that for integers a, b and $k \geq 0$,

$$\binom{a+b}{k} = \sum_{i=0}^{k} \binom{a}{i}\binom{b}{k-i}.$$

An intuition behind this equality is that choosing a committee of k people from a group of a women and b men is the same as first choosing i women then $k-i$ men independently for each possible i.

Lemma 3.7 *For all integers r, j, k with $k \geq 0$,*

$$\binom{j}{k} = \sum_{i=0}^{k} (-1)^i \binom{r+i}{i}\binom{r+j+1}{k-i}.$$

Proof: Negate the first binomial coefficient on the right hand side (see [11, page 174]) to get

$$\text{(right hand side)} = \sum_{i=0}^{k} \binom{-r-1}{i}\binom{r+j+1}{k-i}.$$

Now apply Vandermonde's convolution to get

$$\sum_{i=0}^{k} \binom{-r-1}{i}\binom{r+j+1}{k-i} = \binom{j}{k}.$$

□

It is important to note that the identity of lemma 3.7 holds for *all* integers j and nonnegative integers k.

Proof of Closure Property 5: Note that if M_1 is a CM running in time $t \stackrel{df}{=} t(n)$, and $k \stackrel{df}{=} k(x)$ is a FP function, then there is a nondeterministic machine M_2 running in time $O(kt)$ such that for all inputs x of length n,

$$\#M_2(x) = \binom{\#M_1(x)}{k(x)}.$$

The machine M_2 simply guesses a sequence of k paths of M_1, and accepts if and only if the paths are in strictly increasing lexicographical order and all of them are accepting. Note further that if $k(x)$ and $t(n)$ are polynomially bounded, then M_2 is a CM.

Let $f = \mathrm{gap}_M$. By setting $r \stackrel{df}{=} \#\overline{M}(x)$, $j \stackrel{df}{=} \mathrm{gap}_M(x)$, and $k \stackrel{df}{=} k(x)$ in lemma 3.7 above, we get

$$\binom{f(x)}{k(x)} = \sum_{i=0}^{k(x)} (-1)^i \binom{\#\overline{M}(x)+i}{i} \binom{\#M(x)+1}{k(x)-i}.$$

By the previous paragraph and lemma 3.3, there is a machine N that can generate a gap equal to each of the binomial coefficients on the right-hand side. By lemma closure properties 1 through 4, it can combine these gaps to generate the whole right-hand side as a gap. (The machine N computes the factors by padding M with one accepting path, padding $\overline{M}$ with i accepting paths, then computing the resulting binomial coefficients.) □

The following "delta" functions will be useful in many places later on: for integers k and B with $0 \leq k \leq B$, define

$$\delta_k^B(x) \stackrel{df}{=} \binom{x}{k}\binom{B-x}{B-k}$$

for all $x \in Z$. Notice that

$$\delta_k^B(x) = \begin{cases} 0 & \text{if } 0 \leq x < k, \\ 1 & \text{if } x = k, \\ 0 & \text{if } k < x \leq B. \end{cases}$$

We now use these delta functions to prove closure property 6, which says that Gap**P** is closed under a limited form of composition.

Proof of Closure Property 6: Notice that for any x,

$$f(\langle x, g(x)\rangle) = \sum_{i=0}^{q} f(\langle x, i\rangle)\delta_i^q(g(x)),$$

where $q = q(|x|)$. The statement follows by the previous closure properties. □

Closure property 6 immediately gives a number of other limited closure properties, among them a strengthening of closure property 5.

Corollary 3.8 *If $f, g \in$ Gap**P** and $0 \leq g(x) \leq q(|x|)$ for some polynomial q, then the functions*

$$\binom{f(x)}{g(x)} \quad \text{and} \quad f(x)^{g(x)}$$

*are in Gap**P**.*

Proof: Apply closure property 6 with $\hat{f}$ and g, where

$$\hat{f}(\langle x, i\rangle) \stackrel{df}{=} \binom{f(x)}{i}$$

for the first function, and

$$\hat{f}(\langle x, i\rangle) \stackrel{df}{=} f(x)^i$$

for the second. □

4 Counting Classes

Most counting classes that have been studied previously can be defined using the gap function alone. We will call such classes *gap-definable*.

Definition 4.1 *A class $\mathcal{C}$ of languages is gap-definable if there exist disjoint sets $A, R \subseteq \Sigma^* \times Z$ such that, for any language L, $L \in \mathcal{C}$ if and only if there exists a CM M with*

$$\begin{aligned} x \in L &\implies (x, gap_M(x)) \in A, \\ x \notin L &\implies (x, gap_M(x)) \in R, \end{aligned}$$

for all $x \in \Sigma^*$. We let Gap(A, R) denote the class $\mathcal{C}$.

We call A and R respectively the *accepting* and *rejecting* sets. We allow them to be completely arbitrary, perhaps nonrecursive. We say that a CM M is (A, R)-*proper* if $(x, \mathrm{gap}_M(x)) \in A \cup R$ for all $x \in \Sigma^*$, and we define

$$L_{A,R}(M) \stackrel{df}{=} \{x \in \Sigma^* \mid (x, \mathrm{gap}_M(x)) \in A\}.$$

To relativize definition 4.1 to an arbitrary fixed oracle, we permit M to be an OCM with access to that oracle. It must be noted, however, that A and R are arbitrary sets independent of any machine. Therefore we have two natural ways of defining gap-definability for a relativized class: we say that a relativized class is *uniformly* gap-definable if it is gap-definable with respect to any oracle, but with the sets A and R fixed and independent of the oracle; a relativized class is *nonuniformly* gap-definable if it is gap-definable with respect to any oracle, where A and R are chosen after the oracle and thus may vary depending on the oracle. This distinction will be important in section 5, especially for corollary 5.7. For now, unless otherwise stated, when we relativize a class Gap(A, R) to an oracle, A and R will remain fixed independent of the oracle, in accordance with our remarks at the beginning of section 2.

There are other more restricted notions of gap-definability that are possible. For a discussion of some of these alternate definitions, see section 9.

Proposition 4.2 *The classes PP, $C_=P$, and Mod_kP (for $k \geq 2$) are all (uniformly) gap-definable; in fact, the following are true for any language L:*

1. $L \in PP \iff (\exists M)(\forall x)[x \in L \leftrightarrow gap_M(x) > 0]$.
2. $L \in C_=P \iff (\exists M)(\forall x)[x \in L \leftrightarrow gap_M(x) = 0]$.
3. $L \in Mod_kP \iff (\exists M)(\forall x)[x \in L \leftrightarrow gap_M(x) \not\equiv 0 \bmod k]$.

The proof of proposition 4.2, given below, is straightforward with the aid of a normal form lemma. Unlike the case with #**P** machines, we cannot assume that Gap**P** machines are in normal form (a normal form machine always generates an even gap, for example). The following lemma is almost as good.

Lemma 4.3 *Let f be a function from Σ^* to Z. Then $f = gap_M$ for some CM M in normal form if and only if $f = 2gap_N$ for some arbitrary CM N.*

Proof: Suppose M is a CM in normal form, and let $q(|x|)$ be the rank of any path of M on input x. The machine N guesses a partial path p of $M(x)$ up through the first $q(|x|)-1$ nondeterministic choices. Let p_1 and p_2 be the two extensions of p made by the last branch of M. If both p_1 and p_2 are accepting, then N accepts; if they are both rejecting, N rejects; otherwise, N branches once to one accepting and one rejecting path. From this it is clear that $\mathrm{gap}_M = 2\mathrm{gap}_N$.

Conversely, let N be a CM (not necessarily in normal form). We can assume without loss of generality that N has branching degree at most two. Let q be a polynomial which is strictly greater than the running time of N. The machine M simulates $N(x)$, branching as N does, to guess a path p of N. Then M branches further, extending p with $2^{q(|x|)-rank(p)}$ paths, all of rank $q(|x|)$. If p is an accepting path, M makes exactly $2^{q(|x|)-rank(p)-1}+1$ of these paths to be accepting; if p rejects, then M makes $2^{q(|x|)-rank(p)-1}-1$ of these paths accepting. The contribution to $\mathrm{gap}_M(x)$ of the paths extending p is respectively $+2$ or -2, depending on whether p accepts or rejects. Therefore, $M(x)$ generates twice the gap of $N(x)$, and M is in normal form. □

Proof of Proposition 4.2: All the left-to-right implications follow immediately from lemma 3.3 and the fact that we can subtract a polynomial time computable function from a gap. The first two right-to-left implications are clear by lemma 4.3; we take the threshold function f to be $2^{q(|x|)-1}$, where q is the polynomial associated with the normal form machine. We show the third right-to-left implication by building a CM whose number of accepting paths is congruent mod k to the gap of a given CM as follows: given a CM M, let N be a CM that first generates k branches, then simulates M on one branch and $\overline{M}$ on the other $k-1$ branches. Clearly,

$$\#N = \#M + (k-1)\cdot\#\overline{M} = \mathrm{gap}_M + k\cdot\#\overline{M},$$

so

$$\#N \equiv \mathrm{gap}_M \bmod k.$$

The implication follows.

This proof clearly relativizes, so all the classes mentioned in proposition 4.2 are uniformly gap-definable. □

Lemma 4.3 allows us to characterize Gap**P** in terms of predicates in P.

Proposition 4.4 *If $f:\Sigma^* \to Z$ is any function, then $f \in$ Gap**P** if and only if there is a predicate $R(x,y) \in P$ and a positive polynomial q such that for all $x \in \Sigma^*$,*

$$f(x) = \frac{1}{2}\left(\left|\left\{y \in \{0,1\}^{q(|x|)} : R(x,y)\right\}\right| - \left|\left\{y \in \{0,1\}^{q(|x|)} : \neg R(x,y)\right\}\right|\right).$$

Proof: Immediate by lemma 4.3. □

There is yet another characterization of Gap**P** as the class of functions computed by uniform families of retarded arithmetic programs as described by Babai and Fortnow [2, section 3].

Subtraction has been quite useful in simplifying many existing proofs about counting classes. As an easy example, consider the following proof that $C_{=}P \subseteq PP$:

Proof: Given $L \in C_{=}P$ as witnessed by $f \in \mathrm{Gap}\mathbf{P}$, define

$$g(x) \stackrel{df}{=} 1 - [f(x)]^2.$$

Clearly, $g \in \mathrm{Gap}\mathbf{P}$, and for all x,

$$x \in L \Longleftrightarrow g(x) > 0.$$

Thus $L \in PP$. $\square$

The reader may wish to compare the proof above with the one in [23].

More significantly, Toda and Ogiwara [28] have simplified their results using Gap**P**. We state their main results here, using slightly altered notation. We first define a subfamily of the gap-definable classes.

Definition 4.5 *Let $Q \subseteq Z$ be any set. Define*

$$GapIn[Q] \stackrel{df}{=} Gap(\Sigma^* \times Q, \Sigma^* \times (Z - Q)).$$

Thus GapIn$[Q]$ identifies those gap-definable classes where the accepting and rejecting sets partition $\Sigma^* \times Z$ and the acceptance criterion is independent of the input. Next, we define the $\widehat{BP}$ operator from [28], which is a modification of the **BP** operator of Schöning [22]:

Definition 4.6 ([28], Definition 2.1) *Let $\mathcal{K}$ be any class of languages. A language L is in $\widehat{BP} \cdot \mathcal{K}$ if for every polynomial e, there exist a set $A \in \mathcal{K}$ and a polynomial p such that for every $x \in \Sigma^*$,*

$$|\{w : |w| = p(|x|) \ \ \& \ \ (x \in L \leftrightarrow \langle x, w\rangle \in A)\}| \geq 2^{p(|x|)}(1 - 2^{-e(|x|)}).$$

Remark: Schöning's **BP** operator is defined similarly, except that the polynomial e is replaced with the constant 2. The class BPP (bounded error probabilistic polynomial time) can be defined naturally as $BP \cdot P$.

Toda and Ogiwara showed the following technical lemma:

Lemma 4.7 ([28, Lemma 2.3]) *Let F be any function in Gap$\mathbf{P}^{PH}$ and let e be any polynomial. Then there exist a function $H \in Gap\mathbf{P}$ and a polynomial s such that for every $x \in \Sigma^*$,*

$$|\{w : |w| = s(|x|) \ \ \& \ \ H(\langle x, w\rangle) = F(x)\}| \geq 2^{s(|x|)}(1 - 2^{-e(|x|)}).$$

Their main theorem follows easily:

Theorem 4.8 ([28, Theorem 2.4]) *Let Q be an arbitrary subset of Z. Then*

$$GapIn[Q]^{PH} \subseteq \widehat{BP} \cdot GapIn[Q].$$

This theorem states that PH is "randomly low" for every gap-definable class of the form GapIn$[Q]$. One must bear in mind, however, that the result probably does not extend to all gap-definable classes. See section 6 below.

5 SPP

In definition 4.1, the accepting and rejecting sets need not partition $\Sigma^* \times Z$. That is, we can define new gap-definable counting classes by putting restrictions on the behavior of CM's. We will be interested chiefly in the following class:

Definition 5.1 *SPP is the class of all languages L such that there exists M such that, for all x,*

$$\begin{aligned} x \in L &\implies gap_M(x) = 1, \\ x \notin L &\implies gap_M(x) = 0. \end{aligned}$$

An SPP-like machine was first described in [15], and as mentioned earlier, SPP is the same class as XP and ZUP, studied independently in [19] and [12] respectively. These papers study closure properties of #**P** and Gap**P**. Recently, Köbler, Schöning, & Torán [17] showed that the Graph Automorphism problem (does a given graph have any nontrivial automorphisms) is in SPP. They also showed that the Graph Isomorphism problem is in the class $LWPP$, defined at the end of this section.

Clearly $SPP \subseteq C_=P \cap \text{co-}C_=P$. It is also clear by lemma 3.3 that $UP \subseteq SPP \subseteq \text{Mod}_k P$ for any k. Notice that if we replace gap_M with $\#M$ in the definition of SPP, we get UP. Thus on purely syntactic grounds, we might have called this class Gap-UP, although UP bears little resemblance to its gap analog (SPP is closed under complements, for example). In the same spirit, we may define the gap analog of the class **Few**:

Definition 5.2 *Gap-**Few** is the class of all languages L such that there exists a CM M, a polynomial time predicate $A(x,k)$, and a polynomial q such that, for all x of length n,*

$$0 \leq gap_M(x) \leq q(n),$$

and

$$x \in L \iff A(x, gap_M(x)).$$

If we replace gap_M with $\#M$ above, we get the class **Few**. Clearly, **Few** $\subseteq$ Gap-**Few** by lemma 3.3. It is not obvious, however, that Gap-**Few** is a gap-definable class. The reason is that we must fix the accepting and rejecting sets in advance to work for *all* predicates $A(x,k)$. It is not clear how we can do this. Theorem 5.9, however, provides a relativizable proof that Gap-**Few** $= SPP$, which implies Gap-**Few** is gap-definable, and indeed uniformly gap-definable.

The sets A and R of definition 4.1 can be chosen arbitrarily (as long as they are disjoint). This freedom allows for many small, uninteresting gap-definable classes. For example, if L is any language, then $\{L\}$ is clearly gap definable:

$$\{L\} = \text{Gap}(L \times Z, \overline{L} \times Z).$$

To avoid these cases, we concentrate on *reasonable* gap-definable classes.

Definition 5.3 *A gap-definable class $\mathcal{C}$ is reasonable if $\emptyset \in \mathcal{C}$ and $\Sigma^* \in \mathcal{C}$.*

All the gap-definable classes introduced above are clearly reasonable. The next theorem implies that SPP is the smallest reasonable gap-definable class.

Theorem 5.4 *Let $\mathcal{C} \stackrel{df}{=} Gap(A,R)$ be a gap-definable class. The following are equivalent:*

1. *$\mathcal{C}$ is reasonable.*
2. *$SPP \subseteq \mathcal{C}$.*
3. *There exist $f, g \in Gap\boldsymbol{P}$ such that $(x, f(x)) \in A$ and $(x, g(x)) \in R$ for all $x \in \Sigma^*$.*

Proof: We show $1 \Longrightarrow 3 \Longrightarrow 2 \Longrightarrow 1$.

$(1 \Longrightarrow 3)$: Let M and N be CM's recognizing $\emptyset$ and Σ^*, respectively. Let $f \stackrel{df}{=} \mathrm{gap}_N$ and $g \stackrel{df}{=} \mathrm{gap}_M$.

$(3 \Longrightarrow 2)$: Suppose $L \in SPP$ is recognized by the CM M with gap either 0 or 1. By corollary 3.6, there is a CM N such that

$$\mathrm{gap}_N = \mathrm{gap}_M \cdot (f - g) + g.$$

Thus $L \in \mathcal{C}$ as witnessed by the machine N.

$(2 \Longrightarrow 1)$: Obvious. $\Box$

We still have a great deal of freedom in choosing A and R to get reasonable gap-definable classes. In fact, it will be shown in section 8 that any countable collection of languages is contained in a reasonable gap-definable class, which in turn implies that there are uncountably many reasonable gap-definable classes.

The next theorem says that SPP consists of exactly those languages which are low for Gap$\mathbf{P}$.

Theorem 5.5

$$SPP = \{L \mid Gap\boldsymbol{P}^L = Gap\boldsymbol{P}\}.$$

Remark: It is unlikely that $\#\mathbf{P}^{SPP} = \#\mathbf{P}$, or even that $\#\mathbf{P}^{UP} = \#\mathbf{P}$. It follows immediately from arguments in [16] that the latter equality implies $UP =$ co-UP.

Proof of Theorem 5.5: We first show that SPP contains all Gap$\mathbf{P}$-low languages. Suppose L is a language such that Gap$\mathbf{P}^L =$ Gap$\mathbf{P}$. Let M be an OCM that, on input x, queries the oracle on x. If x is in the oracle, M accepts; if x is not in the oracle, M generates one accepting and one rejecting path. Clearly,

$$\mathrm{gap}_{M^L}(x) = \begin{cases} 1 & \text{if } x \in L, \\ 0 & \text{otherwise.} \end{cases}$$

By hypothesis, there is a CM N which computes the same gap as M^L but without an oracle. Thus $L \in SPP$ as witnessed by N.

Conversely, we show that if M is an OCM and L is a language in SPP, there is a CM N (without an oracle) such that

$$\mathrm{gap}_N = \mathrm{gap}_{M^L}.$$

This part of the proof has the same flavor as the proof that $\oplus P^{\oplus P} = \oplus P$ in [20]. Let M_1 be an SPP machine recognizing L. We may assume without loss of generality that for any oracle A and input x of length n, $M^A(x)$ makes exactly $k(1^n)$ oracle queries on each path, where $k \in FP$.

Fix n and let $k \stackrel{df}{=} k(1^n)$. The CM N does the following—in sequence—on input x of length n:

1. Guesses a sequence $a_1, \ldots, a_k$ of bits (oracle query answers).
2. Guesses a legal path of M, substituting a_i for the answer to the ith oracle query q_i of M. (Let p be the computation path of N defined thus far.)
3. Generates a gap G_p extending p, where G_p is defined as follows: for $1 \leq i \leq k$ let

$$g_i \stackrel{df}{=} \begin{cases} \mathrm{gap}_{M_1}(q_i) & \text{if } a_i = 1, \\ 1 - \mathrm{gap}_{M_1}(q_i) & \text{if } a_i = 0. \end{cases}$$

   If p ends in an accepting state of M, $G_p \stackrel{df}{=} \prod_{i=1}^{k} g_i$. If p ends in a rejecting state, $G_p \stackrel{df}{=} -\prod_{i=1}^{k} g_i$.

For each path p above, N can clearly generate the corresponding gap G_p by simulating M_1 in polynomial time, as is evident by the expressions for G_p and the closure properties of Gap$\mathbf{P}$.

We have

$$g_i = \begin{cases} 1 & \text{if } a_i = L(q_i), \\ 0 & \text{otherwise.} \end{cases}$$

Thus for any path p above, $G_p = \pm 1$ if all of M_1's queries were answered correctly along p (i.e., according to the language L), and $G_p = 0$ otherwise. Thus paths with incorrectly answered queries do not contribute anything to the gap of N, and the remaining gap is simply that of M^L.

More carefully, the gap generated by M on input x is the sum of the gaps generated for each path p, i.e.,

$$\mathrm{gap}_N(x) = \sum_p (\text{gap generated from path } p) = \sum_p G_p.$$

The sum on the right can be divided into three parts depending on the type of the path p. Let A be the set of all paths p ending in an accepting state of M where all of M's oracle queries along p are answered according to L. Let R be the set of all p ending in a rejecting state of M with all oracle queries answered according to L. Let E consist of the remaining paths, i.e., the ones where some query along p is not answered according to L. We have, for any x,

$$\mathrm{gap}_N(x) \quad = \quad \sum_{p \in A} G_p + \sum_{p \in R} G_p + \sum_{p \in E} G_p$$

$$
\begin{aligned}
&= \sum_{p \in A} 1 + \sum_{p \in R} (-1) + \sum_{p \in E} 0 \\
&= \#M^L(x) - \#\overline{M}^L(x) + 0 \\
&= \mathrm{gap}_{M^L}(x).
\end{aligned}
$$

□

Corollary 5.6

$$Gap\boldsymbol{P}^{SPP} = Gap\boldsymbol{P}.$$

Corollary 5.7 *If $\mathcal{C}$ is any uniformly gap-definable class, then $\mathcal{C}^{SPP} = \mathcal{C}$.*

Proof: Let $\mathcal{C} = \mathrm{Gap}(A, R)$ for some $A, R \subseteq Z$, let $L \in SPP$, and let $S \in \mathcal{C}^L$. By the remarks in section 4, there is an OCM M such that for all $x \in \Sigma^*$,

$$
\begin{aligned}
x \in S &\implies \mathrm{gap}_{M^L}(x) \in A, \\
x \notin S &\implies \mathrm{gap}_{M^L}(x) \in R.
\end{aligned}
$$

By corollary 5.6, we have $\mathrm{gap}_{M^L} = \mathrm{gap}_N$ for some unrelativized CM N. Thus N witnesses that $S \in \mathcal{C}$. □

Corollary 5.8 *SPP is closed under polynomial-time Turing reductions.*

Proof:

$$SPP \subseteq P^{SPP} \subseteq SPP^{SPP} \subseteq SPP$$

by corollary 5.7. Thus $SPP = P^{SPP}$. □

It should be noted that there may be languages not in SPP which are low for some particular gap-definable classes. For example, Köbler, *et al.* [17] showed that Graph Isomorphism (GI) is low for PP and $C_{=}P$ (see below), and it is not known that GI $\in SPP$. As another example, all $\oplus P$ sets are low for $\oplus P$ ([20]), and it is not likely that $SPP = \oplus P$. Also, the class WPP, defined later in this section, is low for PP ([26]), and we don't believe that $SPP = WPP$. The same is true for BPP, defined in the remark following definition 4.6 (see later in this section). Köbler *et al.* [15] showed that BPP is low for PP, and it is unlikely that $BPP \subseteq SPP$.

We now generalize [15] to theorem 5.9 below regarding gaps.

Theorem 5.9

$$SPP = Gap\text{-}\boldsymbol{Few}.$$

Proof: Clearly $SPP \subseteq$ Gap-**Few**. Let L be in Gap-**Few** as witnessed by the CM M, the polynomial time predicate $A(x,k)$, and the polynomial q. Let $\tilde{A}(\langle x,k\rangle)$ be the 0-1-valued function corresponding to the truth value of $A(x,k)$. Finally, let $f(x) \stackrel{df}{=} \tilde{A}(x, \mathrm{gap}_M(x))$. By closure property 6, $f \in \mathrm{Gap}\mathbf{P}$, furthermore, $f(x) = 1$ if $x \in L$, and $f(x) = 0$ otherwise. Thus $L \in SPP$ as witnessed by f. □

Corollary 5.10

$$\textbf{\textit{Few}} \subseteq SPP.$$

Corollary 5.11 ***Few*** *is contained in any reasonable gap-definable class. In particular,*

- ***Few*** $\subseteq C_{=}P$ *([15, 5, 4]).*
- ***Few*** $\subseteq Mod_k P$ *for any* $k \geq 2$ *([7, 5, 4]).*

Proof: Immediate from theorem 5.4 and corollary 5.10. □

Corollary 5.11 also follows from related work of Beigel, Gill, & Hertrampf [5]: **Few** $\subseteq P^{\mathbf{CP}_{Q(\alpha)}}$ for any predicate Q such that $Q(0) = 0$ and $Q(1) = 1$. See [4] for a definition of $P^{\mathbf{CP}_{Q(\alpha)}}$. The next corollary subsumes all the lowness results in [15].

Corollary 5.12 ***Few*** *is low for any uniformly gap-definable class. In particular,* ***Few*** *is low for each of the classes* PP, $C_{=}P$, *and* $\oplus P$ *([15]).*

Proof: Immediate from corollaries 5.7 and 5.10. □

The proof of theorem 5.9 relativizes to show that $SPP^X =$ Gap-**Few**^{X} for any oracle X, thus Gap-**Few** is uniformly gap-definable.

Because of theorem 5.4 and its corollaries, there are several counting classes that are not gap-definable unless certain unlikely complexity theoretic inclusions hold. For example, if BPP is gap-definable, then $UP \subseteq BPP$, and if BPP is uniformly gap-definable, then $BPP^{UP} = BPP$. Of course, these facts about BPP also hold for P, UP, and NP.

The following class is a simple generalization of SPP:

Definition 5.13 *WPP ("wide" PP) is the class of all languages L such that there exists a CM M and a function $f \in FP$ with $0 \notin range(f)$ such that for all x,*

$$\begin{aligned} x \in L &\implies gap_M(x) = f(x), \\ x \notin L &\implies gap_M(x) = 0. \end{aligned}$$

Toda has studied this class, which he names Two, and has a clever proof that WPP is low for PP ([26], see Appendix A). It is clear that $SPP \subseteq WPP \subseteq C_{=}P \cap \text{co-}C_{=}P$, and both inclusions appear to be proper. We may also define a restricted version of WPP, where the function f in the definition can depend only on the *length* of x. We'll call this class $LWPP$.

It appears that $SPP \neq LWPP$ as well. The proof of theorem 5.5 can be modified easily to show that $LWPP$ is low for PP and $C_{=}P$. Köbler, *et al.* [17] show that GI and other related problems are low for these classes by showing that GI $\in LWPP$.

Unfortunately, we cannot modify the proof of theorem 5.5 to show that $LWPP$ is low for WPP or for $LWPP$. The reason lies in the way these classes are relativized. If L is a fixed language in $LWPP$, we say that $A \in WPP^L$ if and only if there exists an everywhere nonzero function $f\colon \Sigma^* \to Z$, computable in polynomial time *relative to* L, and a Gap$\mathbf{P}^L$ function g such that, for all $x \in \Sigma^*$,

$$\begin{aligned} x \in A &\implies g(x) = f(x), \\ x \notin A &\implies g(x) = 0. \end{aligned}$$

The problem is that L can be used in the computation of f. There is no reason to believe that A is then in WPP witnessed by a polynomial-time unrelativized function f. The same goes for the class $LWPP^{LWPP}$. We can, however, adapt the proof of theorem 5.5 to show that $SPP^{LWPP} = LWPP$. Thus $LWPP$ is closed under polynomial-time Turing reductions, and so any problem Turing reducible to GI is in $LWPP$.

At first blush, the classes WPP and $LWPP$ appear not to be gap-definable, since the accepting and rejecting sets cannot be fixed once and for all, but rather must vary depending on the choice of the function f. We show in section 8.1, however, that WPP and $LWPP$ are indeed nonuniformly gap-definable. The nonuniformity appears necessary, because to relativize the definitions of the two classes properly to an oracle X, one must allow f to be a function in FP^X as above, thus the accepting set depends on the oracle.

6 Randomized Counting

One might wonder whether theorem 4.8 holds for a class such as SPP, i.e., is it true that $SPP^{PH} \subseteq \widehat{BP} \cdot SPP$, or even that $PH \subseteq \widehat{BP} \cdot SPP$? Toda & Ogiwara address this question in [28] and conclude that this is probably not the case since the definition of any SPP language includes a promise that the gap of some machine is either 0 or 1, and the proof of theorem 4.8 relies on there being no such promise for a language in GapIn$[Q]$. As further evidence that SPP is not as hard as PH, we now show that there is an oracle relative to which $NP \not\subseteq \widehat{BP} \cdot SPP$. (An observation in [28] implies that $\widehat{BP} \cdot SPP = BP \cdot SPP$ since SPP is closed under majority-tt-reductions.) In fact, the oracle constructed in [3] will do.

Proposition 6.1 *There exists an oracle A such that $NP^A \not\subseteq (\widehat{BP} \cdot SPP)^A$.*

Proof: The following implications all relativize:

$$\begin{aligned} NP \subseteq \widehat{BP} \cdot SPP &\implies P^{NP} \subseteq P^{\widehat{BP} \cdot SPP} \\ &\implies P^{NP} \subseteq P^{BPP^{SPP}} \\ &\implies P^{NP} \subseteq BPP^{SPP} \\ &\implies P^{NP} \subseteq PP^{SPP} \\ &\implies P^{NP} \subseteq PP. \end{aligned}$$

The last implication follows from corollary 5.7. Beigel [3] constructed an oracle relative to which $P^{NP} \not\subseteq PP$. Relative to this same oracle then, $NP \not\subseteq \widehat{BP} \cdot SPP$. $\square$

The most we can say at present is that the statement $PH \subseteq SPP$ is "almost" true. If we let F be the characteristic function of some PH language L in lemma 4.7, we get the following corollary:

Corollary 6.2 (to Lemma 4.7) *For every $L \in PH$ and polynomial e, there exist a function $H \in Gap\mathbf{P}$ and a polynomial s such that, for all x,*

$$|\{w : |w| = s(|x|) \ \ \& \ \ H(\langle x, w\rangle) = \chi_L(x)\}| \geq 2^{s(|x|)}(1 - 2^{-e(|x|)}).$$

We may make the following definition: for any relativizable class $\mathcal{C}$, a language L is in **Almost**($\mathcal{C}$) if and only if

$$\Pr_A[L \in \mathcal{C}^A] = 1.$$

Here, the probability is taken over all oracles A where each $x \in \Sigma^*$ is independently put into A with probability 1/2. The next proposition follows by standard techniques from a relativization of lemma 4.7.

Proposition 6.3 *With respect to a random oracle, PH is low for $Gap\mathbf{P}$, i.e.,*

$$\Pr_R[Gap\mathbf{P}^{PH^R} = Gap\mathbf{P}^R] = 1.$$

Proof: Lemma 4.7 can be relativized to any oracle categorically. That is, given any function F^X computed by some appropriate oracle machine M^X so that $F^X \in \mathrm{Gap}\mathbf{P}^{PH^X}$ for all X uniformly, and given any polynomial e, there exist a polynomial s and an OCM N such that for all x of length n and all oracles A,

$$\left|\left\{w : |w| = s(n) \ \ \& \ \ G^A(\langle x, w\rangle) = F^A(x)\right\}\right| \geq 2^{s(n)}(1 - 2^{-e(n)}),$$

where $G^A \stackrel{df}{=} \mathrm{gap}_{N^A}$. We may also assume that all queries to A in the computation of $G^A(\langle x, w\rangle)$ are bounded by the running time of M. Let r be a polynomial bounding the running time of M. Define, for any oracle A and any x of length n,

$$\hat{G}^A(x) \stackrel{df}{=} G^A(\langle x, w_A\rangle),$$

where

$$w_A \stackrel{df}{=} A(x0^{r(n)+1})A(x0^{r(n)+2}) \cdots A(x0^{r(n)+s(n)}).$$

Clearly there is an OCM $\hat{N}$ such that $\hat{G}^A = \mathrm{gap}_{\hat{N}^A}$ for all A. Fix x of length n. The string w_A is made up of bits consisting of the values of A on arguments which are not used in either the computation of $F^A(x)$ or the computation of $G^A(\langle x, w\rangle)$ for any w of length $s(n)$. Because of this independence, we have

$$\Pr_R[\hat{G}^R(x) \neq F^R(x)] \leq 2^{-e(n)}.$$

Letting c be any natural number and letting $e(n) \stackrel{df}{=} 2n + c + 1$, we have

$$\begin{aligned}
&\Pr_R[\hat{G}^R \neq F^R] \\
&= \Pr_R[(\exists x)\hat{G}^R(x) \neq F^R(x)] \\
&\leq \sum_{n=0}^{\infty} \sum_{x:|x|=n} \Pr_R[\hat{G}^R(x) \neq F^R(x)] \\
&\leq \sum_{n=0}^{\infty} \sum_{x:|x|=n} 2^{-2n-c-1} \\
&= \sum_{n=0}^{\infty} 2^{-n-c-1} \\
&= 2^{-c},
\end{aligned}$$

which in turn implies that

$$\begin{aligned}
1 - 2^{-c} &\leq \Pr_R[F^R = \hat{G}^R] \\
&\leq \Pr_R[F^R \in \mathrm{Gap}\mathbf{P}^R].
\end{aligned}$$

Since $\Pr_R[F^R \in \mathrm{Gap}\mathbf{P}^R]$ is independent of c, we may take c arbitrarily large to get

$$\Pr_R[F^R \in \mathrm{Gap}\mathbf{P}^R] = 1.$$

Since $\mathrm{Gap}\mathbf{P}^{PH^X} = \bigcup_F F^X$ where the F's are computed by only countably many machines M described above, we obtain

$$\Pr_R[\mathrm{Gap}\mathbf{P}^{PH^R} = \mathrm{Gap}\mathbf{P}^R] = 1.$$

□

Corollary 6.4 ***Almost***$(SPP^{PH}) =$ ***Almost***(SPP).

Proof: Let L be any language. We have

$$\begin{aligned}
&L \in SPP^{PH^A} \text{ for a.e. } A \\
&\iff \chi_L \in \mathrm{Gap}\mathbf{P}^{PH^A} \text{ for a.e. } A \\
&\iff \chi_L \in \mathrm{Gap}\mathbf{P}^A \text{ for a.e. } A
\end{aligned}$$

by proposition 6.3. □

Subsequent research [8] implies that $\mathbf{Almost}(SPP)$ is also nonuniformly gap-definable.

For the next corollary, a natural way to relativize $\mathbf{Almost}(\mathcal{C})$ to an oracle A is to say that $L \in (\mathbf{Almost}(\mathcal{C}))^A$ if and only if $\Pr_R[L \in \mathcal{C}^{R \oplus A}] = 1$. With this definition, $\mathbf{Almost}(P)$ relativizes the same way as BPP with the usual machine-based definition.

Corollary 6.5 *PH is low for* ***Almost***(SPP).

Proof: It can be easily shown that $(\mathbf{Almost}(SPP))^{PH}$ is a subclass of $\mathbf{Almost}(SPP^{PH})$ and a superclass of $\mathbf{Almost}(SPP)$. The corollary follows from the equality of the two latter classes. □

Corollary 6.6 *$PH \subseteq$ **Almost**(SPP).*

Corollary 6.7 *With respect to a random oracle, $PH \subseteq SPP$, in fact, PH is low for SPP.*

Proof: For a.e. A we have

$$PH^A \subseteq SPP^{PH^A} = \{L \mid \chi_L \in \mathrm{Gap}\mathbf{P}^{PH^A}\} = \{L \mid \chi_L \in \mathrm{Gap}\mathbf{P}^A\} = SPP^A.$$

□

7 Closure Properties of GapP

It is natural to ask if, in addition to the closure properties enumerated in section 3, Gap**P** has any other closure properties. For example, is Gap**P** closed under unrestricted composition with itself? Is Gap**P** closed under left composition with functions in FP? We know from section 3 that Gap**P** is closed under left composition with the "bounded" delta function δ_k^B. Is Gap**P** also closed under left composition with the "unbounded" delta function

$$\delta(x) \stackrel{df}{=} \begin{cases} 1 & \text{if } x = 0, \\ 0 & \text{otherwise,} \end{cases}$$

defined for all $x \in Z$?

The answer to all of these questions is no, unless certain unlikely complexity theoretic identities hold. Ogiwara & Hemachandra [19] have studied closure questions such as these in detail, primarily for the class #**P**. They and Gupta [12] also address closure properties of Gap**P**. We obtained theorem 7.1 independently of their work. See [19] for a nice, unified treatment of these questions.

In theorem 7.1 below, if $P(\vec{x})$ is any predicate, we define the function

$$[P(\vec{x})] \stackrel{df}{=} \begin{cases} 1 & \text{if } P(\vec{x}) \text{ is true,} \\ 0 & \text{otherwise.} \end{cases}$$

For example, $[x = 0] = \delta(x)$ as defined above. Also recall that we have identified Σ^* with Z for computational purposes.

Theorem 7.1 *The following are equivalent:*

1. $\{\delta\} \circ Gap\boldsymbol{P} \subseteq Gap\boldsymbol{P}$.
2. $\{\lambda xy.[x = y]\} \circ (Gap\boldsymbol{P} \times Gap\boldsymbol{P}) \subseteq Gap\boldsymbol{P}$.

3. $Gap\boldsymbol{P} \circ Gap\boldsymbol{P} \subseteq Gap\boldsymbol{P}$.
4. $\{\lambda x.[0 < x]\} \circ Gap\boldsymbol{P} \subseteq Gap\boldsymbol{P}$.
5. $\{\lambda xy.[x < y]\} \circ (Gap\boldsymbol{P} \times Gap\boldsymbol{P}) \subseteq Gap\boldsymbol{P}$.
6. $SPP = PP$.
7. $SPP = C_{=}P$.
8. $(\lambda xy.[x = y]) \circ (\#\boldsymbol{P} \times \#\boldsymbol{P}) \subseteq Gap\boldsymbol{P}$.
9. $FP \circ Gap\boldsymbol{P} \subseteq Gap\boldsymbol{P}$.

Proof Sketch: In what follows, f and g are arbitrary functions in Gap$\mathbf{P}$.

1 $\Longrightarrow$ 2: $[f(x) = g(x)] = \delta(f(x) - g(x))$.

2 $\Longrightarrow$ 3: $g(f(x)) = \sum_{y \in Z} g(y) \cdot [y = f(x)]$.

3 $\Longrightarrow$ 1: Follows from the fact that $\delta \in$ Gap$\mathbf{P}$.

2 $\Longrightarrow$ 4: $[0 < f(x)] = \sum_{y>0} [y = f(x)]$.

4 $\Longrightarrow$ 5: $[f(x) < g(x)] = [0 < g(x) - f(x)]$.

5 $\Longrightarrow$ 1: $\delta(f(x)) = 1 - [0 < f(x)] - [f(x) < 0]$.

4 $\Longrightarrow$ 6: If $L \in PP$ witnessed by $f \in$ Gap$\mathbf{P}$, then $L \in SPP$ witnessed by $[0 < f(x)]$.

6 $\Longrightarrow$ 7: Follows from the fact that $C_{=}P \subseteq PP$.

7 $\Longrightarrow$ 1: The $C_{=}P$ set $\{x \mid f(x) = 0\}$ is in SPP witnessed by the function $[f(x) = 0] = \delta(f(x))$. Hence $\delta \circ f \in$ Gap$\mathbf{P}$.

2 $\Longrightarrow$ 8: Follows from the fact that $\#\mathbf{P} \subseteq$ Gap$\mathbf{P}$.

8 $\Longrightarrow$ 1: If $f = f_1 - f_2$ where $f_1, f_2 \in \#\mathbf{P}$, then $[f(x) = 0] = [f_1(x) = f_2(x)]$.

3 $\Longrightarrow$ 9: Follows from the fact that $FP \subseteq$ Gap$\mathbf{P}$.

9 $\Longrightarrow$ 1: Follows from the fact that $\delta \in FP$.

□

Ogiwara & Hemachandra [19] and independently Gupta [12] show further that statements 6 and 7 are equivalent to the polynomial counting hierarchy collapsing to SPP (see either source for definitions).

8 Structure of the Gap-Definable Classes

In this section we examine the collection $\mathcal{G}$ of all gap-definable classes, partially ordered by inclusion. We show that any countable class of languages is contained in a unique minimum gap-definable class (its 'gap-closure'). From this we show that $\mathcal{G}$ is closed under intersection, and further that $\mathcal{G}$ is a lattice under inclusion, i.e., any two gap-definable classes have a gap-definable least-upper-bound and a gap-definable greatest-lower-bound.

In section 8.1 we will define a gap-closure operator, GapCl, which maps countable classes of languages to other countable classes of languages. There we will show that GapCl satisfies the following axioms for any countable classes $\mathcal{D}$ and $\mathcal{E}$:

1. GapCl($\mathcal{D}$) is gap-definable.
2. $\mathcal{D} \subseteq$ GapCl($\mathcal{D}$).
3. If $\mathcal{D}$ is gap-definable, then GapCl($\mathcal{D}$) = $\mathcal{D}$.
4. $\mathcal{D} \subseteq \mathcal{E} \Longrightarrow$ GapCl($\mathcal{D}$) $\subseteq$ GapCl($\mathcal{E}$) (GapCl is *monotone*).

In order to prove these results, we must build accepting and rejecting sets that are not recursive (see section 8.1). (Despite this fact, the complexity of GapCl($\mathcal{C}$) is not a great deal higher that that of $\mathcal{C}$; in particular, if $\mathcal{C}$ consists only of recursive sets, than so does GapCl($\mathcal{C}$).) We use the same technique in section 8.1 to show that the classes WPP and $LWPP$ are (nonuniformly) gap-definable.

We can use GapCl to get structural information about the gap-definable classes, summarized in the following theorem:

Theorem 8.1

1. *GapCl(GapCl($\mathcal{D}$)) = GapCl($\mathcal{D}$) (GapCl is idempotent).*
2. *If $\mathcal{D}$ is a countable class, there is a unique minimum gap-definable class which contains $\mathcal{D}$.*
3. *Any countable collection of gap-definable classes has a gap-definable least-upper-bound (under inclusion).*
4. *The intersection of an arbitrary collection of gap-definable classes is gap-definable.*
5. *The gap-definable classes form a lattice (under inclusion).*

Proof:

1. Follows immediately from axioms 1 and 3.
2. Clearly, $\mathcal{D} \subseteq$ GapCl($\mathcal{D}$) by axiom 2, and GapCl($\mathcal{D}$) is gap-definable by axiom 1. If $\mathcal{E}$ is any gap-definable class containing $\mathcal{D}$, then by axioms 3 and 4, GapCl($\mathcal{D}$) $\subseteq$ GapCl($\mathcal{E}$) = $\mathcal{E}$. Therefore, GapCl($\mathcal{D}$) is the least gap-definable class containing $\mathcal{D}$.
3. Let $\{\mathcal{D}_i\}_{i \in \Sigma^*}$ be a collection of gap-definable classes. All the $\mathcal{D}_i$ are countable, so $\mathcal{D} = \bigcup_{i \in \Sigma^*} \mathcal{D}_i$ is countable, and GapCl($\mathcal{D}$) is the required least-upper-bound.
4. Let $\{\mathcal{D}_i\}_{i \in I}$ be an arbitrary collection of gap-definable classes, and let $\mathcal{D} \stackrel{df}{=} \bigcap_{i \in I} \mathcal{D}_i$. For all $i \in I$, we have $\mathcal{D} \subseteq \mathcal{D}_i$, so by axioms 3 and 4, we have GapCl($\mathcal{D}$) $\subseteq$ GapCl($\mathcal{D}_i$) = $\mathcal{D}_i$. Thus GapCl($\mathcal{D}$) $\subseteq \mathcal{D}$, and so GapCl($\mathcal{D}$) = $\mathcal{D}$ by axiom 2. Thus $\mathcal{D}$ is gap-definable by axiom 1.
5. This follows immediately from the previous two claims. The least-upper-bound of two classes is the gap-closure of their union, and the greatest-lower-bound is their intersection.

□

The operator GapCl satisfies some other nice properties besides axioms 1–4. For example, if $\mathcal{D}$ is closed downward under ptime m-reductions, then GapCl($\mathcal{D}$) is similarly closed (theorem 8.5 in section 8.1). Thus we know immediately that GapCl(NP) is closed under ptime m-reductions, for instance.

8.1 The Gap-Closure Operator, GapCl

Let W be an immune set, i.e., W has the following two properties:

1. W is infinite.
2. W has no infinite recursively enumerable subsets.

It is well-known that such sets exist (see [21, 24]); for example, we can take

$$W \stackrel{df}{=} \{x \in \Sigma^* \mid K(x) \geq |x|/2\},$$

where $K(x)$ is the Kolmogorov complexity of x with respect to some fixed universal DTM (see [18]). We let $W = \{w_1, w_2, w_3, \ldots\}$, where $w_1 < w_2 < w_3 < \ldots$.

Now suppose $\mathcal{D} = \{L_1, L_2, L_3, \ldots\}$ is a countable collection of languages. Define

$$A_{\mathcal{D}} \stackrel{df}{=} \{(x, w_i) \mid x \in L_i\}$$

and

$$R_{\mathcal{D}} \stackrel{df}{=} \{(x, w_i) \mid x \notin L_i\},$$

and define $\mathrm{GapCl}(\mathcal{D}) \stackrel{df}{=} \mathrm{Gap}(A_{\mathcal{D}}, R_{\mathcal{D}})$.

Fact 8.2 *If M is an $(A_{\mathcal{D}}, R_{\mathcal{D}})$-proper CM, then $range(gap_M)$ is a finite subset of W.*

Proof: Clearly, $range(\mathrm{gap}_M) \subseteq W$ by the definitions of $A_{\mathcal{D}}$ and $R_{\mathcal{D}}$. Since gap_M is a computable function, its range is recursively enumerable and hence finite by the second property of W. □

Theorem 8.3 *The operator GapCl satisfies axioms 1–4 above.*

Proof:

1. $\mathrm{GapCl}(\mathcal{D})$ is gap-definable by definition.
2. If $L \in \mathcal{D} = \{L_1, L_2, \ldots\}$, then $L = L_i$ for some $i \geq 1$. Any CM M that generates a constant gap of w_i is $(A_{\mathcal{D}}, R_{\mathcal{D}})$-proper, and $L = L_{A_{\mathcal{D}}, R_{\mathcal{D}}}(M)$. Thus $L \in \mathrm{GapCl}(\mathcal{D})$.
3. Suppose $\mathcal{D} = \{L_1, L_2, \ldots\} = \mathrm{Gap}(A, R)$ for some A and R, and let $M_1, M_2, \ldots$ be (A, R)-proper CM's such that $L_i = L_{A,R}(M_i)$ for all $i \geq 1$. Suppose L is a language in $\mathrm{GapCl}(\mathcal{D})$. We have
$$L = L_{A_{\mathcal{D}}, R_{\mathcal{D}}}(M)$$
for some $(A_{\mathcal{D}}, R_{\mathcal{D}})$-proper CM M. By fact 8.2 above, there is some $k \geq 1$ such that $range(\mathrm{gap}_M) \subseteq \{w_1, \ldots, w_k\}$. Consider a CM N such that
$$\mathrm{gap}_N(x) = \sum_{i=1}^{k} \delta_{w_i}^{w_k}(\mathrm{gap}_M(x))\mathrm{gap}_{M_i}(x),$$

where the $\delta_{w_i}^{w_k}$ are the delta functions defined in section 3. Such an N clearly exists. Given an input x, suppose $\mathrm{gap}_M(x) = w_{i_0}$ for some $1 \leq i_0 \leq k$. Then $\mathrm{gap}_N(x) = \mathrm{gap}_{M_{i_0}}(x)$. Furthermore,

$$\begin{aligned} x \in L &\implies (x, \mathrm{gap}_M(x)) \in A_{\mathcal{D}} \\ &\implies (x, w_{i_0}) \in A_{\mathcal{D}} \\ &\implies x \in L_{i_0} \\ &\implies (x, \mathrm{gap}_{M_{i_0}}(x)) \in A \\ &\implies (x, \mathrm{gap}_N(x)) \in A. \end{aligned}$$

Similarly, $x \notin L \implies (x, \mathrm{gap}_N(x)) \in R$. Thus N is (A, R)-proper and $L = L_{A,R}(N)$, so $L \in \mathrm{Gap}(A, R) = \mathcal{D}$.

4. Suppose $\mathcal{D} = \{L_1, L_2, \ldots\}$ and $\mathcal{E} = \{L'_1, L'_2, \ldots\}$ are countable language classes and $\mathcal{D} \subseteq \mathcal{E}$. Assume $L = L_{A_{\mathcal{D}}, R_{\mathcal{D}}}(M)$ for some $(A_{\mathcal{D}}, R_{\mathcal{D}})$-proper CM M. We show that $L \in \mathrm{GapCl}(\mathcal{E})$. As before, there exists a k such that $range(\mathrm{gap}_M) \subseteq \{w_1, \ldots, w_k\}$. Since $\mathcal{D} \subseteq \mathcal{E}$, there exist $n_1, \ldots, n_k$ such that $L_i = L'_{n_i}$ for $1 \leq i \leq k$. Let N be a CM such that

$$\mathrm{gap}_N(x) = \sum_{i=1}^{k} \delta_{w_i}^{w_k}(\mathrm{gap}_M(x)) w_{n_i}.$$

By an argument similar to the one above, we have that N is $(A_{\mathcal{E}}, R_{\mathcal{E}})$-proper and $L = L_{A_{\mathcal{E}}, R_{\mathcal{E}}}(N)$. Thus $L \in \mathrm{GapCl}(\mathcal{E})$.

□

The preceding proof relativizes to any oracle, but only nonuniformly. This is because given an oracle, we must choose W to be immune *relative to that oracle*. Thus the accepting and rejecting sets that we construct must depend on the oracle. This means that the gap-closure of a class is not necessarily uniformly gap-definable.

We now use the same technique to show that WPP and $LWPP$ are nonuniformly gap-definable.

Proposition 8.4 *The classes WPP and $LWPP$ are (nonuniformly) gap-definable.*

Proof: We show that $WPP = \mathrm{GapCl}(WPP)$. The proof for $LWPP$ is similar. Let $WPP = \{L_1, L_2, \ldots\}$ such that for all $i > 0$ and $x \in \Sigma^*$,

$$\begin{aligned} x \in L_i &\implies \mathrm{gap}_{M_i}(x) = f_i(x), \\ x \notin L_i &\implies \mathrm{gap}_{M_i}(x) = 0, \end{aligned}$$

for CM's $M_1, M_2, \ldots$ and FP functions $f_1, f_2, \ldots$. As in the proof of theorem 8.3, let $L \stackrel{df}{=} L_{A_{\mathbf{WPP}}, R_{\mathbf{WPP}}}(M)$ for some CM M, and let k be as before. Define $F \in FP$ by

$$F(x) \stackrel{df}{=} \prod_{j=1}^{k} f_j(x).$$

Let N be a CM such that

$$\mathrm{gap}_N(x) = \sum_{i=1}^{k} \left[\delta_{w_i}^{w_k}(\mathrm{gap}_M(x)) \cdot \mathrm{gap}_{M_i}(x) \cdot \prod_{1 \le j \le k \ \& \ j \neq i} f_j(x) \right].$$

By arguments similar to theorem 8.3, $L \in WPP$ as witnessed by the CM N and FP function F. $\square$

What closure properties of a class $\mathcal{D}$ are inherited by GapCl($\mathcal{D}$)? We can show the following:

Theorem 8.5 *Let $\mathcal{D}$ be a countable class of languages.*

1. *If $\mathcal{D}$ is closed downward under ptime m-reductions, then GapCl($\mathcal{D}$) is closed downward under ptime m-reductions.*
2. *If $\mathcal{D}$ is closed under complements, then GapCl($\mathcal{D}$) is closed under complements.*
3. *If $\mathcal{D}$ is closed downward under ptime 1-tt-reductions, then GapCl($\mathcal{D}$) is closed downward under ptime 1-tt-reductions.*

Proof: We only prove the first statement. The other two are similar. Suppose $\mathcal{D}$, as above, is closed under ptime m-reductions, $L \in \mathrm{GapCl}(\mathcal{D})$, and f is any function in FP. We must show that $f^{-1}[L] \in \mathrm{GapCl}(\mathcal{D})$. Let $L = L_{A_\mathcal{D}, R_\mathcal{D}}(M)$ and k be as before. Since $\mathcal{D}$ is closed under ptime m-reductions, there exist $n_1, \ldots, n_k$ such that $L_{n_i} = f^{-1}[L_i]$ for $1 \le i \le k$. Let N be a CM such that

$$\mathrm{gap}_N(x) = \sum_{i=1}^{k} \delta_{w_i}^{w_k}(\mathrm{gap}_M(f(x)))w_{n_i}.$$

By arguments similar to those for theorem 8.3, we get $f^{-1}[L] = L_{A_\mathcal{D}, R_\mathcal{D}}(N)$. $\square$

Subsequent research [8] has shown that GapCl also preserves closure under union, intersection, join, and finite difference. Moreover, the definition of GapCl and gap-definability can be greatly simplified for classes closed under union and intersection.

9 Alternative Notions of Gap-Definability

There are three natural conditions one can add to the definition of gap-definability:

1. The accepting and rejecting sets A and R must partition $\Sigma^* \times Z$, i.e., $A \cup R = \Sigma^* \times Z$.
2. The criteria for acceptance/rejection must be independent of the input, i.e., $A = \Sigma^* \times A'$ and $R = \Sigma^* \times R'$ for disjoint sets $A', R' \subseteq Z$.
3. The sets A and R must be of low complexity.

The second and third conditions both lead to proper restrictions of the notion of gap-definability, even when one considers only reasonable gap-definable classes (exercise). This is not known for the first condition, however (see section 10). Each restriction has its own advantages: the first restriction guarantees that all CM's are (A, R)-proper, and hence the resulting classes are all recursively presentable, at least relative to A and R; the second restriction guarantees that the resulting classes are closed under joins, finite differences (provided the classes are reasonable), and polynomial-time m-reductions; the third restriction ensures that the resulting classes are of reasonably low complexity. The first two conditions taken together yield the classes GapIn[Q] (see definition 4.5) considered by Toda & Ogiwara [28], which we will call *nice* classes. As well as having all the properties mentioned above, nice classes also have complete sets (under polynomial time m-reductions). Despite these restrictions, all the well-known gap-definable classes—PP, $C_{=}P$, and $\mathrm{Mod}_k P$—are nice, and have simple acceptance/rejection criteria.

A disadvantage of these restrictions is that the theorems of section 8 apparently do not hold for any of them. At present, we see no way of getting around the use of (nonrecursive) immune sets to verify the properties of GapCl. It also appears that the intersection of two nice classes is most likely not nice, in fact, we have the following proposition:

Proposition 9.1 *If $Q_1, Q_2 \subseteq Z$ are chosen independently at random, then*

$$GapIn[Q_1] \cap GapIn[Q_2] = SPP$$

with probability 1.

Proposition 9.1 follows immediately from lemmas 9.3 and 9.4, below, with a simple application of Fubini's Theorem. Recall that we identify Σ^* with Z.

Definition 9.2 *Fix an oracle $A \subseteq \Sigma^*$. A set $S \subseteq Z$ is immune relative to A if*

1. *S is infinite, and*
2. *every A-r.e. subset of S is finite.*

The set S is bi-immune relative to A if both S and $Z - S$ are immune relative to A.

Lemma 9.3 *For every set $A \subseteq \Sigma^*$,*

$$\Pr_S[S \text{ is bi-immune relative to } A] = 1.$$

Proof: Fix an A-r.e. set $W \subseteq Z$ and let $S \subseteq Z$ be chosen at random. Since there are only countably many finite and cofinite sets, S and $Z - S$ are both infinite with probability 1. Clearly,

$$\Pr_S[W \subseteq S \quad \vee \quad W \subseteq Z - S] = 2^{-|W|+1}$$

if W is finite, and $\Pr_S[W \subseteq S \quad \vee \quad W \subseteq Z - S] = 0$ if W is infinite. Since there are only countably many infinite A-r.e. sets, we have

$$\Pr_S[S \text{ is not bi-immune relative to } A]$$

$$
\begin{aligned}
&= \Pr_S[(\exists W \text{ infinite } A\text{-r.e.})\ W \subseteq S \ \ \vee \ \ W \subseteq Z - S] \\
&\leq \sum_{W \text{ inf } A\text{-r.e.}} \Pr_S[W \subseteq S \ \ \vee \ \ W \subseteq Z - S] \\
&= 0,
\end{aligned}
$$

so the lemma holds. □

Lemma 9.4 *For every $Q_1, Q_2 \subseteq Z$, if $Q_1 \notin \{\emptyset, Z\}$ and Q_2 is bi-immune relative to Q_1, then*

$$GapIn[Q_1] \cap GapIn[Q_2] = SPP.$$

Proof: Clearly, $SPP \subseteq \text{GapIn}[Q_1] \cap \text{GapIn}[Q_2]$ by theorem 5.4 and the fact that $\text{GapIn}[Q_1]$ and $\text{GapIn}[Q_2]$ are both reasonable gap-definable classes.

Let $L \subseteq \Sigma^*$ be a language in $\text{GapIn}[Q_1] \cap \text{GapIn}[Q_2]$. There exist $f, g \in \text{Gap}\mathbf{P}$ such that for all $x \in \Sigma^*$,

$$x \in L \iff f(x) \in Q_1 \iff g(x) \in Q_2.$$

The first biconditional implies that L is recursive in Q_1, which in turn implies that both $g[L]$ and $g[\overline{L}]$ are Q_1-r.e. But since $g[L] \subseteq Q_2$ and $g[\overline{L}] \subseteq Z - Q_2$, both sets are finite, and thus g has finite range. It is then clear that $L \in \text{Gap-}\mathbf{Few}$, and so by theorem 5.9, $L \in SPP$. □

10 Open Questions

There are several interesting questions regarding gap-definable classes.

- Because WPP and $LWPP$ are only nonuniformly gap-definable, it is not at all clear that $WPP^{SPP} = WPP$. The best we are able to show is that $WPP^{SPP} \subseteq C_=P \cap \text{co-}C_=P$.
- Is WPP uniformly gap-definable?
- Does $WPP = SPP$, or even $LWPP = SPP$?
- Is WPP closed under polynomial-time Turing reductions?
- Is there a Gap$\mathbf{P}$ function Turing equivalent to an NP-complete language?
- How does BPP relate to the gap-definable classes? In particular, is it the case that $\text{GapCl}(BPP) = PP$?
- Does GapCl preserve closure under less restricted reductions, e.g., ptime tt-reductions?
- Is there a reasonable gap-definable class which does not satisfy the first condition in section 9? Is SPP such a class?
- Are there two nice classes whose intersection is known not to be nice?
- Are there other interesting gap-definable classes not previously studied?

Acknowledgments

We would like to thank Seinosuke Toda, Richard Beigel, Nick Reingold, and Lane Hemachandra for many helpful discussions and suggestions. We would also like to thank Krzysztof Lorys for pointing out an error in an earlier version of the paper.

References

[1] E. W. Allender. The complexity of sparse sets in P. In *Structure in Complexity Theory*, volume 223 of *Lecture Notes in Computer Science*, pages 1–11. Springer-Verlag, June 1986.

[2] L. Babai and L. Fortnow. Arithmetization: A new method in structural complexity theory. *Computational Complexity*, 1(1):41–67, 1991. A previous version appeared in *Proceedings of the 31st annual IEEE Symposium on Foundations of Computer Science*, pages 26–34, 1990.

[3] R. Beigel. Perceptrons, PP and the polynomial hierarchy. In *Proceedings of the 7th Structure in Complexity Theory Conference*, pages 14–19, 1992.

[4] R. Beigel and J. Gill. Counting classes: Thresholds, parity, mods, and fewness. Unpublished manuscript, October 1990.

[5] R. Beigel, J. Gill, and U. Hertrampf. Counting classes: Thresholds, parity, mods, and fewness. In *Proceedings of the Seventh Annual Symposium on Theoretical Aspects of Computer Science*, volume 415 of *Lecture Notes in Computer Science*, pages 49–57. Springer-Verlag, 1990.

[6] R. Beigel, N. Reingold, and D. Spielman. PP is closed under intersection. In *Proceedings of the 23rd annual ACM Symposium on Theory of Computing*, pages 1–9, 1991.

[7] J. Cai and L. Hemachandra. On the power of parity polynomial time. *Mathematical Systems Theory*, 23(2):95–106, 1990.

[8] S. Fenner, L. Fortnow, and L. Li. Gap-definability as a closure property. Unpublished, 1992.

[9] J. Gill. Computational complexity of probabilistic complexity classes. *SIAM Journal on Computing*, 6:675–695, 1977.

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

[11] R. L. Graham, D. E. Knuth, and O. Patashnik. *Concrete Mathematics.* Addison-Wesley Publishing House, 1989.

[12] S. Gupta. The power of witness reduction. In *Proceedings of the 6th Annual IEEE Structure in Complexity Theory Conference*, pages 43–59, 1991.

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

[14] J. Köbler. *Strukturelle Komplexität von Anzahlproblemen.* PhD thesis, Universität Stuttgart, 1989. Page 62.

[15] J. Köbler, U. Schöning, S. Toda, and J. Torán. Turing machines with few accepting computations and low sets for PP. *Journal of Computer and System Sciences*, 44(2):272–286, 1992.

[16] J. Köbler, U. Schöning, and J. Torán. On counting and approximation. *Acta Informatica*, 26:363–379, 1989.

[17] J. Köbler, U. Schöning, and J. Torán. Graph Isomorphism is low for PP. In *Proceedings of the Ninth Annual Symposium on Theoretical Aspects of Computer Science*, volume 577 of *Lecture Notes in Computer Science*, pages 401–411. Springer-Verlag, 1992.

[18] M. Li and P. M. B. Vitányi. Applications of Kolmogorov complexity in the theory of computation. In A. L. Selman, editor, *Complexity Theory Retrospective*, chapter 6, pages 147–203. Springer-Verlag, 1990.

[19] M. Ogiwara and L. A. Hemachandra. A complexity theory of feasible closure properties. In *Proceedings of the 6th Annual IEEE Structure in Complexity Theory Conference*, pages 16–29, 1991.

[20] C. H. Papadimitriou and S. K. Zachos. *Two Remarks on the Power of Counting*, pages 269–276. Lecture Notes in Computer Science 145. Springer-Verlag, 1983.

[21] H. Rogers. *Theory of Recursive Functions and Effective Computability*. McGraw-Hill, 1967. Reprinted. MIT Press. 1987.

[22] U. Schöning. Probabilistic complexity classes and lowness. *Journal of Computer and System Sciences*, 39:84–100, 1988. Also appeared in Proceedings of the 2nd Annual IEEE Structure in Complexity Theory Conference, pages 2–8, 1987.

[23] J. Simon. *On Some Central Problems in Computational Complexity*. PhD thesis, Cornell University, Ithaca, N. Y., January 1975. Available as Cornell Department of Computer Science Technical Report TR75-224.

[24] R. Soare. *Recursively Enumerable Sets and Degrees*. Springer-Verlag, 1987.

[25] L. Stockmeyer. The polynomial-time hierarchy. *Theoretical Computer Science*, 3:1–22, 1977.

[26] S. Toda, 1990. Private communication.

[27] S. Toda. PP is as hard as the polynomial-time hierarchy. *SIAM Journal on Computing*, 20(5):865–877, 1991.

[28] S. Toda and M. Ogiwara. Counting classes are at least as hard as the polynomial-time hierarchy. *SIAM Journal on Computing*, 21(2):316–328, 1992.

[29] L. Valiant. The complexity of computing the permanent. *Theoretical Computer Science*, pages 189–201, 1979.

[30] K. Wagner. The complexity of combinatorial problems with succinct input representation. *Acta Informatica*, 23:325–356, 1986.

A WPP

We reproduce here Toda's result [26] mentioned in section 5.

Theorem A.1 (Toda) $PP^{WPP} = PP$.

The theorem follows immediately from the following three lemmas:

Lemma A.2 $PP^{WPP} = C \cdot P_{ctt}^{WPP}$, *where* $C\cdot$ *is Wagner's counting operator [30], and* P_{ctt}^{WPP} *is the closure of* WPP *under conjunctive tt-reductions.* □

Lemma A.3 $P_{ctt}^{WPP} = WPP$.

Proof Sketch: Suppose $L \leq_{ctt}^{p} S$ via the function $r(x) \stackrel{df}{=} \langle q_1, \ldots, q_m \rangle$, and $S \in WPP$ witnessed by the FP function f and Gap**P** function g. Then $L \in WPP$ witnessed by the FP function $h(x) \stackrel{df}{=} \prod_{q \in r(x)} f(q)$ and the Gap**P** function $k(x) \stackrel{df}{=} \prod_{q \in r(x)} g(q)$. □

Lemma A.4 $C \cdot WPP = PP$.

Proof Sketch: Obviously $PP = C \cdot P \subseteq C \cdot WPP$. Conversely, let L be in $C \cdot WPP$. Then there exist $A \in WPP$ and a polynomial p such that for all x of length n,

$$x \in L \iff \left| \{ w \in \{0,1\}^{p(n)} \mid x\#w \in A \} \right| > 2^{p(n)-1}.$$

Moreover, there exist functions $F \in$ Gap**P** and $f \in FP$ such that for all y, $f(y) \neq 0$ and

1. $F(y)$ is either 0 or $f(y)$, and
2. $y \in A \iff F(y) = f(y)$.

We can assume without loss of generality that $f(y) > 0$ for all y. Let q be a polynomial satisfying $q(n) > p(n)$ for all n, and $2^{q(n)} > f(x\#w)$ for all x of length n and w of length $p(n)$. Then, define a function G as follows: for all x of length n,

$$G(x) \stackrel{df}{=} \sum_{w \in \{0,1\}^{p(n)}} \left\lceil 2^{2q(n)} / f(x\#w) \right\rceil \cdot F(x\#w).$$

Obviously, $G \in$ Gap**P**. It is now easy to show that for all x of length n,

$$x \in L \iff G(x) \geq (2^{p(n)-1} + 1) \cdot 2^{2q(n)}.$$

Therefore $L \in PP$. □