On the Power of One Bit of a #P Function

Kenneth W. Regan*
SUNY/Buffalo
regan@cs.buffalo.edu

Thomas Schwentick
University of Mainz
tick@uaimzti.mathematik.uni-mainz.de

June 5, 1992

Abstract

We introduce the class MP of languages L which can be solved in polynomial time with an oracle for one selected bit of the value $f(y)$ of a #P function f on a selected argument y. This extends the much-studied language classes $\oplus$P and PP, which correspond to the power of the least and most significant bits, respectively. We show that MP is captured by the power of the middle bit; namely: a language L is in MP iff for some #P function f' and all x, $x \in L \iff$ the middle bit of $f'(x)$ in binary notation is a '1'. Also, S. Toda's proof [Tod89, Tod91] that the polynomial hierarchy (PH) is contained in $\mathrm{P}^{\#\mathrm{P}}$ actually gives: $\mathrm{PH} \subseteq \mathrm{BP}[\oplus\mathrm{P}] \subseteq \mathrm{C}\oplus\mathrm{P} \subseteq \mathrm{MP}$.

The class MP has complete problems, and is closed under complements and under polynomial-time many-one reducibility. We show that MP is closed under intersection iff, for any fixed $k > 0$, k bits of a #P function are no more powerful than one. Moreover, if there is a polynomial-time construction for the closure under intersection, then $O(\log n)$-many bits have the same power as one. We examine the subclass AmpMP of languages whose MP representations can be "amplified," showing that $\mathrm{BP}[\oplus\mathrm{P}] \subseteq \mathrm{AmpMP}$, and that for any $\leq_m^p$-closed subclass $\mathcal{C}$ of AmpMP, $\mathrm{MP}^{\mathcal{C}} = \mathrm{MP}$. Hence $\mathrm{BP}[\oplus\mathrm{P}]$ is low for MP, and if $C_=P \subseteq \mathrm{AmpMP}$, then $\mathrm{PP}^{\mathrm{PP}} = \mathrm{MP}$. Finally, our work leads to a purely mathematical question about the size of integer-valued polynomials $p(x, y)$ which satisfy certain congruence relations, one which also matters to the theory of bounded-depth circuits.

Keywords Computational complexity, theory of computation, polynomial-time hierarchy, randomness, counting classes, oracles.

*The first author was supported in part by NSF Grant CCR-9011248

1. Overview

This paper examines the hardness of computing selected bits of the value of a function f on a given argument y. We suppose that the values $f(y)$ are encoded as binary numbers, possibly with leading zeroes, with the least significant bit written rightmost but indexed first. This subject has been investigated extensively for functions f of cryptographic importance, in particular, for the discrete log and inverse RSA functions [ACGS84, Cho86, SS90]. We raise it for functions f belonging to the class #P, whose typical (complete) members are much harder than the discrete log or RSA^{-1}, unless NP = coNP.

We frame our study in the following terms: if the kth bit of $f(y)$ could be computed in polynomial time (where k and the argument y depend on a given input x), what other problems or languages L could be solved in polynomial time? Our first main result shows that if L reduces to the selected-bit problem for a #P function f then we can find another #P function g such that for all x, $x \in L \iff$ the middle bit of $g(x)$ is a '1'. This justifies the name MP or "middle-bit P" for this class of languages.

The second reason for the importance of MP is that S. Toda's celebrated proof [Tod89, Tod91] that the polynomial hierarchy PH is contained in $\mathrm{P}^{\#\mathrm{P}}$ actually shows the following (where we have written $\mathrm{C}\oplus\mathrm{P}$ for Toda's $\mathrm{P} \cdot \oplus\mathrm{P}$):

Theorem 1. *[after Toda].* $\mathrm{PH} \subseteq \mathrm{C}\oplus\mathrm{P} \subseteq \mathrm{MP}$.

Clearly $\mathrm{MP} \subseteq \mathrm{P}^{\#\mathrm{P}[1]}$, where the '[1]' means that only one oracle query is needed. Thus MP forms a natural complexity class between the polynomial hierarchy and PSPACE. We give an even tighter form of Theorem 1, one stating that for any language D in $\mathrm{C}\oplus\mathrm{P}$ and polynomial s, there is a #P function f such that for all x, there are $s(|x|)$-many 0s immediately to the left of the "decision bit" $D(x)$ in the binary representation of $f(x)$. In Section 4 we define AmpMP to be the class of languages which admit this kind of "amplification" on both sides of the decision bit. We show that $\mathrm{BP}[\oplus\mathrm{P}] \subseteq \mathrm{AmpMP}$, and that for any class $\mathcal{C} \subseteq \mathrm{AmpMP}$ which is closed downward under polynomial-time many-one reductions, such as $\mathrm{BP}[\oplus\mathrm{P}]$ itself, $\mathrm{MP}^{\mathcal{C}} = \mathrm{MP}$. If AmpMP = MP, or even if $\mathrm{PP} \subseteq \mathrm{AmpMP}$ or $C_{=}P \subseteq \mathrm{AmpMP}$, then the *counting hierarchy* [Wag86] collapses with $\mathrm{PP}^{\mathrm{PP}} = \mathrm{P}^{\mathrm{PP}} = \mathrm{MP}$.

Third, there is evidence that the bits toward the right or left end of $f(x)$ are not as hard to compute as the bits in the middle. Essentially by definition, the most significant bit gives the power of PP, while the least significant bit gives the power of $\oplus$P. From the proof that $\mathrm{Few} \subseteq \oplus\mathrm{P}$ [BGH90] it is known that the rightmost $O(\log n)$ bits still give the power of $\oplus$P; we show that the leftmost $O(\log n)$ bits do likewise for PP. Meanwhile the interval between them of bits which are as hard to compute as the middle bit comes close to either side.

Finally, it is natural to ask whether two bits of a #P function f are more powerful than one. We show that the answer is *yes* if and only if MP is *not* closed under intersection. The problem of whether MP is closed under $\cap$ leads to a question of general mathematical interest about the size of integer-valued polynomials which satisfy certain congruence equations. We also show that MP is closed under intersection with sets in AmpMP.

Related work has been carried out by F. Green, J. Köbler, and J. Torán. They have studied the class of languages L such that for some #P function f and some polynomial-time computable function t, we have $(\forall x) : x \in L \iff f(x)$ bitwise majorizes $t(x)$. This class contains MP (which is the special case $t(x) := 0^m 10^m$ where $m := (|f(x)| - 1)/2$), and is closed under intersection (consider the bitwise OR of $t_1(x)$ and $t_2(x)$), but is not known to be closed under union or complement. We show that the two classes are equal iff there is a "polynomial-time computable construction" which witnesses that MP is closed under intersection. They have also observed [personal communication] that $\oplus$P is *low* [Sch85] for both these classes; i.e. that $\mathrm{MP}^{\oplus\mathrm{P}} = \mathrm{MP}$ and likewise for the latter class.

2. Counting classes and "middle-bit P"

We use the alphabets $\Sigma := \{0, 1\}$ and $\Gamma := \{0, 1, ;\}$, where ; is an extra symbol used to form tuples. Given a language D, we write $D(x)$ either for '$x \in D$' or for the 0-1 valued characteristic function of D; we likewise identify a predicate R with the language $\{z : R(z)\}$. We study languages over the alphabet Γ, but use Σ^* as a domain for quantifiers to range over. For conciseness we use the following "counting quantifier notation":

Definition 2. For any predicate $Q(\cdot)$, $m \in \mathbf{N}$, and real $\delta \in (0, 1)$:
(a) (*threshold counting*) $(T^m_\delta\, y)\, Q(y)$ holds iff $\|\{ y \in \Sigma^m : Q(y) \}\| > \delta \cdot 2^m$.
(b) (*parity counting*) $(\oplus^m\, y)\, Q(y)$ holds iff $\|\{ y \in \Sigma^m : Q(y) \}\|$ is odd.

Several well-known operators on complexity classes are then definable as follows. (We add square brackets to Toda's '$\cdot$' notation in order to clarify the order of application in compound expressions.)

Definition 3. Let $\mathcal{C}$ be any class of languages over Γ. Then a language $A \subseteq \Gamma^*$ belongs to, respectively, (a) NP[$\mathcal{C}$], (b) $\oplus$P[$\mathcal{C}$], (c) C[$\mathcal{C}$], (d) BP[$\mathcal{C}$], iff there is a language $R \in \mathcal{C}$, a polynomial p, and a $\delta > 0$ such that for all $x \in \Gamma^*$, with $m := p(|x|)$:
(a) $A(x) = (\exists^m\, y)\, R(x\,; y)$,
(b) $A(x) = (\oplus^m\, y)\, R(x\,; y)$,

(c) $A(x) = (T^m_{.5}\, y)\, R(x\,;y)$,
(d) $(T^m_{.5+\delta}\, y)[\, A(x) = R(x\,;y)\,]$.

Taking this twice gives us definitions of Toda's classes BP[$\oplus$P] and C$\oplus$P.

We define a P-*predicate* to be a pair (R, p) such that $R \in$ P and for all $x \in \Gamma^*$ and $y \in \Sigma^*$, $R(x\,;y) \implies |y| < p(|x|)$. The #P function associated to (R, p) is defined by $\#_p R(x) := \|\{\, y : R(x\,;y)\,\}\|$. The original definition of #P in [Val79] was in terms of the number $acc_N(x)$ of accepting computations of polynomial-time NTMs N on given inputs x. Without loss of generality, we restrict attention to those NTMs N which correspond to P-predicates (R, p) as follows: On any input x, N flips $p(|x|)$-many coins and records the results as a 0-1 string y. The computation path of N in which y is the all-zero vector halts without accepting; while each other path calculates the unique string z such that $y \in 0^*1z$, and then evaluates $R(x\,;z)$. Then a function f belongs to #P iff there is a p-time NTM (of this kind) such that for all strings x, $acc_N(x) = f(x)$.

We consider the bounding polynomial p to be a part of either representation, (R, p) or (N, p), of a #P function f. Then f on an argument x takes values between 0 and $2^{p(|x|)} - 1$. We stipulate that these values y are encoded as binary strings of length exactly $p(|x|)$, with the *most* significant bit written leftmost and padded out with leading 0s if necessary. The bits are numbered right-to-left from 1 to $p(|x|)$, so that the ith bit corresponds to the "2^{i-1} place" of the binary number y. We write $BIT(y, i)$ for this bit. Following recent parlance, we also say that y is *top mod* 2^i iff $BIT(y, i) = 1$. If $i > |y|$, then we define $BIT(y, i)$ to be 0. We also write $p/2$ for the function $\lfloor (p(|x|) + 1)/2 \rfloor$. Finally, we define the main class studied in this paper:

Definition 4. A language L belongs to MP if there is a P-predicate (R, p) such that for all x, $x \in L \iff BIT(\#_p R(x), p/2) = 1$.

We also call (R, p) an MP-*representation* for L. Sometimes we omit p, and say '$L \in$ MP via R' or '$L \in$ MP via f', where f is the #P function represented by R.

The following basic structural properties of the class MP follow quickly from this definition.

Proposition 5. *(a)* (PP $\cup$ $\oplus$P) $\subseteq$ MP $\subseteq$ $\mathrm{P}^{\#\mathrm{P}[1]}$.
(b) MP *is closed under complementation.*
(c) MP *has complete sets under* $\leq^{\mathrm{p}}_{\mathrm{m}}$, *and is closed under this reducibility.*

Proof. (a) Given a P-predicate (R, p) for a language $L \in$ PP, define the polynomial $p' := 2p - 1$. Then (R, p') is an MP-epresentation for L, since the leading bit of $\#_p R(x)$ is the middle bit of $\#_{p'} R(x)$. If (R, p) is a $\oplus$P-representation for L, then define $p' := 2p - 1$ as before, and define R' for all $x, z \in \Sigma^*$ by: $R'(x, z) \iff$ there are strings y, w such that $z = yw$, $|w| = p(|x|) - 1$, and $R(x, y)$. Then (R', p') is an MP-representation for L. That $\mathrm{MP} \subseteq \mathrm{P}^{\#\mathrm{P}[1]}$ is immediate.

(b) If $L \in \mathrm{MP}$ via R, then $\overline{L} \in \mathrm{MP}$ via $\neg R$.

(c) The language $U_{\mathrm{MP}} := \{ N ; x ; 0^k : \mathit{BIT}(acc_N(x), k) = \text{`}1\text{'} \}$ belongs to MP, and like the standard "universal sets" for NP, $\oplus$P, etc., is complete even under 1-way log-space reductions. Now suppose that $B \in \mathrm{MP}$ via some #P function f, and that $A \leq_{\mathrm{m}}^{\mathrm{p}} B$ via some PF function g. Since the function $f(g(x))$ is in #P, A is also in MP. □

The proof of (a) shows how "any bit can be moved to the middle"; this idea is used at the end of the proof of Theorem 7, and justifies $U_{\mathrm{MP}} \in \mathrm{MP}$ in (c). In fact, the proof of Theorem 7 shows that MP is closed downward under polynomial-time 1-truth-table reductions ($\leq_{1\mathrm{tt}}^{\mathrm{p}}$); while closure under $\leq_{2\mathrm{tt}}^{\mathrm{p}}$ and so on is tied to the open problem about closure under intersection. Languages such as "middle-bit #SAT" are also complete for MP, but we do not have any examples which are more natural than that.

Now we look at a seemingly more-general notion of dependence on one bit of a #P function.

3. Bits of #P functions

Define an oracle TM M to be a *bit-query machine* if, with some function $f : \Sigma^* \longrightarrow \Sigma^*$ as oracle, M writes queries of the form (y, i), and receives $\mathit{BIT}(f(y), i)$ from the oracle. A bit-query machine may write no other kinds of queries. We still write $L(M^f)$ for the language accepted by the bit-query machine M with oracle function f. We focus first on the power of one bit-query, deferring the concept for multiple queries until Section 5.

Definition 6. A language L belongs to the class $\mathrm{P}^{\#\mathrm{P}[1\text{ bit}]}$ if L is accepted by a deterministic polynomial-time bit-query machine M with some function $f \in \#\mathrm{P}$ as oracle, such that on any input x, M makes at most one bit query.

The definition of MP is the restriction of Definition 6 whereby M on input x must select the argument y to be x itself, makes the query $(x, p(|x|)/2)$, (where p is the bounding polynomial of f), and accepts iff the answer to the query is 'yes'. Intuitively, this is a "many-one" kind of reduction to the middle-bit of f, whereas Definition 6 is intuitively a 1-truth-table reduction. Nevertheless, our first main theorem shows that MP captures the idea of solvability with one bit of a #P function in a fairly robust manner.

Theorem 7. $\mathrm{MP} = \mathrm{P}^{\#\mathrm{P}[1\text{ bit}]}$.

Proof. Let $L \in \mathrm{P}^{\#\mathrm{P}[1\text{ bit}]}$, let M be the polynomial-time oracle machine in Definition 6, and let (R, p) be a P-predicate such that $f = \#_p R$. Then there are polynomial-time computable functions g and d such that for all x, $g(x)$ is the argument y selected for the sole bit-query to f, and $d(x)$ is the selected bit. For any x, one can compute in polynomial time the vector $v(x) := (M(x\,;0), M(x\,;1))$ of answers for the two possible values of the selected bit. Let 'Y/N' abbreviate these answers, so that $v(x)$ is one of (Y,Y), (Y,N), (N,Y), or (N,N). Define $R_{(Y,Y)} :=$ TRUE, $R_{(N,N)} :=$ FALSE, and $R_{(N,Y)} := R$. Then define the predicate $R_{(Y,N)}$ for all $y, z \in \Sigma^*$ to be $(\neg R(y,z) \;\wedge\; |z| < p(|y|))$. This represents the function $(2^{p(|y|)} - 1) - f(y))$. Finally define the predicate S for all x, z by $S(x,z) := R_{v(x)}(g(x), z)$. Then for all x,

$$x \in L \iff BIT(\#S(x), d(x)) = 1.$$

Moreover, there is a polynomial q (depending on g and p) such that (S, q) is a P-predicate. For instance, in the case $v(x) = (Y, Y)$, which guarantees that M accepts x, $\#S(x)$ is one less than a power of 2 of exponent greater than $t(x)$, so that $BIT(\#S(x), t(x)) = 1$.

It remains to "move" bit $t(x)$ to the middle position for all x. Let $q' := 2q - 1$, so $q = q'/2$. Now let $S' := \{\,(x, zw) : |w| = q(|x|) - 1 - d(x) \;\wedge\; S(x,z)\,\}$. Then (S', q') is a P-predicate such that for all x, $x \in L \iff BIT(\#S(x), d(x)) = 1 \iff BIT(\#S'(x), q(|x|)) = 1$. So $L \in \mathrm{MP}$. $\square$

Hence no bit of a general #P function is harder to compute than the middle bit. A more precise statement of this uses the notion of a *bit-selection function*. We define this to be a function d_p which takes a bounding polynomial p as a parameter, such that for all x, $1 \leq d_p(x) \leq p(|x|)$. Sometimes we omit p or write d as a function of the length n of x. For any such d and function $f \in \#\mathrm{P}$, we write $L(f, d)$ for $\{\, x : BIT(f(x), d(x)) = 1 \,\}$.

Corollary 8. *For any polynomial-time computable bit selection function d and #P function f, $L(f,d) \in \mathrm{MP}$. That is, we can find a #P function f' such that the problem of computing the $d(x)$-th bit of $f(x)$ reduces to that of computing the middle bit of $f'(x)$. There is a fixed string w_0 such that we can take $f'(x) := \#SAT(xw_0)$ for all x.* $\square$

Now we look at the complexity of bits other than those near the middle. Keeping the parameter p and writing $L(R, d_p)$ for $\{\, x : BIT(\#_p R(x), d_p(x)) = 1 \,\}$, we give

Definition 9. For any bit-selection function d_p, define $\mathrm{MP}[d_p]$ to be the class of languages $L(R, d_p)$, where (R, p) is a P-predicate.

We assume that bit-selection functions are polynomial-time computable; then by Corollary 8, $\mathrm{MP}[d_p] \subseteq \mathrm{MP}$. Definition 4 originally defined MP as $\mathrm{MP}[p/2]$. The next result tells us formally that many bits near the middle are as hard as the middle bit.

Theorem 10. *Let d_p be a bit-selection function such that for some fixed δ, ϵ, and $k > 0$, and all sufficiently large n, $\delta n^{1/k} < d_p(n) < (1 - \epsilon n^{-k})p(n)$. Then* $\mathrm{MP}[d_p] = \mathrm{MP}$.

Proof. The proof, which is given in the Appendix, is via a standard padding argument. □

The next result tells us that bits very close to either end are probably easier than the middle bit.

Theorem 11.

(a) *[BGH90] If $d_p = O(\log n)$, then* $\mathrm{MP}[d_p] = \oplus\mathrm{P}$.

(b) *If $p - d_p = O(\log n)$, then* $\mathrm{MP}[d_p] = \mathrm{PP}$.

Proof.

(a) The basic idea from [BGH90] is that by *Lucas' theorem*, $BIT(z,k) = 1$ iff the binomial coefficient $\binom{y}{2^{k-1}}$ is odd. If $k = O(\log n)$, then this coefficient gives a $\oplus$P representation.

(b) By the closure under intersection of PP [BRS91] it follows that the language

$$L' := \{(x, k, k') | k \leq \#_p R(x) < k'\}$$

is in PP. The $d_p(x)$-th bit of $\#_p R(x)$ equals '1' if and only if $\#_p R(x)$ lies in one of at most $2^{p(|x|)-d_p(x)}$ many intervals of the form $[c2^{d_p(x)} + 2^{d_p(x)-1}, (c+1)2^{d_p(x)} - 1]$. If $p(|x|) - d_p(x)$ is logarithmically bounded in $|x|$ there are only polynomially many such intervals, therefore L can be decided by polynomially many non-adaptive questions to a L'-oracle. Hence $L \in \mathrm{PP}$ by the closure of PP under polynomial-time truth-table reductions [FR91].

□

4. Subclasses of MP and Lowness

S. Toda [Tod89] proved that $\mathrm{PH} \subseteq \mathrm{BP}[\oplus\mathrm{P}]$ and that $\mathrm{C}\oplus\mathrm{P} \subseteq \mathrm{P}^{\#\mathrm{P}}$. His proof of the latter actually showed that $\mathrm{C}\oplus\mathrm{P} \subseteq \mathrm{MP}$. We observe that a slight modification of this proof highlights a special property of languages in $\mathrm{C}\oplus\mathrm{P}$.

In this section, we will use MP-representations in which the "decision bit" of a value $f(x)$ is not necessarily in the middle, but is selected by a function $d_{p_0}(n)$. We will be interested in the numbers s and t of 0s to the left and right, respectively, of bit $d_{p_0}(|x|)$ in the binary representation of $f(x)$. Throughout this section, p, q, r, s, t, u, and d_{p_0} itself will be polynomials, either of $n := |x|$ or of $m := u(|x|)$; often we leave the argument out to save clutter. We also use regular-expression notation as a convenient shorthand for representing sets of possible values of $f(x)$.

Theorem 12. *For any language $D \in \mathrm{C}\oplus\mathrm{P}$, there is a polynomial p such that for all polynomials s, we can find a $\#\mathrm{P}$ function f such that*

(i) $D = L(f,p)$ (so $D \in \mathrm{MP}$), and

(ii) for every x, $f(x) \in \{0,1\}^* \cdot 0^{s(n)} \cdot D(x) \cdot \{0,1\}^{p(n)-1}$.

Here $D(x) =$ '1' if $x \in D$ and $D(x) =$ '0' if $x \notin D$. What this says is that D has MP-representations in which there are any desired polynomial number of 0s to the left of the "decision bit," where the location of the decision bit remains fixed at $p(n)$. We call this property *amplification* on the *left* of the decision bit."

Proof. (For details and specific references to Toda's original proof of $\mathrm{C}\oplus\mathrm{P} \subseteq \mathrm{P}^{\#\mathrm{P}}$, see the Appendix.) $\square$

Our notion of "right-amplification" is simpler; namely, that for any polynomial t, there should exist an MP-representation via a $\#\mathrm{P}$ function f which for any x gives $t(x)$-many 0s immediately to the right of the decision bit. To motivate this, we observe:

Lemma 13.

(a) Let $D \in \mathrm{BPP}$. Then for every polynomial t there is a polynomial p and a $\#\mathrm{P}$-function f such that for every x

$$f(x) \in D(x) \cdot 0^t \cdot \{0,1\}^p.$$

(b) Let $D \in \oplus\mathrm{P}$. Then for every polynomial s there is a $\#\mathrm{P}$ function f such that for every x

$$f(x) \in \{0,1\}^* \cdot 0^s \cdot D(x).$$

(c) *Let* $D \in \mathrm{BP}[\oplus\mathrm{P}]$*. Then for every polynomial* t *there is a polynomial* p *such that for every polynomial* s *there is a* #P*-function* f *such that for every* x

$$f(x) \in \{0,1\}^* \cdot 0^s \cdot D(x) \cdot 0^t \cdot \{0,1\}^p.$$

Proof.

(a) Let t be given. By the well-known amplification lemma for BPP there is a polynomial p and a #P-function g such that for every x

$$x \in D \Longrightarrow g(x) \in 1^{t+1} \cdot \{0,1\}^{p(|x|)}$$

$$x \notin D \Longrightarrow g'(x) \in 0^{t+1} \cdot \{0,1\}^{p(|x|)}.$$

By setting $f(x) := g(x) + 2^{p(|x|)}$ we obtain (a).

(b) This follows directly from Toda's recurrence lemma [Tod91].

(c) Because $\oplus$P is closed under polynomial-time Turing-reductions we can amplify BP[$\oplus$P] in the same manner as BPP. Hence there is a set $A \in \oplus\mathrm{P}$ and a nondeterministic transducer M such that for any input x, the number of computation paths which compute a string $y \in A$ has a binary representation which matches the regular expression $D(x) \cdot 0^t \cdot \{0,1\}^p$. By Toda's lemma A can be decided by a NTM N which on input y has $\{0,1\}^* \cdot 0^{s+p+t+1} \cdot A(y)$ many accepting computations. By combining M and N we get a NDTM which behaves as stated.

□

Intuitively speaking, languages in BPP and $\oplus$P give the strongest kinds of right and left amplification, respectively, while languages in BP[$\oplus$P] have MP representations which simultaneously give left and right amplification. We focus on this last case in the key formal definition of this section:

Definition 14. Let AmpMP be the class of all sets D such that for every polynomial t, there is a polynomial p such that for every polynomial s, there is a function $f \in$ #P such that for every $x \in \{0,1\}^*$,

$$f(x) \in \{0,1\}^* \cdot 0^s \cdot D(x) \cdot 0^t \cdot \{0,1\}^p.$$

It is immediate that BP[$\oplus$P] $\subseteq$ AmpMP. It is not obvious whether these classes are equal; indeed, it does not seem that AmpMP is closed under $\leq_m^p$-reductions. However, we can show that AmpMP is closed under complementation. For a string $v \in \{0,1\}^*$ let $\overline{v}$ denote its bitwise complement.

Lemma 15. AmpMP *is closed under complementation.*

Proof. Let $D \in$ AmpMP and let the polynomials p, s, t and function $f \in \#\mathrm{P}$ (with bounding polynomial p_0) be as in Definition 14; i.e.,

$$f(x) \in \{0,1\}^* \cdot 0^s \cdot D(x) \cdot 0^t \cdot \{0,1\}^p.$$

Define f' to be the polynomial-time computable function whose value on any input x of length n has the bitstring representation $1^{s(n)} \cdot 0 \cdot 1^{t(n)} \cdot 0^{p(n)}$. Then the function $g(x) := 2^{p_0(n)} - 1 - (f(x) + f'(x))$, which gives the bitwise complement of $f + f'$, is in $\#\mathrm{P}$, and for every x,

$$g(x) \in \{0,1\}^* \cdot 0^s \cdot \overline{D(x)} \cdot 0^t \cdot \{0,1\}^p.$$

Hence $\overline{D} \in$ AmpMP. $\square$

The following theorem says that the number of accepting computations of an oracle-NTM M with oracle from AmpMP is encoded in the number of accepting computations of a certain non-oracle NTM N uniformly for every input.

Theorem 16.

(a) *Let M be an ONTM with oracle set $A \in$ BPP. Then there are a NTM N (without oracle) and a polynomial v such that for every x,*

$$acc_N(x) \in acc_{M^A}(x) \cdot \{0,1\}^{v(|x|)}.$$

(b) *Let M be an ONTM with oracle set $A \in \oplus\mathrm{P}$. Then there is a NTM N (without oracle) such that for every x,*

$$acc_N(x) \in \{0,1\}^* \cdot acc_{M^A}(x).$$

(c) *Let $\mathcal{C} \subseteq$ AmpMP be a class which is closed under $\leq_m^p$-reductions. Let M be an ONTM with oracle set $A \in \mathcal{C}$. Then there are a NTM N without oracle and a polynomial v such that for every x,*

$$acc_N(x) \in \{0,1\}^* \cdot acc_{M^A}(x) \cdot \{0,1\}^{v(|x|)}.$$

Proof. The idea of the proof of (a) is as follows. We may assume wlog. that M has the same number of nondeterministic guesses and oracle questions on every computation path. Since BPP is closed under $\leq_{\mathrm{m}}^{\mathrm{p}}$, we may suppose that the oracle set A is one which allows all queries y made by M on inputs of length n to have equal length; this "honesty property" also allows us to regard all polynomials as functions of n rather than $|y|$. We use two NTMs N_1, N_2 that accept $A, \overline{A}$ respectively with right amplification from Lemma 13(a) which is chosen sufficiently large. The non-oracle machine N then simulates a single query of M on a string y by the construct:

(1) Guess the answer YES or NO of the oracle,

(2) If the guessed answer is YES then simulate $N_1(y)$ otherwise $N_2(y)$,

(3) If the simulation ends with REJECT, then reject immediately; otherwise go on as M.

(This is essentially the same simulation as in the proof by C. Papadimitriou and S. Zachos of $\oplus\mathrm{P}^{\oplus\mathrm{P}} = \oplus\mathrm{P}$ [PZ83].) Then the accepting computation paths of N on input x can be partitioned into those in which all "guessed" oracle answers are correct, and those which have some wrong guess. The former amount to a term of the form $1 \cdot 0^t \cdot \{0,1\}^p$ for some polynomials p, t (both depending on the right amplification of N_1 and N_2). The latter only contribute a much smaller term of the form $\{0,1\}^p$. Summing this up over all computations of M gives us (a).

(b) is similar.

To obtain (c) we choose the polynomial s from Definition 14 for the machines N_1, N_2 large enough so that the leading $\{0,1\}^* \cdot 0^s$ bitstring doen't disturb the bitstring obtained in (a). (Essentially, all calculations are modulo 2 to the power $s+1+t+p$, which can be made bigger than all the numbers occuring in (a).) For technical details see the complete proof in the Appendix. □

Corollary 17.

(a) $\mathrm{BP}[\oplus\mathrm{P}]$ *is low for* MP.

(b) If $\mathrm{C_=P} \subseteq \mathrm{AmpMP}$ *then* $\mathrm{PP}^{\mathrm{PP}} = \mathrm{MP}$.

By combining Theorem 16 with $\mathrm{PH} \subseteq \mathrm{BP}[\oplus\mathrm{P}] \subseteq \mathrm{AmpMP}$ one may quickly obtain a result obtained via lengthy calculations by Toda and O. Watanabe [TW90]:

Corollary 18. $\#\mathrm{PH} \subseteq \mathrm{PF}^{\#\mathrm{P}[1]}$.

5. The power of more than one bit

Definition 19. Let k be a number (or a numerical function of $|x|$). A language L belongs to the class $\mathrm{P}^{\#\mathrm{P}[k \text{ bits}]}$ if L is accepted by a deterministic polynomial-time bit-oracle machine M with some function $f \in \#\mathrm{P}$ as oracle, such that on any input x, M makes at most k-many bit queries.

When k is fixed, allowing M to have k-many different #P functions $f_1, \ldots, f_k$ in the oracle does not add any more power, since the function $F(x\,; k) := f_k(x)$ is also in #P. When $k(|x|)$ is variable, there are issues we address later.

The intuition behind our next result is that for fixed $k > 1$, the k-many "adaptive" queries allowed can be replaced by 2^k many non-adaptive queries. Unlike the case of the open problem of whether $\mathrm{P}^{\#\mathrm{P}[1]} = \mathrm{P}^{\#\mathrm{P}[2]}$, where the oracle returns the whole function value $f(y)$ on a query y, this works because the amount of information returned by each query is finite and independent of x. Then a standard "truth-table reduction" argument can compress these down to one query to a powerful #P function f. However, this argument requires that MP be closed under all Boolean operations, and the result shows that the conditions are actually equivalent.

Theorem 20. MP *is closed under intersection if and only if, for every fixed* $k \geq 1$, $\mathrm{P}^{\#\mathrm{P}[k \text{ bits}]} = \mathrm{MP}$.

Proof (sketch). ($\Longleftarrow$): Given $A, B \in \mathrm{MP}$, $A \cap B \in \mathrm{P}^{\#\mathrm{P}[2 \text{ bits}]}$.

($\Longrightarrow$): Let M be a k-query bounded bit-query machine which accepts L with the #P function f as oracle. For any x we can label the up to $(2^k - 1)$-many possible queries made by M by their position i in the query tree, and thus associate to M, $(2^k - 1)$-many bit-selection functions $d_i(x)$, each of which is polynomial-time computable. Then for each i, the language $A_i := \{\, x : BIT(f(x), d_i(x)) = 1 \,\}$ belongs to MP. By the closure of MP under intersection, all of the 2^{2^k-1}-many Boolean combinations of these languages belong to MP, and by extending the idea of the proof of Theorem 7, one can combine these into a single MP-representation for L. (The Appendix gives details for the case of two nonadaptive queries.) $\square$

This result highlights the interest in showing that MP is closed under intersection. The closest thing to this which we have been able to prove is:

Proposition 21. *If* $A \in \mathrm{AmpMP}, B \in \mathrm{MP}$ *then* $A \cap B \in \mathrm{MP}$.

Proof. Let (R, p) be a P-predicate witnessing that B is in MP. We choose a $\#P$-function f such that for every x,

$$f(x) \in \{0,1\}^* \cdot 0^p \cdot D(x) \cdot 0^p \cdot \{0,1\}^q$$

for some polynomial q, where $D(x) = 1 \iff x \in A$. By setting $g(x) := f(x)\#_p R(x)$ we get a #P-function which has a 1 in position $p + q + p/2$ if and only if $x \in A \cap B$. $\square$

In the proof of Theorem 20, the languages $A_1, ..., A_m$ (where $m := 2^k$) are fixed and independent of the input x to the machine M recognizing L. Hence we need only suppose that the MP-representations of all the Boolean combinations of these languages *exist*. Many closure properties studied in complexity theory are non-effective, and this suffices for many results. However, when k varies with $|x|$, say

$k(n) = O(\log n)$, we may need to *calculate* a variable number of MP-representations, as needed. This is one of few instances we know where the distinction between a closure property which can be realized by a polynomial-time construction and the above is important. In our formal definition we suppose that P-predicates (R, p) and bit-selection functions d (which are wlog. polynomials) are encoded by strings over Γ in some reasonable manner.

Definition 22. MP is *polynomial-time constructively closed under intersection* if there is a polynomial-time computable function F such that for any two P-predicates (R, p) and (S, q) and bit-selection functions d_1, d_2, $F(R, S, d_1, d_2)$ outputs a P-predicate (T, r) and a bit-selection function d_3 such that for all x,

$$BIT(\#_r T(x), d_3(x)) = 1 \iff BIT(\#_p R(x), d_1(x)) = 1 \wedge BIT(\#_q S(x), d_2(x)) = 1.$$

In addition we require that the length of (T, r) is no more than a constant factor greater than the maximum length of (R, p) and (S, q).

We say *p-closed* for short. Since the operation of "moving a bit to the middle" is polynomial-time effective, there would be no loss of generality in restricting this definition to the case $d_R = p/2$ etc. But the above form is more convenient. Note that MP is p-closed under complements. The well-known closures of #P functions under addition, multiplication, and concatenation of bitstring representations $f(x)g(x)$ are also polynomial-time constructive; the last operation expands by a factor of 2. The closure of PP under intersection [BRS91] also meets the conditions of Definition 22.

Lemma 23. *Let D be a language, and let F be a polynomial-time computable function which is used by some deterministic polynomial-time procedure P. Suppose that on any input x, P invokes F to obtain a* P*-predicate (R_x, p_x) and a bit-selector d_x (both depending on x) such that $x \in D \iff BIT(\#_{p_x} R_x(x), d_x(x)) = 1$, and such that $p_x(|x|)$ is bounded by a polynomial in $|x|$. Then $D \in$* MP.

Proof. The predicate $S(x, y) := |y| \leq p_x(|x|) \ \wedge \ R_x(x, y)$ is a P-predicate, and $d(x) := d_x(x)$ is a polynomial-time computable bit-selection function. □

Theorem 24. *If* MP *is p-closed under intersection, then* $\mathrm{MP} = \mathrm{P}^{\#\mathrm{P}[O(\log n)\ \mathrm{bits}]}$.

Proof. The main point of the proof, which is given in the Appendix, is that the polynomial-time effectivity saves us a level of exponentiation over the method of Theorem 20. □

J. Köbler, J. Torán, and F. Green [personal communication] have studied the class of languages L such that for some $f \in \#\mathrm{P}$ and $g \in \mathrm{PF}$, and all x, $x \in L$ iff the binomial coefficient $\binom{f(x)}{g(x)}$ is odd. By *Lucas' Theorem* (see [Smo91]), this happens iff no bit of $g(x)$ is '1' where the corresponding bit of $f(x)$ is '0'; i.e., if $f(x)$ "masks" $g(x)$. For want of a better or agreed-on name, call this class MaskP. Then MaskP contains MP, which is the special case where the bitstring $g(x)$ has a single '1' in it. Observe that in terms of "MaskP-representations" of L_1 by (f_1, g_1) and L_2 by (f_2, g_2), MaskP is closed under intersection by the polynomial-time construction which gives $f_3(x) := 2^{p(|x|)} f_1(x) + f_2(x)$ (where p is the bounding polynomial for f_2) and $g_3(x)$ as the concatenation of $g_1(x)$ and $g_2(x)$. However, it is not known whether MaskP is closed under complements, even in the noneffective sense.

Proposition 25. MP *is p-constructively closed under intersection* $\Longleftrightarrow$ MP *p-constructively equals* MaskP. *(The latter entails, not only is* MP = MaskP, *but also there is a polynomial-time computable translation from* MaskP*-representations into* MP*-representations.)*

Proof. ($\Longrightarrow$): Given $D \in$ MaskP, let f and g represent D. For every fixed j, let $B_j := \{\, y : BIT(f(y), j) = 1 \,\}$. Given x, one can compute an MP-representation R_x for $\bigcap\{\, B_j : BIT(g(x), j) = 1 \,\}$. Then Lemma 23 gives an MP-representation for D.

($\Longleftarrow$): Given such a translation, and the previously mentioned translation from MP to MaskP, the construction for intersection in MaskP gives one for MP. $\square$

6. Conclusions and Open Problems

We have shown that the concept of the power of individual bits of a #P function is natural and robust. It leads to a class MP which has many interesting properties. The regions between $\mathrm{C}\!\oplus\!\mathrm{P}$ and P^{PP} and between P^{PP} and $\mathrm{PP}^{\mathrm{PP}}$ seem to represent the end of our current knowledge about counting classes; not even relativized separations are known. The results we have given for MP may be helpful in this study.

We were not able ourselves to obtain relativized separations of MP from either $\mathrm{C}\!\oplus\!\mathrm{P}$ or $\mathrm{P}^{\#\mathrm{P}[1]}$. The most interesting open problem, which we investigated thoroughly, is whether MP is closed under intersection. A related purely mathematical question which we have circulated among mathematicians is:

In terms of k, what is the minimum degree (and/or size of rational coefficients) of an integer-valued polynomial $p(x, y)$ such that for all x, y, $p(x, y)$ is top modulo 2^k $\Longleftrightarrow$ both x and y are top modulo 2^k?

If such p can be found with degree polynomial in k, then p can be written as a polynomial-sized sum of small binomial coefficients in x and y, which can then be used in building P-predicates (or polynomial-time NTMs). Then it would follow (after "lining-up" decision bits) that MP is (p-)closed under intersection. The simplest polynomial we know for this congruence relation is $p(x,y) := 2^{k-1}\binom{x}{2^{k-1}}\binom{y}{2^{k-1}}$; while there are somewhat smaller ones, they all have exponential degree and coefficient size. A similar congruence relation with the same open problem is (everything modulo 2^k) $p(x,y) = 0 \iff (x = 0 \ \wedge \ y = 0)$.

Acknowledgments. We would like to thank Jim Royer, Richard Beigel, Clemens Lautemann, Andrew Odlyzko, Mattijas Costner, Johannes Köbler, Jacobo Torán, Frederic Green, and Lane Hemachandra for helpful comments and suggestions related to this work. We are also grateful to the administrators and benefactors of the *Internet*: some of the writing and even some of the actual research was conducted by us "in real time" over a UNIX 'talk' connection between Mainz and Buffalo.

References

[ACGS84] W. Alexi, B. Chor, O. Goldreich, and C.P. Schnorr. RSA/Rabin bits are $1/2 + 1/poly(logN)$ secure. In *The Proceedings of the 25th Annual IEEE Symposium on Foundations of Computer Science*, pages 449–457, 1984.

[BGH90] R. Beigel, J. Gill, and U. Hertrampf. Counting classes: thresholds, parity, mods, and fewness. In *The Proceedings of the Symposium on Theoretical Aspects of Computer Science*, pages 49–57, 1990.

[BRS91] R. Beigel, N. Reingold, and D. Spielman. PP is closed under intersection. In *The Proceedings of the 23rd ACM Symposium on the Theory of Computation*, pages 1–11, 1991.

[Cho86] B. Chor. *Two Issues in Public-Key Cryptography: RSA Bit Security and a New Knapsack-Type System.* The MIT Press, 1986.

[FR91] L. Fortnow and N. Reingold. PP is closed under truth-table reductions. In *The Proceedings of the 6th Annual IEEE Conference on Structure in Complexity Theory*, pages 13–15, 1991.

[PZ83] C. H. Papadimitriou and S. Zachos. Two remarks on the power of counting. In *The 6th GI Conference on Theoretical Computer Science*, Lecture Notes in Computer Science No. 145, pages 269–276. Springer-Verlag, 1983.

[Sch85] U. Schöning. *Complexity and Structure*, volume 211 of *Lecture Notes in Computer Science*. Springer-Verlag, 1985.

[Smo91] C. Smorynski. *Logical Number Theory*, volume 1. Springer-Verlag, 1991.

[SS90] A. Schrift and A. Shamir. The discrete log is very discreet. In *The Proceedings of the 22nd Annual ACM Symposium on the Theory of Computing*, pages 405–415, 1990.

[Tod89] S. Toda. On the computational power of PP and $\oplus$P. In *The Proceedings of the 30th Annual IEEE Symposium on Foundations of Computer Science*, pages 514–519, 1989.

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

[Tor91] J. Torán. Complexity classes defined by counting quantifiers. *Journal of the ACM*, 38:753–774, 1991.

[TW90] S. Toda and O. Watanabe. Polynomial time 1-turing reductions from #PH to #P. Technical report, University of Electro-Communications, Tokyo, 1990.

[Val79] L. Valiant. The complexity of enumeration and reliability problems. *SIAM Journal on Computing*, 8:410–421, 1979.

[Wag86] K. Wagner. The complexity of combinatorial problems with succinct input representation. *Acta Informatica*, 23:325–356, 1986.

Appendix

Proof of Theorem 10.

Fix (R_0, p_0) to be the P-predicate for SAT, where p_0 can be the identity. First, for any string x put $n_x := \lceil |x|^k / 2^k \delta \rceil - |x| - 1$. Then define the predicate R_1 as the set $\{\, (0^{n_x} 1x, y) : R_0(x, y) \,\}$. Take $p_1 := p_0$. In terms of the length n_x of the first argument of R_1, bit $\lceil \delta n_x^{1/k} \rceil$ of $\#_{p_1} R_1(x)$ equals bit $\lceil |x|/2 \rceil$ of $\#_{p_0} R_0(x)$. Since the map from x to $0^{n_x} 1x$ is computable in polynomial time, the two bits have the same power. Second, letting $n := |x|$, define $m_x := \lceil n^k p_0(n) / 2^k \epsilon \rceil$, $p_2(n) := p_0(n) + m_x$, and $R_2 := \{\, (x, wy) : |w| = m_x \wedge R_0(x, y) \,\}$. Then bit $\lceil (1 - \epsilon n^{-k}) p(n) \rceil$ of $\#_{p_2} R_2(x)$ equals bit $\lceil |x|/2 \rceil$ of $\#_{p_0} R_0(x)$. Similar padding methods apply for any bit $d_p(n)$ between these two extremes. $\square$

Proof of Theorem 12. (This is essentially the proof in [Tod91].) Let $D \in \mathrm{C}{\oplus}\mathrm{P}$. Then there is a language $B \in \oplus\mathrm{P}$ and a polynomial p such that for all x, putting $n := |x|$, $S := \Sigma^{p(n)}$, we have $x\,;0^{p(n)} \notin B$ and

$$x \in D \iff \|\{\, w \in S : x\,; w \in B \,\}\| \geq 2^{p(n)-1}.$$

Without loss of generality we may suppose that $x\,;0^{p(n)} \notin B$, so that $g(x) := \|\{\, w \in S : x\,; w \in B \,\}\|$ is at most $2^{p(n)} - 1$. This p is the polynomial in the statement of the theorem. Let $m := |x\,; w| = n + 1 + p(n)$. Given the polynomial s, take q to be any polynomial such that for all n, $q(m) > s(n) + p(n)$. Then by the key recurrence lemma of [Tod91], there is a function $h \in \#\mathrm{P}$ such that for all x and w,

$x\,; w \in B \Longrightarrow h(x\,; w) \equiv 1 \pmod{2^{q(m)}}$, and
$x\,; w \in B \Longrightarrow h(x\,; w) \equiv 0 \pmod{2^{q(m)}}$.

(The bounding polynomial of h is large and of no consequence.) Then for all x define $f(x) := \sum_{w \in S} h(x\,; w)$. This is also a #P function. Modulo $2^{q(m)}$, the terms in the sum for those w such that $x\,; w \notin B$ drop out, while each other term contributes 1. Hence $f(x) \equiv g(x) \pmod{2^{q(m)}}$, so that $x \in D \iff BIT(f(x), p(|x|)) = 1$. Moreover, every bit i such that $p(n) < i < q(m)$ must be 0, and the second conclusion follows. $\square$

The next lemma has a more technical flavor. It states essentially that one can decide by looking at a single bit of a product of r numbers of the form $\{0,1\} \cdot 0^t \cdot \{0,1\}^P$ whether all factors have a leading 1, given that $t > r$ and that behind this deciding bit there are again about $t - r$ zeros. It is used in the proof of Theorem 16.

Lemma 26. *Let k, p, r, t be natural numbers, $t > r > k$. Then the following is true*

(a)

$$\prod_{i=1}^{r} 1 \cdot 0^t \cdot \{0,1\}^p \subseteq 1 \cdot 0^{t-r+1} \cdot \{0,1\}^{r(t+p)+r-t-1},$$

(b)

$$(\prod_{i=1}^{k} 1 \cdot 0^t \cdot \{0,1\}^p) \times (\prod_{i=1}^{r-k} \{0,1\}^p) \subseteq \{0,1\}^{r(t+p)-t+1}.$$

Proof of Theorem 16.

(a) Let the number of steps of M be bounded by the polynomial q. Wlog. we may assume that M asks the oracle exactly $r(|x|)$ times with strings of length $u(|x|) > |x|$ each on any computation for some polynomials r, u.

Let t be a polynomial with $t > q + r + 1$.

By lemma 13 there is a polynomial p and an $f \in \#\mathrm{P}$ s.t for every y queried by M

$$f(y) \in D(y) \cdot 0^{t(|x|)} \cdot \{0,1\}^{p(|x|)},$$

where $D(y) = 1$ iff $y \in X$, otherwise $D(y) = 0$. Here we can write polynomials in x because of our general assumption that all queries have the same polynomial length.

Because of the closure under complementation of BPP there are NTMs N_1, N_2, s.t.

$$acc_{N_1}(y) \in D(y) \cdot 0^t \cdot \{0,1\}^p$$

and

$$acc_{N_1}(y) \in \overline{D(y)} \cdot 0^t \cdot \{0,1\}^p.$$

Now we describe the simulation of M by N.

N works in general as M. Whenever M queries the oracle about some string y, N nondeterministcally guesses the answer of the oracle, YES or NO. If the answer is YES, it simulates N_1 on input y, if the answer is NO, it simulates N_2 on input y.

If any simulation ends in a REJECT state, that computation path of N rejects. If a simulation ends in an ACCEPT state of N_1 or N_2, N goes on simulating M.

If N passes all simulations without reaching a rejecting state it accepts if and only if M would accept.

Each computation of M results in 2^r computation trees of N (corresponding to the 2^r different guesses of the oracle answers), each consisting of at most $2^{r(t+p)}$ computation paths generated by the simulation of the oracle.

We now estimate the number of accepting computations of N.

Let y be a computation path of M on input x. We define $a_1(y)$ to be the number of accepting computations of N contributed by the correct simulation (i.e the simulation with correct guesses) of y and for an incorrect simulation z let $a_2(y,z)$ to be the number of accepting computations of N contributed by all incorrect simulations of y. With A_1, A_2 we denote the total sum of accepting computations contributed by correct and incorrect computation trees respectively. We show

(1) $A_1 \in acc_{M^X}(x) \cdot 0 \cdot \{0,1\}^{v-1}$ and
(2) $A_2 \in \{0,1\}^{v-1}$

for some polynomial v, so that we get

$$acc_N(x) = A_1 + A_2 \in acc_{M^X}(x) \cdot \{0,1\}^{v(|x|)}$$

and the theorem follows.

For any rejecting computation y of M we have $a_1(y) = 0$ because with correct guesses, N will reject if M does. For an accepting computation y every single oracle simulation contributes a factor of $1 \cdot 0^t \cdot \{0,1\}^p$ to $a_1(y)$. By the Lemma 26 we get

$$a_1(y) \in 1 \cdot 0^{t-r+1} \cdot \{0,1\}^{r(t+p)+r-t-1}.$$

Therefore, by summing up over all y,

$$A_1 \in acc_{M^X} \cdot 0^{t-r+1-q} \cdot \{0,1\}^{r(t+p)+r-t-1+q}$$

because $acc_{M^X} < 2^q$ and (1) follows from $t > r+q$ by setting $v := r(t+p)$.

For an incorrect simulation z of a computation y of M any single correct oracle simulation contributes a factor of $1 \cdot 0^t \cdot \{0,1\}^p$ to $a_2(y,z)$ whereas any single incorrect simulation contributes $\{0,1\}^p$. By the previous lemma, because at most $r-1$ guesses are correct, $a_2(y,z)$ is in $\{0,1\}^{r(t+p)-t+1}$ for every y. By summing up over all y and z we get

$$A_2 \in \{0,1\}^{r(t+p)-t+1+q+r}$$

from which we conclude (2) by observing $v > r(t+p) - t + 1 + q + r$.

So we have proven (a).

(b) The proof is subsumed by that of (c).

(c) We need only a slight modification to extend the proof of (a) to the case of an AmpMP-oracle X. Because C is closed under polynomial-time many-one-reductions we can assume in this case too that all queries have the same length. After having chosen t and determining p we choose s such that

$$s > r(t+p) + q$$

and get by the definition of AmpMP and its closure under complementation NTMs N_1, N_2, s.t.

$$acc_{N_1}(y) \in \{0,1\}^* \cdot 0^{s-p-t-1} \cdot D(y) \cdot 0^t \cdot \{0,1\}^p$$

and

$$acc_{N_1}(y) \in \{0,1\}^* \cdot 0^{s-p-t-1} \cdot \overline{D(y)} \cdot 0^t \cdot \{0,1\}^p.$$

Then s is big enough to assure that the "garbage terms" to the left of $D(y)$ does not disturb the bit representation of A_1 and A_2. We therefore have

(1) $A_1 \in \{0,1\}^* \cdot acc_{M^X}(x) \cdot 0 \cdot \{0,1\}^{v-1}$ and
(2) $A_2 \in \{0,1\}^* \cdot 0^{s-v} \cdot 0 \cdot \{0,1\}^{v-1}$

for the same $v = r(t+p)$ as in (a) and we get

$$acc_N(x) = A_1 + A_2 \in \{0,1\}^* \cdot acc_{M^X}(x) \cdot \{0,1\}^{v(|x|)}$$

□

Proof of Theorem 20. (in the case of $k = 2$ nonadaptive queries): Let $L \in \mathrm{P}^{\#\mathrm{P}[2 \text{ bits}]}$ via a bit-query machine M which on any input x makes two queries to its #P oracle function f; let the queries be $(y_1, d_1(x))$ and $(y_2, d_2(x))$, where d_1, d_2, and the functions g_1, g_2 giving y_1, y_2 in terms of x are polynomial-time computable. Let $L_1 := \{\, x : \mathit{BIT}(f(g_1(x)), d_1(x)) = 1 \,\}$, and let $L_2 := \{\, x : \mathit{BIT}(f(g_2(x)), d_2(x)) = 1 \,\}$. Then L_1 and L_2 belong to MP by Theorem 7. If MP is closed under intersection, then MP is closed under all Boolean operations. Let $L_{(1,1)}$ be $L_1 \cap L_2$, let $L_{(1,0)}$ be $L_1 \setminus L_2$, let $L_{(0,1)}$ be $L_2 \setminus L_1$, and let $L_{(0,0)}$ be $\overline{L_1 \cup L_2}$. There are sixteen languages which can be formed as unions of these four, and these unions correspond to the sixteen yes/no 4-vectors $v(x)$ representing possible behaviors of the oracle machine M in Definition 19 according to the four possible values of the decision bits $(d_1(x), d_2(x))$. For each $v(x)$ we can take an MP representation of the corresponding language by a P-predicate $R_{v(x)}$ and a polynomial q. By "moving bits" as in the last proof, we can arrange that q is the same for each of the sixteen languages. Finally, define $S(x, z) := R_{v(x)}(g(x), z)$. □

Proof of Theorem 24. Let $D := L(M^f)$. First, consider the case where M on any input x makes $k_x := \lfloor c \log n \rfloor$-many nonadaptive bit queries $(y_1, b_1), \ldots (y_{k_x}, b_{k_x})$ (of course these depend on x). For each i, $1 \leq i \leq k_x$, let $A_i := \{\, x : k_x \geq i \wedge \mathit{BIT}(f(y_i), b_i) = 1 \,\}$. Then each A_i is in MP, and MP-representations for all the A_i and complements $\overline{A_i}$ can be found in polynomial time.

Now enumerate the $m := 2^{k_x}$ possible vectors of yes/no answers to these queries as $v_1, \ldots, v_m$. For each j, $1 \leq j \leq m$, use the polynomial-time constructor F from Definition 22 to find an MP-representation for the language B_j of strings x such that v_j gives the answers to the queries made by $M^f(x)$. (For instance, if $k_x = 3$ and v_5 = '(yes,no,yes)' then $B_5 = A_1 \cap \overline{A_2} \cap A_3$.) Finally, in deterministic polynomial time, we can calculate the outcomes a_j of $M(x)$ with each possible query response v_j.

Then using F again we compute an MP-representation (R_x, p_x, d_x) for the language $C := \bigcup\{\, B_j : a_j = \mathit{accept} \,\}$. Here it is important that the "application tree" of F to the polynomially many B_j have logarithmic depth. Lemma 23 then completes the MP-representation for D.

In the adaptive case, there are up to $(2^{k_x} - 1)$-many different bit queries which $M^f(x)$ might make. Then we obtain polynomially many MP languages $A_1, \ldots A_l$ defined as before, where $l \leq 2^{k_x} \leq n^c$. However, there are still only 2^{k_x}-many possible vectors v_j of outcomes to the k_x-many queries actually made by $M^f(x)$. Then the MP-representations of the languages B_j and finally C are defined and computed as before. □