Descriptive Complexity of \#P Functions

Sanjeev Saluja*  K.V. Subrahmanyam†
Department of Computer Science
Tata Institute of Fundamental Research
Bombay 400 005, India

Madhukar N. Thakur‡
Computer & Information Sciences
University of California
Santa Cruz, CA 95064, USA

Abstract

We give a logic based framework for defining counting problems and show that it exactly captures the problems in Valiant's counting class \#P. We study the expressive power of the framework under natural syntactic restrictions and show that some of the subclasses obtained in this way contain problems in \#P with interesting computational properties. In particular, using syntactic conditions, we isolate a class of polynomial time computable \#P problems, and also a class in which every problem is approximable by a polynomial time randomized algorithm. These results set the foundation for further study of the descriptive complexity of the class \#P.

In contrast, we show, under reasonable complexity theoretic assumptions, that it is an undecidable problem to tell if a counting problem expressed in our framework is polynomial time computable or is approximable by a randomized polynomial time algorithm. Finally, we discuss some open problems which arise naturally from this work.

1 Introduction

In computer science, computational models like Turing machines have been well studied as recognizers of languages (subsets of \{0,1\}^*). This is especially true in complexity theory where resources like time and space can be defined naturally using these machine models and the resource complexity of recognizing languages can be studied. On the other hand, if strings in \{0,1\}^* are viewed as encodings of finite structures over an appropriate vocabulary, then formulae in a certain logic can be viewed as recognizers of languages in the following way : Let \sigma be a vocabulary and let \phi be a sentence in a certain logic. One can associate with the formula \phi the language \{e(A) \in \{0,1\}^* : A \models \phi\} where A is a finite structure over the vocabulary \sigma and e(A) is an encoding of the structure A as a string in a natural way.

Such a logical framework for defining subsets of \{0,1\}^* becomes much more interesting to a complexity theorist if there are appropriate logics that capture exactly the languages in the complexity classes that are otherwise defined using computational machine models. Such logical characterizations capture computational complexity without involving computation directly and suggest that logical expressibility of problems may determine their computational complexity. This is the direction taken by Descriptive Complexity Theory [Imm89].

Such studies began with the work of Fagin who provided a logical characterization of \text{NP} [Fag74]. A language L is in the class \text{NP} if and only if it is definable by an existential second order sentence, i.e., if and only if there is a vocabulary \sigma and a formula \phi(S) with predicate symbols amongst those in \sigma \cup S such that

\[ e(A) \in L \iff A \models (\exists S)\phi(S), \]

where A is a finite structure over the vocabulary \sigma and e(A) is an encoding of the structure A as a string.

Since this characterization of \text{NP}, researchers have provided logical characterization of many complexity
The purpose of this paper is to propose a logical characterization for the class \( \#P \) of counting problems and study the expressiveness and feasibility of syntax-restricted subclasses naturally obtained in this setting of logical definability. Valiant [Val79] defined \( \#P \) to be the class of counting problems with an associated counting function on the input space. The counting function applied to an input is the number of accepting paths of an NP machine on that input. The class contains various natural and interesting counting problems such as \( \#SAT \) (given a propositional formula, count the number of satisfying assignments), \( \#HAMiltonian \) (given an undirected graph, count the number of Hamiltonian cycles), and \( \#EXT \) (given a poset, count the number of linear extensions). Since counting the number of solutions is at least as hard as checking if there is a solution, \( \#P \) contains NP-hard problems. The hardness of problems in \( \#P \) is further demonstrated by Toda's result [Tod89], which says that every language in Polynomial Hierarchy [Sto76] can be recognized in deterministic polynomial time using at most one query to an oracle in \( \#P \). Seeing the apparent difficulty of computing \( \#P \) functions, as well as their significance, researchers have used randomization to approximate some of these problems in polynomial time. In this direction, Karp and Luby [KL83] studied the notion of randomized approximability (see Definition 4.1) in polynomial time and showed that the \( \#DNF \) problem has a fully polynomial time randomized approximation scheme (FPTRAS).

There have been other approaches too for approximating counting problems in \( \#P \), which are of theoretical interest, e.g., enumerative counting [CH89], randomized approximation using NP oracles [VV86], deterministic polynomial time approximation using oracles in Polynomial Hierarchy [Sto83].

Though \( \#P \) problems are well studied using the model of Turing machines and polynomial time reductions, there are certain anomalies which cannot be explained well in the setting of computational models. For example, both \( \#DNF \), \( \#CNF \) (the value of the counting function is the number of satisfying assignments of a DNF, respectively CNF, propositional formula) are complete for \( \#P \) under polynomial time Turing reductions; the former has a polynomial time randomized approximation algorithm [KL83], whereas the latter is not known to have such an algorithm. The intuitive support for this difference which one gets from the fact that the decision version of \( \#CNF \) is NP-complete while that of \( \#DNF \) is in \( P \), is not enough because there are other counting problems (like counting the number of perfect matchings in a graph), whose decision versions are in \( P \), but for which there is no known polynomial time randomized approximation algorithm.

Motivated by Fagin's characterization of \( NP \), we examine the class of all problems \( \mathcal{Q} \), with an associated counting function \( f_\mathcal{Q} \), definable using first order formulae \( \phi(z,T) \) as

\[
f_\mathcal{Q}(A) = |\{ (z,T) : A \models \phi(z,T) \}|,
\]

where \( A \) is an ordered finite structure, \( T \) is a sequence of predicate variables and \( z \) is a sequence of first order variables. Throughout this paper we shall deal with ordered finite structures as inputs to the
counting problems. These are finite structures along with a binary relation, which is always interpreted as a total order on the universe of the structure.

We show that on ordered finite structures the above framework captures exactly the class \#P. In fact, \#P, the syntactic subclass of the above class obtained by using only \Pi_2 formulae is \#P. We study subclasses of \#P obtained by restricting the quantifier complexity of the first order formulae involved. We obtain the classes \#Sigma_0, \#Sigma_1, \#P, \#Sigma_2 using \Sigma_0, \Sigma_1, \Pi_1, \Sigma_2 formulae respectively.

In general, the set of properties definable by \Sigma_1 and \Pi_1 sentences are incomparable, but in this framework, it turns out that the class \#Sigma_1 \subseteq \#P. As a result we have an hierarchy of classes

\[ \#P_0 \subseteq \#Sigma_1 \subseteq \#Pi_1 \subseteq \#Sigma_2 \subseteq \#Pi_2 = \#P. \]

Unlike the hierarchies of classes found in computational complexity theory, we prove that this is a true hierarchy, i.e., the five classes are distinct. These proofs are model theoretic in nature and do not take recourse to any complexity theoretic assumptions.

Next, we study the computational complexity of counting problems in these classes and get two positive results. We show that all the counting problems in \#Sigma_0 are computable in deterministic polynomial time. This result seems levelwise optimal in the above hierarchy, i.e., it is not likely to be true for the levels above \#Sigma_0 because the very next level \#Sigma_1 has \#P-complete problems. Next we show that all the functions in \#Sigma_1 are product reducible (see definition 4.2) to restricted versions of \#DNF problem. As a corollary, we show that every problem in \#Sigma_1 has a fully polynomial time randomized approximation scheme (FPTRAS). In fact, the FPTRAS for a \#Sigma_1 problem is very simple and does not require the full power of the method of Karp and Luby [KL83]. Once again, this result is not likely to hold upwards in the hierarchy because the next level \#Pi_1 contains the \#3CNF problem which cannot have a polynomial time randomized approximation algorithm unless \text{NP} = \text{RP}. Nevertheless, we isolate a syntax-defined subclass \#R.\Sigma_2 of \#Sigma_2, which contains the \#DNF problem and all the functions in it are product reducible to \#DNF and hence have a FPTRAS.

The computational characteristics of the classes \#Sigma_0 and \#Sigma_1 are not likely to hold for the classes \#Pi_1 and \#Sigma_2 in the above logical hierarchy. We consider a method of avoiding this situation by taking closure of these classes under product reductions. We show that the functions in \#P form a 3-level hierarchy of closure classes in which the first two levels have counting functions with "feasible" computational properties, while the third is the whole of \#P. By "feasible" we mean either polynomial time solvable by approximable by a polynomial time randomized approximation algorithm.

Though \#Sigma_0 (and \#Sigma_1) capture counting problems in \#P which are computable in deterministic polynomial time (respectively, approximable by a randomized polynomial time algorithm), they cannot capture all such problems. There are polynomial time computable counting problems which are not in the class \#Sigma_0 and \#Sigma_1 (see the proof of Theorem 2). This prompts the following question. Given a counting problem specified in this framework, is the problem polynomial time computable, or does it have a polynomial time randomized approximation algorithm? We show that under reasonable complexity theoretic assumptions, such questions are undecidable. For example, assuming \text{P} \neq \text{P}^\text{NP}, it is an undecidable problem to determine for a given first order formula \phi, whether the associated counting function is polynomial time computable or not. Similar results follow for the existence of polynomial time randomized approximation algorithms.

The organization of the paper is as follows. In Section 2, we give the logical characterization of \#P and in Section 3 we show that functions in \#P form a logical hierarchy with 5 distinct levels. In Section 4, we show that the lower two levels of the hierarchy capture functions in \#P which are polynomial time computable and approximable by polynomial time randomized algorithms respectively and also prove the undecidability results. In Section 5, we study the class \#R.\Sigma_2 and the hierarchy of closure classes. In Section 6, we conclude with discussion of some open problems.

2 Descriptive Characterization of \#P

In this section, we introduce a logic based framework for expressing counting problems and show that the class of functions definable in this framework is
Definition 2.1: A Counting Problem is a tuple \( Q = (I_Q, F_Q, f_Q) \) such that

- \( I_Q \) is the set of input instances. \( I_Q \) is recognizable in polynomial time.

- \( F_Q(I) \) is a set of feasible solutions for the input \( I \in I_Q \).

- \( f_Q : I_Q \rightarrow \mathbb{N} \) is a counting function, corresponding to the problem \( Q \), and \( f_Q(I) = |F_Q(I)| \).

A counting Problem \( Q \) is in \#P if there is an NP machine for which the number of accepting paths on input \( I \) is given by \( f_Q(I) \).

For this work, we shall assume that the instance space of a counting problem is a set of ordered finite structures over a certain vocabulary \( \sigma \). Graphs are naturally represented as finite structures using a vocabulary with a single binary relation. Counting problems having inputs other than graphs can be represented as finite structures using an appropriate vocabulary.

As mentioned above, the inputs are ordered finite structures, i.e., the finite structures are provided with a built-in total order on the elements of the universe. Henceforth we shall denote ordered finite structures by \( A \), their universe by \( A \), and the total order by \( \preceq \). We shall always assume \( A \) to be the set \( \{1, \ldots, n\} \), with the natural ordering on the elements, unless stated otherwise. We shall also use the notation \( x < y \) to mean \( (x \preceq y) \wedge (x \neq y) \). We shall also use \( S \) to denote a finite sequence of predicate symbols, and \( w, x, y, z \) to denote to a finite sequence of first order variables.

Definition 2.2: Let \( \sigma \) be a vocabulary containing a relation symbol \( \preceq \). Let \( Q \) be a counting problem, with finite structures \( A \) over \( \sigma \) as instances. The relation \( \preceq \) is interpreted as a total order on the elements of the universe of \( A \). Let \( T = (T_1, \ldots, T_r) \), \( r \geq 0 \) be a sequence of predicate symbols and let \( z = (z_1, \ldots, z_m) \), \( m \geq 0 \) be a sequence of first order variables such that at least one of the sequences is of non zero length, i.e., \( m + r > 0 \). We say \( Q \) is in the class \#\( \mathcal{P} \) if there is a first order formula \( \phi_Q(z, T) \) over \( \sigma \cup T \), such that

\[
f_Q(A) = |(T, z) : A \models \phi_Q(T, z)|.
\]

We define subclasses \#\( \Pi_n \), \#\( \Sigma_n \), \( n \geq 0 \) analogously using \( \Sigma_n \), \( \Pi_n \) formulae respectively instead of arbitrary first order formulae. Further, for a counting problem in \#\( \mathcal{F} \), we refer to the following problem as the associated decision problem: Given an input structure \( A \), is there an assignment to \( (T, z) \) so that \( A \models \phi_Q(T, z) \)?

Some examples of problems in these classes are as follows.

- \#3CLIQUE: Given a graph \( G = (V, E) \), the counting function is the number of triangles in the graph.

\[
f_{\#3\text{CLIQUE}}(G) = |\{(z_1, z_2, z_3) : G \models z_1 < z_2 \wedge z_2 < z_3 \wedge \forall z \in \mathbb{Z}, E(z_1, z_2) \wedge E(z_2, z_3) \wedge E(z_3, z_1)\}|
\]

From the above formula it is clear that \#3CLIQUE is in \#\( \Sigma_0 \).

- \#3DNF: Given a boolean formula in disjunctive normal form with 3 literals per disjunct, the counting function is the number of satisfying assignments. An instance \( I \) of a 3DNF formula is encoded as a finite structure \( A(I) \) using a vocabulary \( \{D_0, D_1, D_2, D_3\} \), where \( D_i, i = 0, \ldots, 3 \), are ternary relations. For \( i = 0, \ldots, 3 \), \( D_i(x_1, x_2, x_3) \) if and only if \( \{\neg x_1, \ldots, \neg x_i, x_{i+1}, \ldots, x_3\} \) appear as a disjunct in the 3DNF formula (cf. [KT90]).

\[
f_{\#3\text{DNF}}(A(I)) = |\{T : A(I) \models (\exists x_1)(\exists x_2)(\exists x_3) (D_0(x_1, x_2, x_3) \wedge T(x_1) \wedge T(x_2) \wedge T(x_3)) \vee (D_1(x_1, x_2, x_3) \wedge \neg T(x_1) \wedge T(x_2) \wedge T(x_3)) \vee (D_2(x_1, x_2, x_3) \wedge \neg T(x_1) \wedge \neg T(x_2) \wedge T(x_3)) \vee (D_3(x_1, x_2, x_3) \wedge \neg T(x_1) \wedge \neg T(x_2) \wedge \neg T(x_3))\}|
\]

This shows that \#3DNF is in the class \#\( \Sigma_1 \).

- \#3CNF: Given a boolean formula in conjunctive normal form with three literals per clause, the counting function is the number of satisfying assignments. An instance \( I \) of a 3CNF formula \( A(I) \) is encoded as a finite structure \( A(I) \) using a vocabulary \( \{C_0, C_1, C_2, C_3\} \) where \( C_i, i = 0, \ldots, 3 \), are ternary relations. \( C_i(x_1, x_2, x_3) \) if and only if \( \{\neg x_1, \ldots, \neg x_i, x_{i+1}, \ldots, x_3\} \) appear as a clause in the 3CNF formula \( I \) (cf. [KT90]). Using such a vocabulary, it can be seen that \#3CNF is the class \#\( \Pi_1 \).
• #CNF: Given a boolean formula in conjunctive normal form, the counting function is the number of satisfying assignments. We use the vocabulary \{C, P, N\}, with a unary relation C and two binary relations P, N, to encode a CNF formula \(I\) as a finite structure \(A(I)\). The structure \(A(I)\) has universe \(A = C \cup V\), where \(V\) is the set of variables and \(C\) is the set of clauses of \(I\). The predicate \(P(c, v)(N(c, v))\) expresses the fact that clause \(c\) contains variable \(v\) positively (respectively, negatively). Let \(S\) denote the set of variables set to true in a satisfying truth assignment to the instance \(I\). Under this encoding,

\[f_{\#CNF}(A(I)) = | \{ (T, z) : A(I) = \langle \forall c \exists v (P(c, v) \land S(v)) \lor (N(c, v) \land \neg S(v)) \rangle \} |.\]

Hence, #CNF is in the class \#P.

• #DNF: The counting function for this problem is the number of satisfying assignments for a given DNF formula. Using a vocabulary very similar to the one given above, it is easy to show that #DNF is in the class \#P.

We now prove the following theorem which gives a descriptive characterization of the class #P.

**Theorem 1:** The class #P coincides with the class #\(\mathcal{FO}\). In fact, the class #P is the class #\(\Pi_2\). Hence, #P = #\(\Pi_2\) = #\(\Pi_1\) \(\subseteq\) #\(\Sigma_2\), \(n > 2\).

**Proof:** It is clear that if \(Q\) is a counting problem in #\(\mathcal{FO}\), then \(Q\) is a problem in #P. For an input structure \(A\), the witnessing NP machine has to guess a tuple \((T, z)\) and verify in polynomial time that \(A \models Q(T, z)\).

To prove the other direction, assume \(Q\) is a problem in the class #\(P\), the instances of \(Q\) being finite structures \(A\) over some vocabulary \(\sigma\), which includes the relation symbol \(\preceq\). There is an NP machine for \(Q\) such that the number of accepting paths of the machine on input an encoding of \(A\) is given by \(f_Q(A)\). Hence, to check if \(f_Q(A)\) is nonzero is a problem in NP. By Fagin's characterization of NP [Fag74] in terms of definability in existential second order logic, there is a first order formula \(\phi(T)\) with relation symbols from \(\sigma\) and the sequence \(T\) of predicate symbols, such that \(A\) is in the NP language, i.e., \(f_Q(A) \neq 0\), if and only if \(A \models (\exists T)\phi(T)\). It is straightforward to verify from the proof of Fagin's result that if \(\preceq\) is a built-in total order on the elements of the universe \(A\), the formula \(\phi\) can be chosen to be a \(\Pi_2\) formula involving the binary relation symbol \(\preceq\). The formula \(\phi\) is such that, every accepting computation of the NP machine, on input an encoding of \(A\), corresponds to a unique value of the sequence \(T\) which satisfies \(\phi(T)\). In other words, the number of accepting computations of the NP machine is exactly equal to \(| \{(T) : A \models \phi(T)\} |\). Hence \(Q\) is in the class #\(\mathcal{FO}\).

Therefore #P = #\(\mathcal{FO}\) = #\(\Pi_2\). \(\Box\)

It should be noted at this point that the built-in total order \(\preceq\) is crucial to the proof of the above theorem, though it is not required in Fagin's characterization of the class NP. For every finite (unordered) structure \(A\) in an NP language there is an existential second order formula \(\phi(S)\) that \(A\) satisfies. In the absence of a built-in order, one can quantify out (using an existential quantifier) a binary relation \(\leq\) and assert, as a subformula of \(\phi\) that, \(\leq\) represents a total order on the universe. However any total order will work in satisfying the formula. In such a case, the number of distinct existential quantifiers \(S\), which satisfy \(\phi(S)\) includes \(|A|\) possible binary relations, and hence the number of distinct existential quantifiers \(S\) which satisfy \(\phi(S)\) is \(|A|\) times the number of accepting paths in the corresponding NP machine. This problem disappears when we use a unique built-in order \(\preceq\).

The above proof shows that counting the assignments to a sequence \(T\) of predicate variables gives the value of the function \(f_Q\), but we have considered a more generalized format in definition 2.2 which counts the assignments to \((T, z)\) i.e., a sequence of second order variables and a sequence of first order variables. This is done because, as a number of examples show, there are counting functions in #P, which are more naturally expressible by counting the assignments of just first order variables or a combination of first and second order variables.

### 3 Logical Hierarchy in #P

Having shown that #\(\Pi_2\) captures the class #P, we now study the subclasses of #\(\Pi_2\) that are obtained by restricting the quantifier complexity of the first order formulae. From the definition of the subclasses, #\(\Sigma_0 = \#\Pi_0\), #\(\Sigma_1 = \#\Pi_1\), and #\(\Sigma_2\), one would expect that the containment between the
classes is as follows:

\[ \#\Sigma_0 = \#\Pi_0 \subseteq \#\Sigma_1 \subseteq \#\Pi_1 \subseteq \#\Pi_2 = \#P \]

Contrary to such expectations, we show in this section, that the problems in \#\Pi_2 form a linear hierarchy with five distinct levels. We state a few more examples of #P problems which will be useful to us in the sequel.

- #DIST2: Given a graph, the counting function is the number of unordered pairs of vertices \( \{x, y\} \) in \( G \) such that the shortest path from \( x \) to \( y \) is of length 2. It can be easily seen that this problem is in the class #\Sigma_1.

- #DEG-LNGB: Given a graph, the counting function is the number of vertices which have a neighbor of degree 1. It is easy to see that this problem is in the class #\Sigma_2.

- #HAMILTONIAN: Given a graph, the counting function is the number of Hamiltonian cycles in the graph. It follows from Theorem 1 that this problem is in the class #\Pi_2.

Theorem 2: The class #\Sigma_1 is contained in the class #\Pi_1. As a result,

\[ \#\Sigma_0 = \#\Pi_0 \subseteq \#\Sigma_1 \subseteq \#\Pi_1 \subseteq \#\Sigma_2 \subseteq \#\Pi_2 = \#P. \]

Moreover, these containments are strict.

Proof: We give this proof in five parts.

Part 1: We show here that #\Sigma_1 is contained in #\Pi_1. Let \( Q \) be a counting problem in #\Sigma_1 with \( f_Q(A) = |\{(T, z) : A \models \exists x \psi(x, z, T)\}| \), where \( \psi(x, z, T) \) is quantifier-free. Instead of counting the tuples \( (T, z) \) as above, we count the tuples \( (T, (z, x^*)) \), where \( x^* \) is the lexicographically smallest \( x \) such that \( A \models \psi(x, z, T) \). It is clear that the latter count is also equal to \( f_Q(A) \). Let \( \theta(x^*, x) \) be a quantifier-free formula that expresses the fact that that \( x^* \) is lexicographically smaller than \( x \). Hence,

\[ f_Q(A) = \{(T, (z, x^*)) : A \models \psi(z, x^*, T) \land (\forall x)(\psi(x, z, T) \rightarrow \theta(x^*, x))\} \].

As a result \( Q \in \#\Pi_1 \) and consequently \( \#\Sigma_1 \subseteq \#\Pi_1 \).

As \( \#\Pi_2 = \#P \), it follows that \( \#\Sigma_2 \subseteq \#\Pi_2 \). It is now clear that the the functions in class #P form a hierarchy with five levels,

\[ \#\Sigma_0 = \#\Pi_0 \subseteq \#\Sigma_1 \subseteq \#\Pi_1 \subseteq \#\Sigma_2 \subseteq \#\Pi_2. \]

Part 2: We now show that #DIST2, a rather trivial problem defined earlier, is not in the class #\Sigma_0. We have already noted before that #DIST2 is in the class #\Sigma_1.

Towards a contradiction, assume #DIST2 is in the class #\Sigma_0. Therefore,

\[ f_{\#\text{DIST2}}(G) = |\{(T, z) : G \models \psi(z, T)\}| \]

where \( \psi(z, T) \) is a \( \Sigma_0 \) (quantifier-free) formula. Now consider the ordered graph \( G = (V, E) \), with \( V = \{1, 2, 3\} \) and \( E = \{(1, 2), (2, 3), (1, 3), (2, 4), (3, 4)\} \). Note that on this graph \( f_{\#\text{DIST2}}(G) = 1 \). Let \( (T^*, x^*) \), \( \mathbf{z}^* = (z^*_1, \ldots, z^*_n) \), be the only pair such that \( G \models \psi(x^*, T^*) \). Let \( Z \) denote the set \( \{z^*_1, \ldots, z^*_n\} \), \( G_Z \) denote the ordered subgraph of \( G \) induced by \( Z \), and \( T_Z^* \) denote the sequence of predicates \( T^* \) restricted to \( Z \). We discuss two cases depending on the cardinality of \( Z \).

If \( |Z| \leq 2 \), then we have that \( G_Z \models \psi(z^*, T_Z^*) \). This is because \( \Sigma_0 \) formulae are preserved under induced substructures. Therefore, in this case, we have \( f_{\#\text{DIST2}}(G_Z) \geq 1 \), which is a contradiction. Consequently, \( |Z| > 2 \).

So, \( Z = \{1, 2, 3\} \). Now, consider the graph \( G = (V, E) \), where \( V = \{1, 2, 3, 4\} \) and the edge set \( E = \{(1, 2), (2, 3), (1, 3), (2, 4), (3, 4)\} \). It is clear that \( f_{\#\text{DIST2}}(G) = 1 \). We denote by \( G_1, G_2 \), the subgraphs of \( G \), induced by \( V_1 = \{1, 2, 4\} \), \( V_2 = \{1, 3, 4\} \) respectively. Observe that \( G_1, G_2 \) are isomorphic to \( G \). Let \( (T_1^*, z_1^*), (T_2^*, z_2^*) \) be the images of \( (T^*, x^*) \) under the appropriate isomorphism for \( G_1, G_2 \). It can be seen that \( (T_1^*, z_1^*) \) and \( (T_2^*, z_2^*) \) are distinct and that \( G_i \models \psi(z_i^*, T_i^*) \), for \( i = 1, 2 \). Since \( \Sigma_0 \) formulae are preserved under extensions, we have that \( (T_1^*, z_1^*) \) satisfy \( G \models \psi(z_1^*, T_1^*) \), for \( i = 1, 2 \). Hence, we have that

\[ f_{\#\text{DIST2}}(G) = |\{(T, z) : G \models \psi(z, T)\}| \geq 2, \]

which is a contradiction. This proves that #DIST2 is not in the class #\Sigma_0.

Part 3: We indicated before that #3CNF is in the class #\Pi_1. We now show that #3CNF is not in the
of incident on them, we would have,

$$f_{\#\text{DEG-L-NGB}}(H_i) \geq 1,$$

which is untrue.

Since the arity of $x_0^*$ is $m$ and $p > 2m + 1$, it follows that

$$\left( \frac{p}{2} \right) > mp.$$

It can be seen that, there exist $u \neq v$, such that $x_u^*$ does not have any of \{a, b, c\} as a component and $x_v^*$ does not have any of \{a, b, c\} as a component.

Let $G_{u,v}$ be the subgraph of $G_u$ obtained by deleting \{a, b\} from $G_u$ and all edges incident on these vertices and let $G_{v,u}$ be the subgraph of $G_v$ obtained by deleting \{a, b\} from $G_v$ and all edges incident on these vertices. It is clear that the graphs $G_{u,v}$ and $G_{v,u}$ are identical.

Let $(T^*_u, x_u^*)$ be the substructure of $(T^*_u, x_u^*)$ induced by $G_{u,v}$, and let $(T^*_v, x_v^*)$ be the substructure of $(T^*_v, x_v^*)$ induced by $G_{v,u}$. Since $x_u^*$ has $c$, but not $c_0$ as a component, and $x_v^*$ has $c_0$ but not $c_0$ as a component, $(T^*_u, x_u^*)$ and $(T^*_v, x_v^*)$ are distinct. As universal formulae are preserved under substructures, we have that

$$G_{u,v} \models (\exists x)(\forall y)(\forall z)(T^*_u, x_u^*, T_*)$$

and

$$G_{v,u} \models (\exists x)(\forall y)(\forall z)(T^*_v, x_v^*, T_*)$$

As a result, $f_{\#\text{DEG-L-NGB}}(G_{u,v}) \geq 2$, which is a contradiction. Therefore $\#\text{DEG-L-NGB}$ is not in the class $\#\Pi_1$.

Part 5: We now show that $\#\text{HAMILTONIAN}$ is not in the class $\#\Sigma_2$. Towards a contradiction, let us assume $\#\text{HAMILTONIAN}$ is in the class $\#\Sigma_2$, i.e., there is a quantifier-free formula $\psi(x, y, z, T)$ such that

$$f_{\#\text{HAMILTONIAN}}(G) = \{(T, z) : G \models (\exists x)(\forall y)(\forall z)(T^*_u, x_u^*, T_*)\}.$$ 

Let $m$ be the arity of $z$ and $t$ be the arity of $x$.

Now consider a graph $G$ which is a cycle of $n = m + t + 1$ vertices. Since $f_{\#\text{HAMILTONIAN}}(G) = 1$, there are tuples $(T^*, x^*)$ and $x^*$ such that $G \models (\forall y)(\forall z)(T^*_u, x_u^*, T_*)$. It is now clear that there is one vertex in $G$, say $a$, such that $a$ does not appear as a component in either $x^*$ or $z^*$. Let $G_1$ be the subgraph of $G$ obtained by deleting $a$ and the two edges incident on $a$. Note that $G_1$ is not Hamiltonian, but $G_1 \models (\forall y)(\forall z)(x^*, y, z^*, T_1)$, where $T_1$ is the subset of $T^*$ obtained by deleting all occurrences of
a from $T^*$. Therefore,

$$f_{\text{Hamiltonian}}(G_1) = \{ (T, z) : G_1 \models (\exists x)(\forall y)(\psi(x, y, z, T)) \geq 1, $$

which is a contradiction. Therefore #HAMILTONIAN is not in the class $\# \Sigma_2$.

Remark 1: Using model theoretic arguments similar to those used in the above proof, the following problems can be exactly placed in the hierarchy: #CNF, #PM (counting the number of perfect matchings in bipartite graphs), and #DNF. The problems #PM and #CNF can be shown to lie in $\# \Pi_2 - \# \Sigma_2$ and the problem, #DNF can be shown to lie in $\# \Sigma_2 - \# \Pi_1$.

4 Feasibility of logically defined subclasses

One of our objectives in using logical definability to study counting problems is to obtain syntactically defined subclasses of $\# P$ which have computationally feasible properties. By computationally feasible we mean in this paper, either polynomial time computable, or approximable within a constant factor by a polynomial time randomized algorithm. In this section, we first show that every problem in the class $\# \Sigma_0$ is polynomial time computable and that every problem in the class $\# \Sigma_1$ is approximable in polynomial time by a randomized algorithm. The above classes give only sufficient conditions for polynomial time computability and randomized approximability of counting functions. We show that under reasonable complexity theoretic assumptions, it is an undecidable problem to tell if a given formula defines a polynomial time counting problem, or a problem that is approximable by a polynomial time randomized algorithm.

Theorem 3: Every counting problem in $\# \Sigma_0$ is computable in deterministic polynomial time.

Proof: Let $Q$ be a counting problem in $\# \Sigma_0$ which is defined on ordered finite structures over a vocabulary $\sigma$. Then there is a quantifier free formula $\psi_Q$ such that

$$f_Q(A) = \{ (T, z) : A \models \psi_Q(z, T) \},$$

where $A$ is a finite ordered structure over the vocabulary $\sigma$, $T$ is a sequence $(T_1, T_2, \ldots, T_i)$ of second order predicate variables of arities $a_1, a_2, \ldots, a_r$, respectively, and $z$ is an $m$-tuple $(z_1, \ldots, z_m)$.

To compute $f_Q(A)$, we count, for each $z^* \in A^m$, the number of assignments to $T$ so that $A \models \psi_Q(z^*, T)$. The number of such $z^*$ is $|A|^m$. To show that $Q$ is polynomial time computable, it suffices to show that for every $z^* \in A^m$, we can compute $f_Q'(z^*) = \{ (T : \psi(z^*, T)) \}$ in polynomial time. Then, $f_Q(A) = \Sigma_{z^* \in A} f_Q'(z^*)$.

For every $z^* \in A^m$, consider the formula $\psi_Q(z^*, T)$. The formula $\psi_Q(z^*, T)$ can also be viewed as a propositional formula with variables of the form $T_i(y_i)$, where $y_i$ are tuples of arity $a_i$, $i = 1, \ldots, r$. The total number of such variables is $\sum_{i=1}^r |A|^{a_i}$. Let $c(z^*)$ denote the number of such variables such that either the variable or its negated literal appear in the formula $\psi_Q(z^*, T)$. In other words, $c(z^*)$ is the total number of $a_i$-tuples $y_i$, for $i = 1, \ldots, r$ such that either $T_i(y_i)$ or $\neg T_i(y_i)$ appear in the formula $\psi_Q(z^*, T)$. Note that $c(z^*)$ is a constant, i.e., it does not depend on the size of the structure $A$. Hence, we can find all the truth assignments to the variables that occur in this propositional formula in constant time. Let the number of such satisfying assignments be $s(z^*)$. The number of propositional variables that do not appear in this propositional formula is $(\sum_{i=1}^r |A|^{a_i}) - c(z^*)$. It can now be seen that

$$f_Q'(z^*) = s(z^*)2^{(\sum_{i=1}^r |A|^{a_i}) - c(z^*)},$$

which can be computed in polynomial time.

Remark 2: It is unlikely that every problem in the next higher class in the hierarchy, i.e., $\# \Sigma_1$ has only polynomial time computable problems, because $\# \Sigma_1$ has $\# P$-complete problems, e.g., #3DNF.

We now study the computational properties of the class $\# \Sigma_1$. We show that every problem in $\# \Sigma_1$ has a polynomial time randomized approximation algorithm. The following definition makes precise what we mean by randomized polynomial time approximation.

Definition 4.1: A counting problem $Q$ is said to have a polynomial time randomized $(\epsilon, \delta)$ approximation algorithm, if there is a randomized algorithm $M$
and there are constants $\epsilon, \delta > 0$ such that for all inputs $I \in \mathcal{I}_Q$,

1. $Pr[|f_M(I)/f(I)| > \epsilon] < \delta$. We assume, without loss of generality, that $f(I) \neq 0$. 

2. the running time of $M$ on input $I$ is bounded by a polynomial in $|I|$.

If $\epsilon, \delta$ are part of the input and $M$ satisfies the above two conditions for every $\epsilon, \delta > 0$, then $Q$ is said to have a polynomial time randomized approximation scheme. Further, if the running time of $M$ is bounded by a polynomial in $\frac{1}{\epsilon}, \frac{1}{\delta}$, then $Q$ is said to have a fully polynomial time randomized approximation scheme (FPTRAS).

In the spirit of this paper, we shall use finite structures $A$ as inputs to the algorithms, and replace $I$ by $A$ in the above definition.

We show the existence of a FPTRAS for every problem in $\#\Sigma_1$ in two steps.

(1) Every problem in $\#\Sigma_1$ is reducible to a restricted version of $\#\text{DNF}$ problem, under a reducibility which preserves approximability.

(2) The restricted version of the $\#\text{DNF}$ has a FPTRAS.

Before we achieve step 1 above, we define the appropriate notion of reduction and briefly comment on its properties.

**Definition 4.2:** Given counting problems $Q, R$, we say $Q$ is polynomial time product reducible to $R$ (written as $Q \preceq_{pr} R$) if there are polynomial time computable functions $g, h$, $g : I_Q \to I_R, h : N \to N$, such that for every finite structure $A$, with universe $A$, which is an input to $Q$, the value of the counting function is given by $f_Q(A) = f_R(g(A)) \times h(|A|)$. If $h$ is the constant 1 function, the reduction is said to be parsimonious.

Throughout this paper, we will use product reducible to mean polynomial time product reducible.

**Proposition 1:** Given counting problems $Q, R$, if $Q \preceq_{pr} R$ and $R$ is computable in polynomial time, then $Q$ is computable in polynomial time.

**Proposition 2:** Given functions $Q, R$, if $Q \preceq_{pr} R$ and $R$ has a polynomial time randomized $(\epsilon, \delta)$ approximation algorithm, then $Q$ also has a polynomial time randomized $(\epsilon, \delta)$ approximation algorithm.

**Definition 4.3:** For any positive constant $k$, the counting problem $\#k\cdot\text{logDNF}(\#k\cdot\text{logCNF})$ is the problem of counting the number of satisfying assignments to a propositional formula in disjunctive normal form (respectively, conjunctive normal form) in which the number of literals in each disjunct (respectively, conjunct) is at most $k\log(n)$ where $n$ is the number of propositional variables in the formula.

**Lemma 1:** For every counting problem $Q \in \#\Sigma_1$, there is positive constant $k$ so that $Q \preceq_{pr} \#k\cdot\text{logDNF}$ problem.

**Proof:** Let $Q$ be a problem in $\#\Sigma_1$ such that $f_Q(A) = |\{(T, z) : A \models (3y)(\psi(y, z, T))\}|$, where $\psi(y, z, T)$ is a quantifier-free formula in DNF form, with at most $t$ literals per disjunct, and $T$ is a sequence $(T_1, T_2, \ldots, T_r)$ of second order predicate variables whose arities are $a_1, a_2, \ldots, a_r$ respectively. Let $p$ denote the arity of $y$ and let $m$ denote the arity of $z$. Let $\{y_1, \ldots, y_{|A|^p}\}$ be the set $A^p$, and let $\{z_0, \ldots, z_{|A|^m-1}\}$ be the set $A^m$.

For every $z_i \in A^m$, we write the formula $(\exists y)\psi(y, z_i, T)$ as a disjunction $\bigvee_{j=1}^{A^p} \psi(y_j, z_i, T)$. Let $\psi'(z_i, T)$ denote the formula obtained from $\bigvee_{j=1}^{A^p} \psi(y_j, z_i, T)$ by replacing every subformula that is satisfied by $A$ by the value TRUE and every subformula that is not satisfied by $A$ by FALSE. It is clear that $\psi'(z_i, T)$ is a propositional formula in DNF with variables of the form $T_i(w_i)$, with $w_i \in A^{a_i}$ and $1 \leq i \leq r$.

We define new propositional variables $z_1, z_2, \ldots, z_l$ where $2^{l-1} < |A|^m \leq 2^l$. For every $s \in \{0, 1\}^l$, let $x(s)$ represent the conjunction of these $l$ variables so that for $1 \leq i \leq l$, $z_i$ appears complemented in $x(s)$ if and only if the $i$-th component of $s$ is 0. We shall interpret $s$ as the binary representation of an integer and use the integer and the sequence $s$ interchangeably.

Consider now the propositional formula $\theta_A$ defined as

$$\theta_A \overset{def}{=} [\psi'(z_0, T) \land x(0)] \lor [\psi'(z_1, T) \land x(1)] \lor \cdots \lor [\psi'(z_{|A|^m-1}, T) \land x(|A|^m - 1)],$$
with one term for each \( z_i \in A^m \). This is a DNF formula with variables \( x_1, x_2, \ldots, x_t \) and \( T_i(w_i) \), with \( w_i \in A^m \). Let \( c(A) \) be the number of variables among these that do not appear in \( \theta_A \). It is straightforward to check that

(i) \( \theta_A \) is a propositional formula in disjunctive normal form with at most \( t + l \) literals per disjunct.
Since \( l = O(\log(n)) \), \( \theta_A \) is a \( k \)-logDNF formula for suitable \( k \) depending on the size of \( \psi(T, x) \).

(ii) \( f(A) = 2^{c(A)} \times \) (the number of satisfying assignments of \( \theta_A \)).

Finally, note that we can construct \( \theta_A \) in polynomial time.

Remark 3: By a similar proof, it can also be shown that every counting problem in \( \#\Pi_1 \) is product reducible to \( \#k\text{-logCNF} \).

Remark 4: Let \( Q \) be a counting problem in \( \#\Sigma_1 \). The decision version of the counting problem \( Q \) is:

Given a finite structure \( A \), is \( f_Q(A) \neq 0 ? \) This has a YES answer if and only if \( A \models (\exists T)(\exists x)\psi_Q(T, x) \), where \( \psi_Q \) is a quantifier-free formula. We know from definability theory that this is in \( P \). Hence it follows that the decision version of every \( \#\Sigma_1 \) problem is polynomial time computable.

Lemma 2: For every \( k \), there is a fully polynomial time randomized approximation scheme for the \( \#k\text{-logDNF} \) problem.

The above lemma is a special case of the result of Karp and Luby [KL83], who give a FPTRAS for the general \( \#\text{DNF} \) problem. However, as our proof below shows, there is a simple FPTRAS which works for the \( \#k\text{-logDNF} \) problem (for every \( k \)) and does not need the full power of the method of Karp and Luby. Before giving the proof, we recall a result on a Chernoff-type bound for random variables.

Lemma 3: Let \( x \) be a random variable taking values in the interval \([0,1]\) with the expected value \( p < 0.5 \). Let \( x_1, x_2, \ldots, x_t \) be \( t \) random variables with the same distribution as \( x \). Then for every \( \epsilon > 0 \), \( \Pr\left[ \left| \frac{1}{t}(x_1 + x_2 + \ldots + x_t) - p \right| > \epsilon p \right] < 2 \exp\left(-\frac{(\epsilon \sqrt{2p(1-p)} \epsilon^2)}{(2(1-p)^2)}\right) \).

The proof of this lemma follows from Theorem 2, pp. 41, of Rényi [Rény70] and was used by Karp and Luby [KL83] in their analysis on a FPTRAS for the \( \#\text{DNF} \) problem.

Proof: [of Lemma 2] Let \( d \) be a DNF formula so that there are at most \( k \log(n) \) literals per disjunct where \( n \) is the number of propositional variables occurring in the formula. If \( d \) is satisfiable, (which can be checked in polynomial time), then consider any disjunct which is satisfiable. Observe that this disjunct is satisfiable for at least \( \frac{1}{2^n} \) fraction of all the satisfying assignments. Therefore if an assignment is chosen randomly from the uniform distribution over all the assignments, the probability that it is a satisfying assignment of \( d \) is at least \( \frac{1}{n} \).

Define a random variable \( x \) as follows. Choose a random vector \( v \) from the uniform distribution on \([0,1]^m\). If \( v \) represents a satisfying assignment of \( d \), set \( x := X + 1 \) else set \( x := X \).

Output \( 2^{\frac{x}{t}} \).

Using Lemma 3, \( \Pr\left[ \left| \frac{X}{t} - p \right| > \epsilon p \right] < 2 \exp\left(-\frac{2\epsilon^2 p(1-p)\epsilon^2}{(2(1-p)^2)}\right) \).

If we choose the number of trials \( t \) to be \( \frac{2(1-p)}{\epsilon^2 p(1-p)} \log\left(\frac{2}{\delta}\right) \), then the above probability is less than \( \delta \). Note that \( t \) is bounded by a polynomial in \( n, \frac{1}{p}, \log\left(\frac{1}{\delta}\right) \).}

The next Theorem now follows from Lemmas 1 and 2.

Theorem 4: Every counting function in \( \#\Sigma_1 \) has a FPTRAS.

Remark 5: It is unlikely that every problem in the next higher class in the hierarchy, viz., \( \#\Pi_1 \) has a polynomial time randomized approximation algorithm. Such a result would imply \( \text{NP} = \text{RP} \), since \( \#3\text{CNF} \) lies in the class \( \#\Pi_1 \) and its decision version 3SAT is \( \text{NP} \)-complete.

The class \( \#\Sigma_1 \) has a lot of interesting problems like, \( \#\text{MONOCHROMATIC-k-CLIQUE-PARTITIONS}, \#\text{NON-VERTEX-COVERS}, \#\text{NON-CLIQUES} \), which
are defined in the appendix. It follows from Theorem 4 that all these problems have a FPTRAS.

We have seen that the syntax of a first order formula used in defining the \#P problems has some impact on the computational complexity of the corresponding counting function. A natural question to ask in this context is: Given an arbitrary first order formula \( \phi(T, z) \), is the counting problem defined using this formula polynomial time computable, or is the problem approximable by a polynomial time \((\epsilon, \delta)\) randomized algorithm? We show below that these are undecidable problems.

Before we state and prove the theorem, we need the following definitions of relativized formulae from logic.

**Definition 4.4:** If \( \varphi \) is a first order formula and \( R \) is a unary predicate, then the relativized formula \( \varphi^R \) is obtained from \( \varphi \) by replacing every universal subformula \((\forall x)\varphi(x)\) of \( \varphi \) by \((\forall x)(R(x) \rightarrow \varphi(x))\) and by replacing every existential subformula \((\exists x)\varphi(x)\) of \( \varphi \) by \((\exists x)(\neg R(x) \land \varphi(x))\).

Let \( \sigma \) be a vocabulary and let \( \phi(z, T) \) be a first order formula with predicate symbols from \( \sigma \cup T \).

We say a counting problem \( Q_\phi \) with instances finite structures over \( \sigma \), is defined by the formula \( \phi(z, T) \), if the counting function \( f_Q \) is defined using the formula \( \phi \) as follows:

\[
 f_Q(A) = | \{ (T, z) : A \models \phi(T) \} | 
\]

We also write \( f_\phi \) for this function.

**Theorem 5:** Let \( \sigma \) be a vocabulary with a unary predicate \( \{C\} \) and three binary predicate symbols \( \{E, P, N\} \).

(a) Assuming \( \text{NP} \neq \text{RP} \), the following is an undecidable problem: Given a first order sentence \( \phi(z, T) \) over \( \sigma \cup T \), does the counting problem \( Q_\phi \) have a polynomial time \((\epsilon, \delta)\) randomized approximation algorithm, for some constants \( \epsilon, \delta > 0 \)?

(b) Similarly, assuming \( \text{P} \neq \text{P}^{\#P} \), the following is an undecidable problem: Given a first order sentence \( \phi(z, T) \) over \( \sigma \cup T \), is the counting problem \( Q_\phi \) polynomial time computable?

**Proof:** (a) The proof is similar to that given in [KT91] to show an analogous result for NP optimization problems. We will use Trakhtenbrot's theorem which asserts that for any vocabulary \( \tau \) with at least one binary symbol, the set of first order sentences true on all finite structures over \( \tau \) is not recursive. Assuming \( \text{NP} \neq \text{RP} \), we'll give a reduction from the above problem to the problem of deciding "approximability" thereby showing the undecidability of the latter.

Consider the \#CNF problem and assume that the instances of CNF formula are encoded as finite structures \( A \) over vocabulary \( \{C, P, N\} \) (see Section 2), with universe \( A \). Let

\[
 f_{\#\text{CNF}}(A) = | \{ (T) : A \models \phi(T) \} | 
\]

Given an arbitrary first order formula \( \psi \) over vocabulary \( \{E\} \), define the first order \( F_\psi(T) \) as:

\[
 F_\psi(T) \triangleq \neg \psi^V \land \phi^V(T) \land (T \subseteq V') \text{ where } V' \text{ is the complement of } V. 
\]

Let \( Q_{F_\psi} \) be the problem defined by \( F_\psi \), which has as inputs, finite structures of the form \( B = (B, V, C, E, P, N) \), with the universe \( B \). Consider \( f_{F_\psi} \), the counting function of \( Q_{F_\psi} \) defined as,

\[
 f_{F_\psi}(B) = | \{ (T) : B \models F_\psi(T) \} | . 
\]

Observe that, if \( \psi \) is true over all finite structures, then \( F_\psi(T) \) is false over all finite structures and the counting function \( f_{F_\psi} \) has answer 0 on all structures \( B \). Hence the problem \( Q_{F_\psi} \) is trivially approximable.

On the other hand, if \( \psi \) is false on some finite structure, say \((V, E)\), with universe \( V \), then there is a polynomial time parsimonious reduction from \#CNF to the problem \( Q_{F_\psi} \). Therefore assuming \( \text{NP} \neq \text{RP} \), \#CNF, and hence \( Q_{F_\psi} \), is not approximable by any randomized \( \epsilon, \delta \) approximation algorithm. The parsimonious reduction from \#CNF to \( Q_{F_\psi} \) is as follows: Given an instance of CNF formula represented as \( A = (A, C, P, N) \), with universe \( A \), construct the structure \( B = (V \cup A, V, C, E, P, N) \).

It can be checked easily that

\[
 f_{\#\text{CNF}}(A) = f_{F_\psi}(B). 
\]

Hence \( Q_{F_\psi} \) is approximable by an polynomial time \((\epsilon, \delta)\) randomized approximation algorithm for some \( \epsilon, \delta \), if and only if \( \psi \) is true on all finite structures.

(b) The proof is similar to that of (a). We omit the details. \( \blacksquare \)
5 Feasibility beyond \( \#\Sigma_1 \)

We would like to see whether the classes \( \#\Pi_1 \) and \( \#\Sigma_2 \) also capture some aspect of the computational complexity of the counting problems in them, i.e., are functions in \( \#\Pi_1 \) (respectively, \( \#\Sigma_2 \)) easier under some reasonable measure of computational complexity than those which are not in the class? We do not know of a direct answer to this question.

One intuitive reason for the difficulty in answering this question is that \( \#\Pi_1 \) (and hence \( \#\Sigma_2 \)) has counting functions which are complete for the class \( \#P \) under parsimonious reductions, e.g., \( \#3CNF \).

Therefore any reasonably defined complexity class which contains \( \#3CNF \), contains all of \( \#P \).

5.1 The class \( \#R \Sigma_2 \)

As one approach, we turn our attention to restricting the quantifier-free part of the first order formulae to study the classes we may obtain, and their computational properties. We isolate a syntactic subclass of \( \#\Sigma_2 \) that has complete problems and every problem in this class has a FPTRAS.

**Definition 5.1:** Let \( \sigma \) be a vocabulary and \( Q \) be a counting problem, with finite ordered structures \( A \) over \( \sigma \) as instances. We say \( Q \) is in the class \( \#R \Sigma_2 \) if there is a quantifier-free first order formula \( \psi_Q (z, T) \) over \( \sigma \cup T \), \( z \) being the sequence of free first order variables in the formula, such that

\[
f_Q (A) = | \{(T, z) : A \models \exists x y \psi_Q (T, z, x, y)\} |,
\]

where \( \psi_Q (T, z, x, y) \) is a quantifier-free formula and when \( \psi_Q \) is expressed by an equivalent formula in conjunctive normal form, each conjunct has at most one occurrence of a predicate symbol from \( T \).

**Lemma 4:** \( \# \)DNF is complete for \( \#R \Sigma_2 \) under product reductions.

**Proof:** To see that \( \# \)DNF is in \( \#R \Sigma_2 \), it is easy to observe that the \( \Sigma_2 \) formula which defines the \( \# \)DNF problem has the required restrictions. To show that \( \# \)DNF is hard for the class \( \#R \Sigma_2 \), consider a counting function \( Q \) in \( \#R \Sigma_2 \) which is expressible as \( f_Q (A) = | \{(T, z) : A \models (\exists x)(\forall y) \psi_Q (T, z, x, y)\} | \), where \( \psi_Q (T, z, x, y) \) is a quantifier-free formula in conjunctive normal form, \( T \) is a sequence \((T_1, T_2, \ldots, T_r)\) of second order predicate variables, and \( z \) is an \( m \)-tuple \((x_1, \ldots, x_m)\). Let the arity of \( x \) be \( p_1 \) and the arity of \( y \) be \( q_1 \). Let \( |A|^p = p_1, |A|^q = q_1 \). Let \( A^{n_1} = \{x_1, x_2, \ldots, x_p\} \), and let \( A^{n_2} = \{y_1, y_2, \ldots, y_q\} \).

It can be seen that \( A \models (\exists x)(\forall y) \psi_Q (T, z, x, y) \) if and only if \( A \models \bigwedge_{i=1}^q \bigvee_{j=1}^p \psi_{Q,i,j} (T, z) \), where \( \psi_{Q,i,j} (T, z) \) is obtained from \( \psi_Q (T, z, x_i, y_j) \) by replacing every subformula of \( \psi_Q (T, z, x_i, y_j) \) that is true in \( A \) by the logical value TRUE and every subformula of \( \psi_Q (T, z, x_i, y_j) \) that is false in \( A \) by the logical value FALSE. In this way, we obtain a DNF formula, say \( \theta_A \), with propositional variables of the form \( T_i(w_i) \) where \( w_i \in A^{n_1} \). This uses the fact that the formula \( \psi_Q (T, z, x_i, y_j) \) has only one occurrence of any predicate from the sequence of predicates \( T \) per conjunct.

Let \( e_A \) be the number of propositional variables amongst those above, which do not occur in \( \theta_A \). It can be easily verified that the value of \( f_Q (A) \) is exactly \( 2^e_A \times \text{the number of satisfying assignments of } \theta_A \). To conclude, note that the above reduction from \( A \) to \( \theta_A \) is computable in polynomial time.

**Theorem 6:** (a) Every counting problem in \( \#R \Sigma_2 \) has a FPTRAS.
(b) The decision version of every counting problem in \( \#R \Sigma_2 \) is in \( P \).

**Proof:** (a) Using Proposition 2 and Lemma 4, the theorem follows from the result of Karp and Luby [KL83] which says that \( \# \)DNF has a FPTRAS.
(b) The proof of this part follows from Proposition 1 and Lemma 4.

These results are similar in flavor to that obtained by Kolaitis and Thakur [KT91] in the context of optimization problems. They isolated a class \( \text{MIN } F^+ \Pi_2(1) \) of NP optimization problems by restricting the quantifier-free part of \( \Pi_2 \) formulae which had \( \text{MIN SET COVER} \) as a complete problem, via an approximation preserving reduction.

Examples of other problems in \( \#R \Sigma_2 \) are \#NON-HITTING-SETS, \#NON-DOMINATING-SETS, and \#NON-EDGE-DOMINATING-SETS, which are defined in the appendix. By Theorem 6 these problems have a FPTRAS.
5.2 Closure classes

Another approach to obtain classes of counting problems with feasible computational properties is to consider the closure of the classes \(\#S_0\) and \(\#S_1\) under product reductions. Then, the functions in \(\#P\) form a hierarchy with three levels.

**Definition 5.2:** Given a class \(C\) of counting problems, the closure class \(Ppr(C)\) is the class of counting problems which are product reducible to some problem in \(C\). In particular, we are interested in the closure classes \(Ppr(\#S_0), Ppr(\#S_1),\) and \(Ppr(\#\Pi_1)\).

The following theorem states the computational properties of the closure classes of interest.

**Theorem 7:**
(a) Every counting problem in the class \(Ppr(\#S_0)\) is computable in polynomial time. In fact, \(Ppr(\#S_0)\) is exactly the class of polynomially computable counting functions.
(b) Every problem in \(Ppr(\#S_1)\) has a FPTRAS.
(c) The decision version of every problem in \(Ppr(\#S_1)\) is in \(P\).

**Proof:**
(a) The forward direction follows from Theorem 3 and Proposition 1. Converse follows trivially.
(b) The proof follows from Theorem 4 and Proposition 2.
(c) The proof follows from the definition of product reduction.

We now study the relationship between the closure classes \(Ppr(\#S_0), Ppr(\#S_1),\) and \(Ppr(\#\Pi_1)\). The following theorem shows that though these classes form a hierarchy with three levels, it is very hard to prove that the classes are distinct. The distinctness of these classes has complexity theoretic consequences, as shown below.

**Theorem 8:**
(a) The classes \(Ppr(\#S_0), Ppr(\#S_1),\) and \(Ppr(\#\Pi_1)\) form a hierarchy with three levels as follows:
\[
Ppr(\#S_0) \subseteq Ppr(\#S_1) \subseteq Ppr(\#\Pi_1) = \#P,
\]
and their distinctness has consequences as below.
(b) \(Ppr(\#S_0) = Ppr(\#S_1)\) if and only if \(P = \#P\).
(c) If \(Ppr(\#S_1) = Ppr(\#\Pi_1)\) then \(NP = RP\).

**Proof:**
(a) The containments follow from Theorem 2. The last equality follows from the fact that \(\#3CNF\), a problem complete for \(\#P\) under parsimonious reductions, is in \(\#\Pi_1\).
(b) To show the forward direction, it suffices to note that \(\#S_1\) has \(\#P\)-complete problems, for example, \(\#3DNF\). The other direction follows trivially.
(c) The proof of this part follows from Proposition 1 and Theorem 4 and the fact that if \(\#CNF\) has a polynomial time randomized approximation algorithm, then \(NP = RP\).

It should be pointed out that a similar situation holds with problems in the class \(NP\). For example, \(3COLORABILITY\) is expressible using a strict \(\Sigma_1\) formula, i.e., an existential second order formula whose first order part is universal. This is a provably strict subclass of \(\Sigma_1\) while the closure of strict \(\Sigma_1\) under polynomial reducibilities is the entire class \(NP\).

6 Concluding Remarks

This work has initiated logic based investigations of the function class \(\#P\) and obtained some positive results. We discuss below some unresolved issues which, either arise from this work, or are more naturally expressible in this setting.

As noted already, counting problems like \(\#3CNF\) and \(\#CNF\) which are parsimoniously complete for \(\#P\), are not likely to be in the class \(\#S_1\) or its closure \(Ppr(\#S_1)\) unless there are unlikely complexity theoretic collapses such as \(NP = RP\) (see Remark 5). However, there are several other \(\#P\)-complete problems which are not members of \(\#S_1\), but may still be in \(Ppr(\#S_1)\) without seemingly having any drastic complexity theory consequences. For example, is \(\#2CNF\) (the restriction of \(\#CNF\) with at most two literals per clause) product reducible to \(\#3DNF\) or \(\#DNF\)? More generally, are counting problems whose decision versions known to be in \(P\), product reducible to \(\#DNF\)? Such reductions, if they exist, would give polynomial time randomized approximation algorithms for problems outside \(\#S_1\) or \(\#RE_2\). Even though such reductions do not seem to have any unlikely complexity theoretic consequence, we believe they are unlikely. For example, we conjecture that there is no product reduction from \(\#2CNF\) to \(\#DNF\). We also believe that answer to this con-
jecture will come from finer logic based classification of the functions in \#P.

As mentioned in Section 5, we would like to know if the class \#P_1 (and \#P_2) also captures counting problems which are feasible under some other notion of feasibility. Any such feasibility notion which separates \#P_1 from the rest of \#P would essentially separate the difficulty of \#3CNF from \#CNF and such an answer would be interesting.

There are some counting problems in \#P for which there are efficient superpolynomial time approximation algorithms or for which there are polynomial time randomized approximation algorithms for special cases of inputs (see, for example, [DLMV88]). We have not dealt with such cases in this paper. It will be interesting to see if in these cases too, logical definability of the problems has a role to play.

Finally, we would like a syntactic characterization of problems like \#EXT which are not in \#P_1 but which have polynomial time randomized approximation algorithms [DFK91].

Acknowledgments: We wish to thank J. Radhakrishnan for assisting us with part 4 of the proof of Theorem 2 and C. Papadimitriou for suggesting that we study the closure classes. We also thank Phokion Kolaitis and Phil Long for reading the paper thoroughly and making suggestions that vastly improved the readability of the paper. The last author would also like to thank Phokion Kolaitis for his encouragement and support.

References


Appendix

We list below some natural counting problems, which are in \#\Sigma_1 or \#\Sigma_2 and hence have fully polynomial randomized approximation schemes.

- **#MONOCHROMATIC-kCLIQUE-PARTITIONS (MKCP):** Given a graph \( G = (V, E) \), the counting function is the number of 2-partitions of \( V \) so that at least one of the partitions has a \( k \)-clique. The problem is in \#\Sigma_1. We give below a formulation of this problem for the case \( k = 3 \).

\[
f_{\text{MKCP}}(G) = |\{ S : (3x)(3y)(3z) E(x, y) \land E(x, z) \land E(y, z) \land (x < y) \land (y < z) \land (S(x) \land S(y) \land S(z) \lor \neg S(x) \land \neg S(y) \land \neg S(z))\}|
\]

- **#NON-VERTEX-COVERS:** Given a graph \( G = (V, E) \), the counting function is the number of subsets \( S \subseteq V \) such that \( S \) is not a vertex cover of \( G \). The problem is in \#\Sigma_1 because,

\[
f_{\text{NON-VC}}(G) = |\{ S : (3x)(3y)(E(x, y) \land \neg S(x) \land \neg S(y))\}|
\]

- **#NON-CLIQUEs:** Given a graph \( G = (V, E) \), the counting function is the number of subsets \( S \subseteq V \) such that \( S \) is not a clique in \( G \). The problem is in \#\Sigma_1 because,

\[
f_{\text{NON-CLIQUEs}}(G) = |\{ S : (3x)(3y)(E(x, y) \land \neg S(x) \land \neg S(y))\}|
\]

- **#NON-DOMINATING-SETS:** Given a graph \( G = (V, E) \), the counting function is the number of subsets \( S \subseteq V \) such that \( S \) is not a dominating set, i.e., there is a vertex in \( V - S \) which is not a neighbor of any vertex in \( S \). The problem is in \#\Sigma_2 because,

\[
f_{\text{NON-DS}}(G) = |\{ S : (3x)(3y)(S(x) \land S(y) \land \neg E(x, y))\}|
\]

- **#NON-EDGE-DOMINATING-SETS:** Given a graph \( G = (V, E) \), the counting function is the number of subsets \( S \subseteq E \) such that \( S \) is not an edge dominating set, i.e., there is an edge in \( E - S \) which does not share endpoints with any edge in \( S \). The problem is in \#\Sigma_2 because,

\[
f_{\text{NON-EDGE-DS}}(G) = |\{ S : (3w)(3v)(3y)(3z) (E(w, z) \land E(u, v) \land \neg S(u, v) \land S(y, z) \land \psi(u, v, y, z))\}|
\]

where \( \psi(u, v, y, z) \) is a quantifier-free formula that says \( u, v, y, z \) are distinct.

- **#NON-HITTING-SETS:** Given a collection \( C \) of subsets of a finite set \( U \), the counting function is the number of \( S \subseteq C \) such that there is a set \( S' \in C \) for which \( S \cap S' = \emptyset \). The input instances are represented by finite structures \( A = (A, C, M) \).
where $A = U \cup C$, and $M(x, s)$ is a binary predicate expressing membership of $x$ in the set $s$ (cf. [KT91]). The problem is in $\#\Sigma_2$ because,

$$\text{NP Hitting Set}(A) = \{|S : (\exists a)(\forall x)(C(s)) \land (M(s, x) \rightarrow \neg S(x))\}|.$$