Electronic Colloquium on Computational Complexity, Report No. 166 (2018)

An overview of some semantic and syntactic complexity classes

James L. Cox, and Tayfun Pay

[1] Brooklyn College
Computer and Information Science Department
2900 Bedford Avenue,
Brooklyn, New York 11210
cox@sci.brooklyn.cuny.edu

[2] Graduate Center of New York
Computer Science Department
365 5^{th} Avenue,
New York, New York 10016
tpay@gradcenter.cuny.edu

Abstract. We review some semantic and syntactic complexity classes that were introduced to better understand the relationship between complexity classes **P** and **NP**. We also define several new complexity classes, some of which are associated with Mersenne numbers, and show their location in the complexity hierarchy.

1 Introduction

The study of non-deterministic polynomial-time Turing machines (NPTM) was initiated in [1] with the first NP-Complete problem. This problem asks if a Boolean expression in conjunctive normal form (CNF) has a satisfying truth assignment; and it is referred to as CNF-SAT. It was also shown that CNF-SAT is polynomial time Turing reducible to the problem of finding a tautology for a Boolean expression in disjunctive normal form (DNF), where this problem is CoNP-Complete. Then a whole array of other problems were shown to be NP-Complete in [4] via polynomial time many-one reductions to CNF satisfiability. The polynomial-time hierarchy, **PH**, was introduced in [8] as a hierarchy of complexity classes that are derived from complexity classes **NP** and **CoNP** when used as oracles. A more elaborate list of NP-Complete problems were presented in [32].

Researchers also examined versions of these NP-Complete problems with different constraints. For example, probabilistic polynomial time Turing machines and the corresponding PP-Complete problem Majority-SAT, which asks if a Boolean expression in CNF with n variables have more than 2^{n-1} many satisfying truth assignments, was independently introduced in [6] and [10]. Complexity class $\mathbf{C_{=}P}$ was also introduced in [6], where the canonical $C_{=}P$-Complete problem asks if a Boolean expression in CNF with n variables have exactly 2^{n-1}

ISSN 1433-8092

many satisfying truth assignments, and showed that $\mathbf{C_{=}P} \subseteq \mathbf{PP}$. The complexity class $\mathbf{\#P}$ was introduced in [11], where it's complete problem #SAT asks to find the total number of satisfying truth assignments for a Boolean expression in CNF. It was also proven in [11] that calculating the permanent of a 0-1 matrix is polynomial time Turing reducible to calculating the total number of accepting paths of a NPTM. It was shown in [29] that complexity classes $\mathbf{\#P}$ and $\mathbf{PP}$ are equivalent under polynomial time Turing reductions, that is $\mathbf{P^{PP}} = \mathbf{P^{\#P[1]}}$. Later on, the problems that ask if a Boolean expression in CNF has a unique satisfying truth assignment, and odd number of satisfying truth assignments were examined in [12] and [13], respectively. The complexity classes derived from these studies in [12] and [13] are $\mathbf{US}$ and $\oplus\mathbf{P}$, respectively. It was proven in [18] that $\mathbf{PH} \subseteq \mathbf{BPP^{\oplus P}} \subseteq \mathbf{P^{PP}}$, where $\mathbf{BPP}$ is the bounded-error probabilistic polynomial time as defined in [10]. Some other intriguing complexity classes can be found in [21] and [25].

The study of categorical NPTM was initiated in [9] with the complexity class $\mathbf{UP}$. Other widely used terms instead of categorical are bounded, restricted and semantic. These complexity classes are not believed to possess complete problems, but they rather have promise problems. The promise problem for $\mathbf{UP}$ is called Unambiguous-SAT and it asks the question that when promised to have a unique satisfying truth assignment or none, does the Boolean expression in CNF have a unique satisfying truth assignment? It was shown in [16] that $\mathbf{NP} \subseteq \mathbf{RP}^{Unambiguous-SAT}$, where $\mathbf{RP}$ is the randomized polynomial time as defined in [10]. The semantic version of $\mathbf{C_{=}P}$ was defined in [26] as $\mathbf{Half\_P}$. Another notable semantic complexity class is $\mathbf{EP}$, which was defined in [27]. An $\mathbf{EP}$ machine has an acceptance criterion of power of two and a rejection criterion of zero. It was shown in [27] that the syntactic version of $\mathbf{EP}$, called $\mathbf{ES}$, equals $\mathbf{C_{=}P}$. Various other interesting semantic complexity classes were introduced in [20] and [25].

In the next section, we review several reducibilities and establish what it means to be a semantic and syntactic complexity class. We also provide various examples of these complexity classes along with the proven relationships among them. In the section that follows, we define three new complexity classes, namely the semantic complexity classes $\mathbf{MNP}$ and $\mathbf{F_{=}P}$ and the syntactic complexity class $\mathbf{MNS}$. The complexity classes $\mathbf{MNP}$ and $\mathbf{MNS}$ are associated with Mersenne numbers whereas the complexity class $\mathbf{F_{=}P}$ is closely related to the complexity class $\mathbf{C_{=}P}$. Then in the subsequent sections, we examine the relationship between these new complexity classes and already known ones. More precisely, we prove the following:

- $\mathbf{FewP} \subseteq \mathbf{MNP}$
- $\mathbf{US} \subseteq \mathbf{MNS}$
- $\mathbf{PP} \subseteq \mathbf{NP^{MNS}}$
- $\oplus\mathbf{P} \subseteq \mathbf{NP^{MNS}}$
- $\mathbf{MNP} \subseteq \oplus\mathbf{P}$
- $\mathbf{C_{=}P} \subseteq \mathbf{MNS}$ and $\mathbf{MNS} \subseteq \mathbf{C_{=}P}$ which implies that $\mathbf{MNS} = \mathbf{C_{=}P}$
- $\mathbf{MNP} \cap \mathbf{EP} = \mathbf{UP}$

2 Definitions and Preliminaries

We are interested in polynomial time Turing reducibility (also called Cook reducibility), polynomial time Post reducibility (also called truth table reducibility), polynomial time conjunctive and disjunctive truth table reducibilities and polynomial time many-one reducibility (also called Karp reducibility) as defined below.

Definition 1. Let A and B be classes of languages.

1. A is Turing reducible to B in polynomial time, $(A_{\leq Tur}B)$, such that $A \in \mathbf{P}^B$.
2. A is Post reducible to B in polynomial time, $(A_{\leq Post}B)$, such that $(\exists f \in \mathbf{FP})$ $(\exists C \in \mathbf{P})$ $(\forall x)$ $[(\exists c)(\exists y_1, y_2, \ldots, y_c)$ $[f(x) = y_1\#y_2\#\ldots\#y_c\#] \wedge (x \in A \leftrightarrow x\#X_B(y_1)X_B(y_2)\ldots X_B(y_c) \in C)]$.
3. A is conjunctive truth table reducible to B in polynomial time, $(A_{\leq dtt}B)$, such that $(\exists f \in \mathbf{FP})$ $(\forall x)$ $[(\exists c)(\exists y_1, y_2, \ldots, y_C)$ $[f(x) = y_1\#y_2\#\ldots\#y_C\#]$ $\wedge$ $(x \in A \leftrightarrow y_1 \in B \wedge y_2 \in B \wedge \cdots \wedge y_c \in B)]$.
4. A is disjunctive truth table reducible to B in polynomial time, $(A_{\leq dtt}B)$, such that $(\exists f \in \mathbf{FP})$ $(\forall x)$ $[(\exists c)(\exists y_1, y_2, \ldots, y_C)$ $[f(x) = y_1\#y_2\#\ldots\#y_C\#]$ $\wedge$ $(x \in A \leftrightarrow y_1 \in B \vee y_2 \in B \vee \cdots \vee y_c \in B)]$.
5. A is many-one reducible to B in polynomial time, $(A_{\leq m}B)$, such that $(\exists f \in \mathbf{FP})$ $(\forall x)[x \in A \leftrightarrow f(x) \in B]$.

The following implications hold for all class of languages A and B:

$$(A_{\leq m}B) \begin{array}{c} \nearrow \ (A_{\leq ctt}B) \ \searrow \\ \searrow \ (A_{\leq dtt}B) \ \nearrow \end{array} (A_{\leq Post}B) \rightarrow (A_{\leq Tur}B)$$

Another important reduction is the parsimonious reduction between functions that preserve the number of solutions.

Definition 2. Let F and G be any functions.

1. F is parsimonious reducible to G in polynomial time, $(F_{\leq par}G)$, such that $(\exists h \in \mathbf{FP})$ $(\forall x)[F(x) = G(h(x))]$, where h is a total function.

More detailed explanations about different types of reductions can be found in [7] and more peculiar ones are discussed in [28]. It is well known that if a complexity class is closed under some reduction then a class reduced to it under that reduction is a subset of it.

One way to generalize complexity classes is through leaf languages, where language $L \subseteq \{0,1\}^*$. Then assume that leaf languages L_A and L_R have the property that $L_A \cap L_R = \emptyset$, where L_A is the acceptance criterion and L_R is the rejectance criterion of a leaf language class. [23]

Definition 3. A given complexity class is classified as a syntactic complexity class if and only if it has the property that $L_A \cup L_R = \{0,1\}^*$.

Definition 4. A given complexity class is classified as a semantic complexity class if and only if it has the property that $L_A \cup L_R \neq \{0,1\}^*$.

Definition 5. A language L is in semantic complexity class **UP**, as defined in [9], if there exist a polynomial p and a polynomial-time predicate R such that, for each x,
$x \in L \Rightarrow ||\{y|\ |y| = p(|x|) \wedge R(x,y)\}|| = 1$
$x \notin L \Rightarrow ||\{y|\ |y| = p(|x|) \wedge R(x,y)\}|| = 0$

Definition 6. A language L is in semantic complexity class $\mathbf{UP}_{O_{(k)}}$, as defined in [5], if there exist a constant $k > 1$, a polynomial p and a polynomial-time predicate R such that, for each x,
$x \in L \Rightarrow 1 \leq ||\{y|\ |y| = p(|x|) \wedge R(x,y)\}|| \leq k$
$x \notin L \Rightarrow ||\{y|\ |y| = p(|x|) \wedge R(x,y)\}|| = 0$

Definition 7. A language L is in semantic complexity class **FewP**, as defined in [14], if there exist polynomials p and q and a polynomial-time predicate R such that, for each x,
$x \in L \Rightarrow 1 \leq ||\{y|\ |y| = p(|x|) \wedge R(x,y)\}|| \leq q(x)$
$x \notin L \Rightarrow ||\{y|\ |y| = p(|x|) \wedge R(x,y)\}|| = 0$

Definition 8. A language L is in semantic complexity class **EP**, as defined in [27], if there exist a polynomial p and a polynomial-time predicate R such that, for each x,
$x \in L \Rightarrow ||\{y|\ |y| = p(|x|) \wedge R(x,y)\}|| = 2^t$, where $t \in \mathbb{N}_0 = \{0,1,2,...\}$
$x \notin L \Rightarrow ||\{y|\ |y| = p(|x|) \wedge R(x,y)\}|| = 0$

Definition 9. A language L is in semantic complexity class **Half_P** , as defined in [26], if there exist a polynomial p and a polynomial-time predicate R such that, for each x,
$x \in L \Rightarrow ||\{y|\ |y| = p(|x|) \wedge R(x,y)\}|| = 2^{p(|x|)-1}$
$x \notin L \Rightarrow ||\{y|\ |y| = p(|x|) \wedge R(x,y)\}|| = 0$

It has been shown that $\mathbf{P} \subseteq \mathbf{UP} \subseteq \mathbf{UP}_{O_{(k)}} \subseteq \mathbf{FewP} \subseteq \mathbf{EP} \subseteq \mathbf{NP}$ and that $\mathbf{P} \subseteq \mathbf{Half\_P} \subseteq \mathbf{EP} \subseteq \mathbf{NP}$.

Definition 10. A language L is in syntactic complexity class $\mathbf{C_{=}P}$, as defined in [6], if there exist a polynomial p and a polynomial-time predicate R such that, for each x,
$x \in L \Leftrightarrow ||\{y|\ |y| = p(|x|) \wedge R(x,y)\}|| = 2^{p(|x|)-1}$

An alternate definition of $\mathbf{C_{=}P}$ was provided in [17], such that a language L is in $\mathbf{C_{=}P}$ if there exist a $f \in \mathbf{FP}$ such that $x \in L$ if and only if the total number of accepting paths equals $f(x)$, for every $x \in \Sigma^*$.

Definition 11. A language L is in syntactic complexity class **ES**, as defined in [27], if there exist a polynomial p and a polynomial-time predicate R such that, for each x,
$x \in L \Leftrightarrow ||\{y|\ |y| = p(|x|) \wedge R(x,y)\}|| = 2^t$, where $t \in \mathbb{N}_0 = \{0,1,2,...\}$

Definition 12. A language L is in syntactic complexity class **PP**, as defined in [6], if there exist a polynomial p and a polynomial-time predicate R such that, for each x,

$x \in L \Leftrightarrow ||\{y|\ |y| = p(|x|) \wedge R(x,y)\}|| > 2^{p(|x|)-1}$

Definition 13. Functional complexity class **#P**, as defined in [11], counts the total number of accepting paths of a NPTM.

$\mathbf{\#P} = \{f|(\exists \text{ a NPTM } M)(\forall x)\ [f(x) = \#accept_M(x)]\}$.

It was proven in [27, 6] that $\mathbf{ES} = \mathbf{C_{=}P} \subseteq \mathbf{PP}$ and in [15] that $\mathbf{P^{PP}} = \mathbf{P^{\#P[1]}}$.

Definition 14. A language L is in syntactic complexity class **US**, as defined in [12], if there exist a polynomial p and a polynomial-time predicate R such that, for each x,

$x \in L \Leftrightarrow ||\{y|\ |y| = p(|x|) \wedge R(x,y)\}|| = 1$

Definition 15. A language L is in syntactic complexity class $\oplus$**P**, as defined in [13], if there exist a polynomial p and a polynomial-time predicate R such that, for each x,

$x \in L \Leftrightarrow ||\{y|\ |y| = p(|x|) \wedge R(x,y)\}|| \not\equiv 0 \text{ (Mod 2)}$

Definition 16. A language L is in semantic complexity class **RP**, as defined in [10], if there exist a polynomial p and a polynomial-time predicate R such that, for each x,

$x \in L \Rightarrow ||\{y|\ |y| = p(|x|) \wedge R(x,y)\}|| \geq 2^{p(|x|)-1} + \epsilon$ where $0 < \epsilon < 2^{p(|x|)-1}$

$x \notin L \Rightarrow ||\{y|\ |y| = p(|x|) \wedge R(x,y)\}|| = 0$

Definition 17. A language L is in semantic complexity class **BPP**, as defined in [10], if there exist a polynomial p and a polynomial-time predicate R such that, for each x,

$x \in L \Rightarrow ||\{y|\ |y| = p(|x|) \wedge R(x,y)\}|| \geq 2^{p(|x|)-1} + \epsilon$ where $0 < \epsilon < 2^{p(|x|)-1}$

$x \notin L \Rightarrow ||\{y|\ |y| = p(|x|) \wedge R(x,y)\}|| \leq 2^{p(|x|)-1} + \epsilon$ where $0 < \epsilon < 2^{p(|x|)-1}$

It was shown that $\mathbf{CoNP} \subseteq \mathbf{US} \subseteq \mathbf{C_{=}P}$ and that $\mathbf{Half\_P} \subseteq \mathbf{RP} \subseteq \mathbf{BPP} \subseteq \mathbf{PP}$. It has been proven in [16] that $\mathbf{NP} \subseteq \mathbf{RP^{\oplus P}}$.

Definition 18. Polynomial Hierarchy, as defined in [2], for $i \geq 0$ is

$\mathbf{PH} = \bigcup_i \Sigma_i^p$,

$\Sigma_{i+1}^p = \mathbf{NP}^{\Sigma_i^p}$,

$\Pi_{i+1}^p = \mathbf{CoNP}^{\Sigma_i^p}$,

$\Delta_{i+1}^p = \mathbf{P}^{\Sigma_i^p}$,

where $\Sigma_0^p = \Pi_0^p = \Delta_0^p = \Delta_1^p = \mathbf{P}$, $\Sigma_1^p = \mathbf{NP}$, $\Pi_1^p = \mathbf{CoNP}$, and $\Delta_2^p = \mathbf{P^{NP}} = \mathbf{P^{CoNP}}$.

It has been proven in [18] that $\mathbf{PH} \subseteq \mathbf{BPP^{\oplus P}} \subseteq \mathbf{P^{\#P}}$.

Definition 19. Functional complexity class **GapP**, as defined in [25], is the difference between total number of accepting and rejecting paths of a NPTM.

$\mathbf{GapP} = \{f|(\exists \text{ a NPTM } M)(\forall x)\ [f(x) = \#accept_M(x) - \#reject_M(x)]\}$.

It can be easily seen that $\#\mathbf{P} \subseteq \mathbf{GapP}$. Some complexity classes can also be defined using **GapP** functions. For example, language L is in $\mathbf{C_=P}$ if there exist a $f \in \mathbf{GapP}$ such that $x \in L$ if and only if $f(x) = 0$ for every $x \in \Sigma^*$.

Definition 20. A language L is in semantic complexity class **SPP**, as defined in [25], if there exist a **GapP** function f such that, for each x

$x \in L \Rightarrow f(x) = 2$
$x \notin L \Rightarrow f(x) = 0$

The acceptance criterion is 2 instead of 1 because if the total number of computation paths are even then the difference between the total number of accepting and rejecting paths cannot be an odd number. **SPP** is the smallest complexity class that can be defined using **GapP** functions. It was shown in [25] that $\mathbf{UP} \subseteq \mathbf{SPP}$, $\mathbf{SPP} \subseteq \oplus\mathbf{P}$. It was also proven in [25] that $\mathbf{SPP}^{\mathbf{SPP}} = \mathbf{SPP}$, $\mathbf{C_=P}^{\mathbf{SPP}} = \mathbf{SPP}$ and $\mathbf{PP}^{\mathbf{SPP}} = \mathbf{PP}$.

Definition 21. A language L is in semantic complexity class **WPP**, as defined in [25], if there exist a **GapP** function f and a **FP** function g such that, for each x

$x \in L \Rightarrow f(x) = g(x)$
$x \notin L \Rightarrow f(x) = 0$

It was shown in [25] that $\mathbf{SPP} \subseteq \mathbf{WPP} \subseteq \mathbf{C_=P}$.

3 Even more complexity classes

3.1 Mersenne Number Satisfiability

Mersenne numbers are named after Marin Mersenne whom began the study of these numbers in the 17^{th} century. A Mersenne number is a positive integer that is one less than a power of two, $M_n = 2^n - 1$ and consists of all 1 bits in its binary representation. It is well known that if M_n is a prime number then n is a prime number as well. The study of Mersenne primes has been an alluring field with the emergence of powerful computers, which can do calculations that would be very-hard for humans to do by hand. There have been 50 Mersenne primes discovered to date and the largest known Mersenne prime is $2^{77,232,917} - 1$.

Definition 22. A language L is in syntactic complexity class **MNS** if there exist polynomial p and a polynomial-time predicate R such that, for each x,

$x \in L \Leftrightarrow ||\{y|\ |y| = p(|x|) \wedge R(x, y)\}|| = 2^t - 1$, where $t \in \mathbb{N}_{>0} = \{1, 2, 3, ...\}$

Definition 23. Mersenne-Number-SAT

Given a Boolean expression in CNF, is it true that it has Mersenne number of satisfying truth assignments?

Theorem 1. Mersenne-Number-SAT is **MNS**-complete

Proof. Mersenne-Number-SAT is clearly in **MNS** given the above definition. Completeness follows from the parsimonious reductions in [11] of any problem in **#P** to #SAT.□

Definition 24. A language L is in semantic complexity class **MNP** if there exist polynomial p and a polynomial-time predicate R such that, for each x,
$x \in L \Rightarrow ||\{y|\ |y| = p(|x|) \wedge R(x,y)\}|| = 2^t - 1$, where $t \in \mathbb{N}_{>0} = \{1, 2, 3, ...\}$
$x \notin L \Rightarrow ||\{y|\ |y| = p(|x|) \wedge R(x,y)\}|| = 0$

Definition 25. Promise-Mersenne-Number-SAT
Given a Boolean expression in CNF that is promised to have Mersenne number of satisfying truth assignments or none, does it have Mersenne number of satisfying truth assignments?

We obtain the following inclusions just from the above definitions:
· **MNP** $\subseteq$ **MNS**
· **MNP** $\subseteq$ **NP**
· **MNS** $\subseteq \mathbf{P}^{\mathbf{PP}} = \mathbf{P}^{\#\mathbf{P}[1]}$

3.2 A long lost semantic relative of $\mathbf{C_{=}P}$

A perceptive reader would have noted that two out of the three alternate definitions of $\mathbf{C_{=}P}$ that employ **#P** functions have their own semantic versions, namely **Half_P** and **EP**. And by definition **Half_P** $\subseteq$ **EP**. It is not known whether they are equal, although their syntactic versions have been proven to be equal in [27]. We next define the semantic version of the third alternate definition of $\mathbf{C_{=}P}$ that also employ **#P** functions.

Definition 26. A language L is in semantic complexity class $\mathbf{F_{=}P}$ if there exist a polynomial p, a polynomial-time predicate R and a polynomial time computable function f, such that, for each x,
$x \in L \Rightarrow ||\{y|\ |y| = p(|x|) \wedge R(x,y)\}|| = f(x)$
$x \notin L \Rightarrow ||\{y|\ |y| = p(|x|) \wedge R(x,y)\}|| = 0$

It is not hard to see that we would have still obtained the same complexity class if we had changed the **GapP** function in the definition of **WPP** to a **# P** function.

Definition 27. Promise-Exact-Number-SAT
Given a Boolean expression in CNF that is promised to have $f(x)$ many satisfying truth assignments or none, does it have $f(x)$ many satisfying truth assignments?

We obtain the following inclusions just from the above definitions:
· **FewP** $\subseteq \mathbf{F_{=}P}$
· $\mathbf{F_{=}P} \subseteq \mathbf{C_{=}P}$
· $\mathbf{F_{=}P} \subseteq$ **WPP**

4 Relationship with complexity class FewP

Definition 28. Non-gappy [27]
Let S be any set of positive integers. Then S is non-gappy if $S \neq 0$ and $(\exists k > 0)(\forall n \in S)(\exists m \in S)[m > n \wedge m/n \leq k]$.

Definition 29. P-printable [31]
Let L be any subset of Σ^*. Then L is P-printable if there is a deterministic Turing machine M that runs in polynomial-time such that, for every nonnegative integer n, $M(0^n)$ prints out the set $\{x | x \in L \wedge |x| \leq n\}$.

Furthermore, Theorem 3.4 in [27] states that, "Let T be any set of positive integers such that T has a non-gappy, P-printable subset. Then **FewP** is contained in any complexity class with the acceptance criterion of T and rejectance criterion of zero."

Theorem 2. FewP $\subseteq$ MNP

Proof. The acceptance criterion of **MNP** is clearly Non-gappy and P-printable according to the above definitions. Then given Theorem 3.4 in [27], **FewP** is contained in **MNP**.
Therefore, **FewP** $\subseteq$ **MNP**.□

5 Relationship with complexity class US

Theorem 3. US $\subseteq$ MNS

Proof. Suppose we have a **US** machine M. We construct machine M', that originally has 2^n many paths. Then on each one of it's accepting paths, machine M' non-deterministically decides to accept on $2^n - 1$ many paths. Then we claim that machine M' will have a Mersenne number of accepting paths if and only if machine M has a single accepting path. To observe that this is true, note that if the number of accepting paths of the original machine M is k, then k must satisfy $k(2^n - 1) = 2^m - 1$, where m is a positive integer. Thus $k = \dfrac{2^m - 1}{2^n - 1}$. The general solution of this equation for $k \in \mathbb{N}_{>0}$ is $m = \dfrac{2i\pi k}{\ln 2}$, where i is an imaginary number. The first two values of k that make m an integer (and in fact a real number) are 1 and $2^n + 1$. Since $k \in \{1, ..., 2^n\}$, this means that M' will have a Mersenne number of accepting paths if and only if $k = 1$. As a result, machine M' is a **MNS** machine.
Therefore, **US** $\subseteq$ **MNS**.□

We should note that there are two well known CoNP-complete problems, namely the CNF contradiction and the DNF tautology, where the former asks if no truth assignment satisfies a Boolean expression in CNF and the latter asks if all possible truth assignments satisfies a Boolean expression in DNF. When

one was shown to be CoNP-complete, the other was easily shown to be CoNP-complete by simply reversing the accepting and rejecting states of a NPTM. Then by employing the same methodology, we can easily show that Maximum-Mersenne-Number SAT, which asks if a Boolean expression in DNF with n variables has $2^n - 1$ many satisfying truth assignments, is US-complete. Note that finding a unique satisfying truth assignment for a Boolean expression in DNF and finding a maximum Mersenne number of satisfying truth assignments for a Boolean expression in CNF are both computable in polynomial time.

6 Relationship with complexity class PP

In order to prove the following result using a NPTM equipped with a **MNS** oracle to recognize Majority-SAT, we must overcome a technical difficulty that occurs if the input formula is unsatisfiable. We finesse this problem by initially running a standard NPTM for SAT. And we only begin a new simulation regime and make queries on accepting paths. In this way, we will never make queries if the input is unsatisfiable and we will reject outright.

Theorem 4. $\mathbf{PP} \subseteq \mathbf{NP}^{\mathbf{MNS}}$

Proof. We show that a **NP** machine with access to a **MNS** oracle solves the PP-complete problem Majority-SAT.

Assume that we have an **NP** machine M so that the number of satisfying truth assignments to Boolean expression x is equal to the number of accepting paths of M on input x. Also, assume that x has n variables and thus $M(x)$ has 2^n many paths. Then we construct an **NP** machine M' with access to oracle **MNS**. Initially, machine M' simulates machine M on input x. If x is unsatisfiable then all paths of M' will reject. On the other hand, if x has at least one satisfying truth assignment then M' will reach a state where M would have accepted. At this point, M' enters a query state to determine if M would have accepted on more than half of the paths. Then M' non-deterministically selects a number k from 0 to $2^{n-1} - 1$. After that, for each choice of k, we construct another machine M'' that does the following. It non-deterministically chooses to accept on $2^n - 1 + k$ many paths and simulates M on the other paths. Then we query the **MNS** oracle with the Boolean expression $f(M'', x)$ and M' accepts if the query answers YES.

Observation 1: The fact that we entered the query states implies that the Boolean expression x has at least one satisfying truth assignment.

Observation 2: Let M have p many accepting paths on input x. Then M'' has at most $2^n - 1 + k + p$ many accepting paths by construction. Since $p > 0$ by construction as well, then M'' has at least 2^n many accepting paths.

Observation 3: M'' has less than $2^{n+2} - 1$ many accepting paths for any choice of k. The maximum number of accepting paths that M'' can have is when $k = 2^{n-1} - 1$ and the original input formula is a tautology. This results in $(2^n - 1) + (2^{n-1} - 1) + (2^n)$ many accepting paths, which equals $2^{n+1} + 2^{n-1} - 2$ and is clearly less than $2^{n+2} - 1$.

Observation 4: If $p \leq 2^{n-1}$ then M'' accepts on at most $2^{n+1} - 2$ paths. If $p > 2^{n-1}$ then for some choice of k, M'' will accept on exactly $2^{n+1} - 1$ many paths.

From the observations above, the number of accepting paths of M'' lies between $2^n - 1$ and $2^{n+2} - 1$. The only Mersenne number of accepting paths that M'' can have is $2^{n+1} - 1$, which is achieved when $p > 2^{n-1}$. Thus if one of these queries gives the answer YES then x is in Majority-SAT.

Therefore, $\mathbf{PP} \subseteq \mathbf{NP}^{\mathbf{MNS}}$.□

Lemma 1. $\mathbf{P}^{\mathbf{PP}} \subseteq \mathbf{P}^{\mathbf{NP}^{\mathbf{MNS}}}$

It follows from Theorem 4.

Lemma 2. $\mathbf{PH} \subseteq \mathbf{P}^{\mathbf{NP}^{\mathbf{MNS}}}$

It follows from Theorem 4 and Toda's Theorem [18], $\mathbf{PH} \subseteq \mathbf{P}^{\mathbf{PP}}$.

7 Relationship with complexity class $\oplus$P

Theorem 5. $\oplus\mathbf{P} \subseteq \mathbf{NP}^{\mathbf{MNS}}$

Proof. We show that a **NP** machine with access to a **MNS** oracle solves the $\oplus$P-complete problem Parity-SAT.

Assume that we have an **NP** machine M so that the number of satisfying truth assignments to the Boolean expression x is equal to the number of accepting paths of M on input x. Also, assume that x has n variables and thus $M(x)$ has 2^n many paths. Then we construct an **NP** machine M' with access to oracle **MNS** that on input x behaves as follows: It first non-deterministically selects an even number k, where $0 \leq k < 2^n$. Then we construct a machine M'' that does the following. It non-deterministically chooses to accept on k paths and simulates M on the other path. Then we query the **MNS** oracle with the Boolean expression $f(M'', x)$ and M' accepts if the query answers YES.

If M accepts on an even number of paths, then clearly all queries answer no and M' rejects. If M accepts on some odd number of paths, say j, then there exists an even k, where $0 \leq k < 2^n$, such that $k + j = (2^i - 1)$ for some i, and thus M'' accepts on Mersenne number of paths. As a result, machine M' will accept on this query.

Therefore, $\oplus\mathbf{P} \subseteq \mathbf{NP}^{\mathbf{MNS}}$.□

Lemma 3. $\mathbf{BPP}^{\oplus\mathbf{P}} \subseteq \mathbf{BPP}^{\mathbf{NP}^{\mathbf{MNS}}}$

It follows from Theorem 5.

Lemma 4. $\mathbf{PH} \subseteq \mathbf{BPP}^{\mathbf{NP}^{\mathbf{MNS}}}$

It follows from Theorem 5 and Toda's Theorem [18], which states that $\mathbf{PH} \subseteq \mathbf{BPP}^{\oplus\mathbf{P}}$.

However, clearly Lemma 2 is a stronger result than Lemma 4 since $\mathbf{P}^{\mathbf{NP}^{\mathbf{MNS}}} \subseteq \mathbf{BPP}^{\mathbf{NP}^{\mathbf{MNS}}}$

Theorem 6. MNP $\subseteq \oplus$P

Proof. We are given a machine M that decides some **MNP** promise problem *pp*. If input x should be accepted, then machine M accepts on a Mersenne number of paths. If input x should be rejected, then machine M has no accepting paths. Clearly, machine M satisfies the criterion for a $\oplus$**P** machine, since all Mersenne numbers are odd. Therefore, promise problem *pp* is in $\oplus$**P**.□

8 Relationship with complexity class $\mathbf{C_{=}P}$

It is easy to see that we can revise Theorem 4 and achieve $\mathbf{C_{=}P} \subseteq \mathbf{NP^{MNS}}$. However, we can actually do much better.

Theorem 7. $\mathbf{C_{=}P} \subseteq$ MNS

Proof. Suppose we have a $\mathbf{C_{=}P}$ machine M. We design a **MNS** machine M' that accepts the same language as M. Assume for ease of presentation that on input x our machine M has 2^{n+1} many total paths and thus accepts x if and only if it accepts on exactly 2^n many paths. We also assume without loss of generality that n is sufficiently large. Then machine M' on input x immediately accepts on $2^{2n}-1$ many paths and simulates M the following way: 1) For each of M's rejecting states, it accepts and 2) For each of M's accepting states, it accepts on 2^n-1 many paths. We claim that M' accepts if and only if M accepts on 2^n many paths. We show this with the following three lemmas.

Lemma 5. If M accepts on 2^n many paths, then M' accepts on Mersenne number of paths.

If M accepts on 2^n many paths then M' accepts on $(2^{2n}-1)+(2^n*(2^n-1))+(2^n)$ many paths. This expression evaluates to $2^{2n}-1+2^{2n}-2^n+2^n$, which simplifies to $2^{2n+1}-1$. Therefore, M' will accept since $2^{2n+1}-1$ is a Mersenne number.

Lemma 6. If M accepts on less than 2^n many paths, then M' does not accept on Mersenne number of paths.

Machine M' will accept on greater than $2^{2n}-1$ many paths by construction, since it also accepts on rejecting paths. Furthermore, M' will accept on less than $2^{2n+1}-1$ many paths once again by construction. Thus the number of accepting paths of M' will fall in between two consecutive Mersenne numbers. Therefore, M' will reject.

Lemma 7. If M accepts on more than 2^n many paths then M' does not accept on Mersenne number of paths.

Machine M' will accept on greater than $2^{2n+1}-1$ many paths by construction. Then the maximum number of accepting paths is achieved for M' when M accepts on 2^{n+1} many paths. In this case, M' will accept on $(2^{2n}-1)+((2^{n+1})*(2^n-1))$ many paths once again by construction. This expression equals $2^{2n+1}+2^{2n}-2^{n+1}-1$ which is less than $2^{2n+2}-1$. Thus the number of accepting paths of M' will fall in between two consecutive Mersenne numbers. Therefore, M' will reject.

As shown by the previous three lemmas that the **MNS** machine M' accepts if and only if the $\mathbf{C}_{=}\mathbf{P}$ machine M accepts.

Therefore, $\mathbf{C}_{=}\mathbf{P} \subseteq \mathbf{MNS}$.□

Theorem 8. $\mathbf{MNS} \subseteq \mathbf{C}_{=}\mathbf{P}$

Proof. We provide a polynomial time disjunctive truth-table reduction from Mersenne-number-SAT to a language in $\mathbf{C}_{=}\mathbf{P}$. It is well-known that $\mathbf{C}_{=}\mathbf{P}$ is closed under polynomial-time disjunctive truth table reductions, which was proven in [30]. Assume that we have a **MNS** machine M that recognizes Mersenne-number-SAT. Let x be an input with n variables. We next show the disjunctive truth-table reduction to the canonical $\mathrm{C}_{=}\mathrm{P}$-complete problem Equal-SAT. Assume that we have a polynomial time machine M' that carries out the reduction. For each $i \in \{0, ..., n\}$, machine M' constructs a machine M_i that does the following. It immediately accepts on $2^n - 2^i$ many paths and rejects on 2^i many paths, and also simulates machine M. Then for each i, we query the Equal-SAT oracle with the Boolean expression $f(M_i, x)$ and M' accepts if any of these queries answer YES.

Therefore, $\mathbf{MNS} \subseteq \mathbf{C}_{=}\mathbf{P}$.□

Corollary 1. $\mathbf{MNS} = \mathbf{C}_{=}\mathbf{P}$

Immediate consequence of Theorems 7 and 8.

9 Conclusion

We introduced two new semantic complexity classes and one new syntactic complexity class; and showed their location in the complexity hierarchy. It ended up being the case that **MNS** actually equals $\mathbf{C}_{=}\mathbf{P}$. However, a simple padding argument would have not yielded the result in Theorem 7, that is $\mathbf{C}_{=}\mathbf{P} \subseteq \mathbf{MNS}$. What our proof of Theorem 7 actually demonstrates is that given a Boolean expression F in CNF with n variables, one can in polynomial time construct F' with m variables so that the number of satisfying truth assignments to F' is guaranteed to lie between $2^{m-2} - 1$ and $2^m - 1$. In fact, F' will have exactly $2^{m-1} - 1$ satisfying truth assignments if and only if F was satisfied by exactly half of its assignments.

On the other hand, the relationship between **EP** and **MNP** is not so clear other than the fact that their intersection equals **UP**, that is $\mathbf{EP} \cap \mathbf{MNP} = \mathbf{UP}$. Although, we can change the acceptance criterion of a **MNS** machine to be some specific power of two using the methodology in Theorem 7, we cannot necessarily do the same thing with the acceptance criterion of a **MNP** machine. This is because we would also need to consider the rejection criterion of a **MNP** machine and the result in Theorem 7 does not yield zero accepting paths when the original machine has zero accepting paths. However, **MNP** can be viewed as the analog of **EP** that is contained in $\oplus\mathbf{P}$, which **EP** is not known to be. In fact, a relativized world was shown in [27] such that $\exists \mathbf{A}, \mathbf{EP}^{\mathbf{A}} \not\subseteq \oplus\mathbf{P}^{\mathbf{A}}$.

Another interesting question arises with respect to the relationship between $\mathbf{F}_{=}\mathbf{P}$, and **EP** and **MNP**. There does not seem to be a straightforward proof

to show any type of inclusion among them. However, it seems more likely that both **EP** and **MNP** are contained in $\mathbf{F}_{=}\mathbf{P}$. Also, is $\mathbf{F}_{=}\mathbf{P}$ contained in $\oplus\mathbf{P}$ just like **MNP**; or does there exists a relativized world where $\mathbf{F}_{=}\mathbf{P}$ is not contained in $\oplus\mathbf{P}$, just like **EP**.

Finally, we would like to mention that we attempted to change the base machine in Theorem 5 from **NP** to **RP** with no success. We also tried to derive a result just like Theorem 7 between $\oplus\mathbf{P}$ and **MNS**, but once again we were not able to do better than $\mathbf{NP}^{\mathbf{MNS}}$. However, we do think strongly about the possibility of $\oplus\mathbf{P} \subseteq \mathbf{RP}^{\mathbf{MNS}}$ or even $\oplus\mathbf{P} \subseteq \mathbf{MNS}$.

References

1. S. Cook, The complexity of theorem proving procedures, Proceedings of the Third Annual ACM Symposium on Theory of Computing, 151-158, 1971.
2. A. Meyer and L. Stockmeyer, The equivalence problem for regular expressions with squaring requires exponential space, In Proceedings of the 13th IEEE Symposium on Switching and Automata Theory, 125-129, 1972.
3. W. J. Savitch, Relationships between nondeterministic and deterministic tape complexities, Journal of Computer and Systems Sciences, 4, 177-192, 1970.
4. R. M. Karp, Reducibility Among Combinatorial Problems, In Raymond E. Miller and James W. Thatcher (editors). Complexity of Computer Computations. New York: Plenum Press, 85-102, 1972.
5. R. Beigel, On the relativized power of additional accepting paths, In the proceedings of the 4th Structure in Complexity Theory Conference, IEEE Computer Society Press, 216-224, 1989.
6. J. Simon, On Some Central Problems of Computational Complexity, 1975, Cornell University Ithaca.
7. R. Ladner and N. Lynch and A. Selman, A comparison of polynomial time reducibilities, Theoretical Computer Science, 103-124, 1975.
8. L. J. Stockmeyer, The polynomial hierarchy, Theoretical Computer Science, 1-22, 1976.
9. L. Valiant, The relative complexity of checking and evaluating, Information Processing Letters, 20-23, 1976.
10. J. Gill, Computational complexity of probabilistic Turing machines, SIAM Journal of Computing, 675-695, 1977.
11. L.G. Valiant, The complexity of computing the permanent, Theoretical Computer Science, 181-201, 1979.
12. A. Blass and Y. Gurevich, On the unique satisfiability problem, Information and control, 80-88, 1982.
13. C.H. Papadimitriou and S. Zachos, Two remarks on the power of Counting, Theoretical Computer Science, 269-275, 1983.
14. E. Allender, The complexity of sparse sets in P, Structure in Complexity Theory, 1-11, 1986.
15. J. Balcazar and R. Book and U. Schoning, The Polynomial-Time Hierarchy and Sparse Oracles, Journal of the Association for Computing Machinery, 603-617, 1986.
16. L.G. Valiant and V. Vazirani, NP is as easy as detecting Unique Solutions, Theoretical Computer Science, 85-93, 1986.

17. K. Wagner, The complexity of combinatorial problems with succinct input representations, Acta Informatica, 225-256, 1986.
18. S. Toda, On the computational power of PP and Parity-P, IEEE FOCS, 514-519, 1989.
19. R. Rubinstein, Structural Complexity Classes of Sparse Sets: Intractability, Data Compression and Printability, 1988, Northeastern University Boston.
20. E. Allender and R. Rubinstein, P-printable sets, SIAM Journal on Computing, 1193-1202, 1988.
21. J.-Y. Cai and L. A. Hemachandra, On the power of parity polynomial time, STACS, 229-240, 1989.
22. R. Beigel and R. Chang and M. Ogiwara, Relationships between nondeterministic and deterministic tape complexities, Mathematical Systems Theory, 293-310, 1993.
23. Christos H. Papadimitriou, Computational Complexity, Addison-Wesley Longman, 1994.
24. R. Beigel and H. Buhrman and L. Fortnow, NP might not be as easy as detecting unique solutions, In Proceedings of the 30th ACM Symposium on Theory of Computing, 203-208, 1998.
25. S. Fenner and L. Fortnow and S. Kurtz, Gap Definable counting classes, Journal of Computer and System Sciences, 116-148, 1994.
26. A. Berthiaume and G. Brassard, The quantum challenge to structural complexity theory, Proceedings of Structure in Complexity Theory, 132-137, 1992.
27. B. Borchert and L. Hemaspaandra and J. Rothe, Restrictive Acceptance Suffices for Equivalance Problems, LMS J Comput. Math, 86-95, 2000.
28. Lane A. Hemaspaandra and Mitsunori Ogihara, The Complexity Theory Companion, Springer, 2002.
29. J. Simon, On the difference between one and many, 4th Colloq. on Automata, Languages and Programming, 1977.
30. R. Beigel and R. Chang and M. Ogiwara, A relationship between difference hierarchies and relativized polynomial hierarchies, Mathematical Systems Theory, 293-310, 1993.
31. J. Hartmanis and Y. Yesha, Computation times of NP sets of different densities, Theoretical Computer Science, 17-32, 1984.
32. M.R. Garey and D. S. Johnson, Computers and Intractability, 1979.

ECCC ISSN 1433-8092

https://eccc.weizmann.ac.il