The Complexity of Generating and Checking Proofs of Membership

Harry Buhrman*[1] and Thomas Thierauf[2]

[1] CWI. PO Box 94079, 1090 GB Amsterdam, The Netherlands
email: buhrman@cwi.nl
[2] Abt. Theoretische Informatik, Universität Ulm, 89069 Ulm, Germany
email: thierauf@informatik.uni-ulm.de

Abstract. We consider the following questions:

1. Can one compute satisfying assignments for satisfiable Boolean formulas in polynomial time with parallel queries to NP?
2. Is the unique optimal clique problem (UOCLIQUE) complete for $\mathrm{P}^{\mathrm{NP}[O(\log n)]}$?
3. Is the unique satisfiability problem (USAT) NP hard?

We define a framework that enables us to study the complexity of generating and checking proofs of membership. We connect the above three questions to the complexity of generating and checking proofs of membership for sets in NP and $\mathrm{P}^{\mathrm{NP}[O(\log n)]}$. We show that an affirmative answer to any of the three questions implies the existence of coNP checkable proofs for $\mathrm{P}^{\mathrm{NP}[O(\log n)]}$ that can be generated in $\mathrm{FP}_{\parallel}^{\mathrm{NP}}$. Furthermore, we construct an oracle relative to which there do not exist coNP checkable proofs for NP that are generated in $\mathrm{FP}_{\parallel}^{\mathrm{NP}}$. It follows that relative to this oracle all of the above questions are answered negatively.

1 Introduction

We give more background for the questions mentioned in the abstract.

1.1 Generating Satisfying Assignments

Satisfiability – SAT for short – is the set of satisfiable Boolean formulas. In the early seventies Cook and independently Levin [Co71, Le73] showed that SAT is NP complete. In order to prove P different from NP, many researchers have tried to reveal the computational complexity of SAT.

However, SAT is a decision problem and in most practical circumstances we are not only interested in the knowledge that a solution exists, but we want to compute the solution, in this case a satisfying assignment, as well. Therefore, another fundamental task in computational complexity is to determine the complexity of the *construction problem* for NP complete sets.

* Part of this research was done while visiting the Univ. Politècnica de Catalunya in Barcelona. Partially supported by the Dutch foundation for scientific research (NWO) through NFI Project ALADDIN, under contract number NF 62-376

As an upper bound, it is known that such solutions can be computed in $\mathrm{FP}^{\mathrm{NP}}$, the class of functions computable in polynomial time with access to an oracle in NP. This can be achieved by either doing a binary search or a prefix-computation on the solution space using an appropriately chosen set in NP as an oracle.

Is there a better way to compute solutions for sets in NP? Consider the following subclasses of $\mathrm{FP}^{\mathrm{NP}}$.

- $\mathrm{FP}^{\mathrm{NP}}_{\|}$, the class of functions in $\mathrm{FP}^{\mathrm{NP}}$ that can be computed by making nonadaptive queries to NP, that is, all the queries must be written down before any answers are received from the oracle, and
- NPSV, the class of functions that can be computed by single-valued nondeterministic polynomial-time bounded transducers, that is, on each path where the transducer produces some output, it produces the same output.

Hence, we are especially asking whether it is possible to compute some solution for a given NP complete set in $\mathrm{FP}^{\mathrm{NP}}_{\|}$ or in NPSV. Note that $\mathrm{NPSV} \subseteq \mathrm{FP}^{\mathrm{NP}}_{\|}$.

Define the function class F_{sat} by

$$f \in F_{sat} \iff f(\varphi) = \begin{cases} \text{some satisfying assignment of } \varphi, \text{ if } \varphi \in \mathrm{SAT}, \\ \bot, \qquad\qquad\qquad\qquad\qquad\qquad \text{otherwise,} \end{cases}$$

where $\bot$ means that the function is undefined at that point.

As already mentioned, $F_{sat} \cap \mathrm{FP}^{\mathrm{NP}} \neq \emptyset$. In fact, Krentel [Kr86] showed that the lexicographically smallest satisfying assignment is complete for $\mathrm{FP}^{\mathrm{NP}}$. One of the main open problems [WT93, HNOS94, Og95, BKT94] at this point is the following question: Can satisfying assignments be computed with nonadaptive queries to NP. In other words, is $F_{sat} \cap \mathrm{FP}^{\mathrm{NP}}_{\|} = \emptyset$?

Some progress has been made. Hemaspaandra et.al. [HNOS94] showed that one cannot compute satisfying assignments in NPSV, unless the Polynomial Hierarchy collapses, This result has been improved recently by Ogihara [Og95] who showed that $F_{sat} \cap \mathrm{FP}^{\mathrm{NPSV}[c \ \log(n)]} = \emptyset$, for $c < 1$, unless the Polynomial Hierarchy collapses. It is conjectured that an analog result holds with respect to $\mathrm{FP}^{\mathrm{NP}}_{\|}$. In this paper, we construct an oracle where $\mathrm{FP}^{\mathrm{NP}}_{\|} \cap F_{sat} = \emptyset$. On the other hand, Fortnow [Fo94], extending a result of Watanabe and Toda [WT93], constructed an oracle relative to which $\mathrm{FP}^{\mathrm{NP}}_{\|} \bigcap F_{sat} \neq \emptyset$, and the Polynomial Hierarchy is infinite. This indicates that non-relativizing techniques are needed to settle this question.

1.2 Completeness of UOCLIQUE and USAT

P^{NP} and $\mathrm{P}^{\mathrm{NP}[O(\log n)]}$ are the classes of sets that can be recognized with polynomial, respectively $\log(n)$, many queries to an NP oracle. For many optimization problems, deciding certain properties of an optimal solution is complete for either $\mathrm{P}^{\mathrm{NP}[O(\log n)]}$ or P^{NP} [PZ83, W86, W90].

Consider the UOCLIQUE problem, where, for a given graph G, one has to decide whether G has a unique optimal (that is, largest) clique. UOCLIQUE is clearly in $\mathrm{P}^{\mathrm{NP}[O(\log n)]}$. Papadimitriou and Zachos [PZ83] asked whether UOCLIQUE is complete for $\mathrm{P}^{\mathrm{NP}[O(\log n)]}$, and this is still an open problem. The problems whether a given graph has a unique maximum independent set (UOIS) or a unique minimum vertex cover (UOVC) are easily shown to be many-one equivalent to UOCLIQUE, and hence, the precise complexity of all these problems is also open.

Another well studied set is USAT, the set with formulas that have exactely one satisfying assignment. As an upper bound, USAT is in D^{P}, the class of sets that are the difference of two NP sets. But it is not known to be complete for D^{P}. Blass and Gurevich [BG82] showed that USAT is complete for D^{P} if and only if it is hard for NP. Furthermore, they constructed an oracle such that USAT is not complete for D^{P}. Note, however, that Valiant and Vazirani [VV86] showed that USAT is NP hard under *randomized* many-one reductions. As a lower bound, USAT is coNP hard, but it is not known to belong to coNP. In fact, USAT is not in $\mathrm{coD}^{\mathrm{P}}$, unless the Polynomial Hierarchy collapses [CKR95]. It is widely conjectured that USAT is an "intermediate" problem with respect to coNP and D^{P}, i.e., that it is not complete for D^{P} and does not belong to coNP.

In this paper, we will give some evidence that all the above problems are not complete for the respective classes, $\mathrm{P}^{\mathrm{NP}[O(\log n)]}$ and D^{P}. We will do this by connecting these problems to a general tool: Proof Systems.

1.3 Proof Systems

A satisfying assignment for a Boolean formula is, in some sense, a proof that the formula is satisfiable. That the assignment indeed is a satisfying one can be checked in polynomial time. Furthermore, such assignments can be computed in $\mathrm{FP}^{\mathrm{NP}}$. In the definition below, we essentially allow to vary the complexity of the checking process.

Definition 1. Let $\mathcal{C}$ be a class of sets and $\mathcal{F}$ be a class of functions. A set L has *(polynomially) bounded $\mathcal{C}$-checkable proofs in $\mathcal{F}$*, if there exist a polynomial p, a set $C \in \mathcal{C}$, and a function $f \in \mathcal{F}$ such that $|f(x)| \leq p(|x|)$ for all x, and furthermore

$$\begin{aligned} x \in L &\implies (x, f(x)) \in C \\ x \notin L &\implies \forall y\ (|y| \leq p(|x|))\ \ (x, y) \notin C. \end{aligned}$$

The pair $(\mathcal{C}, \mathcal{F})$ is called a *proof-system for L*. A class of sets $\mathcal{K}$ has a $(\mathcal{C}, \mathcal{F})$ proof-system if every set in $\mathcal{K}$ has a $(\mathcal{C}, \mathcal{F})$ proof-system.

As a first example, clearly NP has a $(\mathrm{P}, \mathrm{FP}^{\mathrm{NP}})$ proof-system. In Section 1.1, we asked whether $F_{sat} \cap \mathrm{FP}^{\mathrm{NP}}_{\|} \neq \emptyset$. A positive answer clearly implies that NP has a $(\mathrm{P}, \mathrm{FP}^{\mathrm{NP}}_{\|})$ proof-system. However, it is not even known whether NP has a $(\mathrm{coNP}, \mathrm{FP}^{\mathrm{NP}}_{\|})$ proof-system.

Intuitively, we have the following trade-off: a more powerful function class can put more information into a proof of membership which makes this proof easier to check. Symmetrically, a more powerful class for checking proofs can compute more information by itself and hence a weaker kind of function class suffices to generate these proofs.

In Section 3, we will construct an oracle relative to which NP does not have a $(\mathrm{coNP}, \mathrm{FP}_{\parallel}^{\mathrm{NP}})$ proof-system. Hence, relative to this oracle $F_{sat} \bigcap \mathrm{FP}_{\parallel}^{\mathrm{NP}} = \emptyset$.

In Section 4, we make a connection to the completeness issue of UOCLIQUE (and UOIS and UOVC). We will see that UOCLIQUE has a $(\mathrm{coNP}, \mathrm{FP}_{\parallel}^{\mathrm{NP}})$ proof-system. Thus, if UOCLIQUE is complete for $\mathrm{P}^{\mathrm{NP}[O(\log n)]}$, then $\mathrm{P}^{\mathrm{NP}[O(\log n)]}$, and hence NP, has a $(\mathrm{coNP}, \mathrm{FP}_{\parallel}^{\mathrm{NP}})$ proof-system, violating the above oracle. Moreover, when considering sets known to be $\mathrm{P}^{\mathrm{NP}[O(\log n)]}$ complete, we observe the following trade-off:

1. For a *natural* candidate $\mathcal{F}$ of proofs that can indeed be checked in coNP, we show that $\mathcal{F} \cap \mathrm{FP}_{\parallel}^{\mathrm{NP}} \neq \emptyset$ if and only if $F_{sat} \cap \mathrm{FP}_{\parallel}^{\mathrm{NP}} \neq \emptyset$ (Theorem 13).
2. The proofs generated by $\mathrm{FP}_{\parallel}^{\mathrm{NP}}$ complete functions can be checked in D^{P}, but not in coNP, unless NP = coNP (Theorem 14).

Both results add some more evidence to the incompleteness of UOCLIQUE.

In Section 5, we show that USAT has a (coNP, NPSV) proof-system. Therefore, if USAT is complete for D^{P} then NP has a $(\mathrm{coNP}, \mathrm{FP}_{\parallel}^{\mathrm{NP}})$ proof-system. This again violates the above oracle

We conjecture that $\mathrm{P}^{\mathrm{NP}[O(\log n)]}$ does not have a $(\mathrm{coNP}, \mathrm{FP}_{\parallel}^{\mathrm{NP}})$ proof-system, and hence, that $F_{sat} \bigcap \mathrm{FP}_{\parallel}^{\mathrm{NP}} = \emptyset$ and that UOCLIQUE and USAT are not complete for $\mathrm{P}^{\mathrm{NP}[O(\log n)]}$ and D^{P}, respectively. However, non-relativizing techniques are necessary to finally settle these questions.

2 Preliminaries

We follow the standard definitions and notations in computational complexity theory (see, e.g., [BDG-I&II, HU79]). We fix an alphabet to $\Sigma = \{0, 1\}$; by a *string* we mean an element of Σ^*, and by a *language* we mean a subset of Σ^*. For a language L, we denote $\overline{L}$ as the complement of L, and for a class $\mathcal{C}$ of languages, $\mathrm{co}\,\mathcal{C} = \{\, \overline{L} \mid L \in \mathcal{C}\,\}$. For any string x, let $|x|$ denote the length of x. The standard lexicographical ordering of Σ^* is used. We consider a standard one-to-one pairing function from $\Sigma^* \times \Sigma^*$ to Σ^* that is computable and invertible in polynomial time. For inputs x and y, we denote the output of the pairing function by (x, y); this notation is extended to denote every n tuple.

For our computation model, we consider a standard Turing machine model. P (NP) denote the classes of languages that are accepted by a polynomial-time deterministic (nondeterministic) Turing machine. E and NE are the analogous classes for exponential time $2^{O(n)}$. FP is the class of polynomial-time computable functions. By using oracle machines, one can define relativized classes like P^{NP} and $\mathrm{FP}^{\mathrm{NP}}$, where the P, resp. FP machine has in addition some NP oracle it can query. We consider several restriction of the oracle access mechanism. In general, a polynomial-time bounded machine can ask polynomially many questions (with respect to the input length) to its oracle. By $\mathrm{P}^{\mathrm{NP}[O(\log n)]}$ and $\mathrm{FP}^{\mathrm{NP}[O(\log n)]}$, we denote the classes where the P, resp. FP machine asks only logarithmically many questions to its oracle. By $\mathrm{P}^{\mathrm{NP}}_{\|}$ and $\mathrm{FP}^{\mathrm{NP}}_{\|}$, we denote the classes where the P, resp. FP machine makes the queries non-adaptive, i.e. queries may not depend on answers to previous queries. For the language classes these two restrictions yield the same class, i.e., $\mathrm{P}^{\mathrm{NP}[O(\log n)]} = \mathrm{P}^{\mathrm{NP}}_{\|}$ [H89]. For the function classes, we only have an inclusion, namely $\mathrm{FP}^{\mathrm{NP}[O(\log n)]} \subseteq \mathrm{FP}^{\mathrm{NP}}_{\|}$ and equality seems unlikely unless the Polynomial Hierarchy collapses [Be88, Se94, To91].

The *Polynomial Hierarchy* is defined as $\mathrm{NP} \cup \mathrm{NP}^{\mathrm{NP}} \cup \mathrm{NP}^{\mathrm{NP}^{\mathrm{NP}}} \cup \ldots$.

The *Exponential Hierarchy* is defined as $\mathrm{E} \cup \mathrm{NE} \cup \mathrm{NE}^{\mathrm{NP}} \cup \mathrm{NE}^{\mathrm{NP}^{\mathrm{NP}}} \cup \ldots$.

The *Boolean Hierarchy* is the closure of NP under the Boolean operations union, intersection, and complement. A subclass of the Boolean Hierarchy is D^{P} [PY84].

$$L \in \mathrm{D}^{\mathrm{P}} \iff \exists A, B \in \mathrm{NP} : \quad L = A - B.$$

When considering reductions between sets, we take the standard many-one reduction. *Hard* and *complete* sets (for some class) are also defined via many-one reductions.

Reductions between functions can be defined as follows. Krentel [Kr86] introduced the *metric reduction*. Let f, g be functions.

$$f \leq^{FP}_{1\text{-}T} g \iff \exists t_1, t_2 \in \mathrm{FP} : \quad f(x) = t_2(x, g \circ t_1(x)).$$

This clearly captures the idea of being able to compute $f(x)$ from one call to g.

We extend this definition to classes of functions F and G. Note that there are many possibilities for such an extension (see [BKT94, CT91, FHOS93, WT93]). We take the following.

$$F \leq^{FP}_{1\text{-}T} G \iff \exists t_1, t_2 \in \mathrm{FP} \;\; \forall g \in G : \quad t_2(x, g \circ t_1(x)) \in F.$$

This is a weak reduction because we don't require that *all* functions in F can be computed with the help of some function from G. There only have to be some FP transducers that, no matter which function from G is used, compute *some* function in F.

3 Proof-Systems for NP

In this section, we address the question whether NP has a $(\mathrm{coNP}, \mathrm{FP}^{\mathrm{NP}}_{\|})$ proof-system. We observe first that NP cannot have a $(\mathrm{coNP}, \mathrm{FP}^{\mathrm{NP}[O(\log n)]})$ proof-system, unless NP = coNP. Suppose an NP set L has such a proof-system, then a coNP machine can accept L by first enumerating all the (polynomially many) potential proofs of membership of the $\mathrm{FP}^{\mathrm{NP}[O(\log n)]}$ function (i.e., without asking the oracle) and then check whether one of them actually is a proof of membership.

Proposition 2. *If* NP *has a* $(\mathrm{coNP}, \mathrm{FP}^{\mathrm{NP}[O(\log n)]})$ *proof-system then* NP = coNP.

Next, we will show the existence of an oracle relative to which NP does not have a $(\mathrm{coNP}, \mathrm{FP}_{\|}^{\mathrm{NP}})$ proof-system. We do this by studying properties of the Exponential Hierarchy. This hierarchy behaves strange in various ways (compared with, say, the Polynomial Hierarchy). It is for example not known whether it possesses the downward separation property, that is, whether E = NE implies that the whole hierarchy collapses to E. Another unresolved issue is the following. Suppose that NE is contained in E/lin[3]. Does this imply that NE = coNE? We have the following connection:

Lemma 3. *If* NP *has a* $(\mathrm{coNP}, \mathrm{FP}_{\|}^{\mathrm{NP}})$ *proof-system and* $\mathrm{NE} \subseteq \mathrm{E/lin}$ *then* $\mathrm{NE} = \mathrm{coNE}$.

However, there exists an oracle such that $\mathrm{NE} \subseteq \mathrm{E/lin}$ but still $\mathrm{NE} \neq \mathrm{coNE}$.

Theorem 4. *There exists an oracle A such that* $\mathrm{NE}^A \subseteq \mathrm{E}^A/1$ *and* $\mathrm{NE}^A \neq \mathrm{coNE}^A$.

Proof. (Sketch) We will borrow techniques from Impagliazzo and Tardos [IT89]. We use the following test language. $L_A = \{0^n \mid \forall y, y1 \notin A, |y| \leq 2^n\}$. $L_A \in \mathrm{coNE}^A$ for all A. We have to construct A such that $L_A \notin \mathrm{NE}^A$. We will use the information theoretical lower bound on the X -search problem to do this. We take a setting of the y 's of length 2^n in such a way that no strategy that searches for one y can do this with a small number of parallel rounds [IT89]. Next we code in $0^{<x,e,l>^2} \in A$ if and only if $< x, e, l > \in K^A = \{< x, e, l > \mid M_e^A$ accepts x in l steps $\}$, a NE^A complete set, assuming that the odd strings $y1$ of length 2^n are in A, according to this setting. However we will not (yet) put these strings, ie the $y1$'s, in A. Next we diagonalize against the n^{th} NE^A machine $M_n^A(0^n)$. Suppose $M_n^A(0^n)$ rejects. In this case we have diagonalized since $0^n \in L_A$. However K^A might not be coded correctly. We will see below that this is not a problem. On the other hand if $M_n^A(0^n)$ accepts, then we put in $y1$'s according to the setting of the lower bound of the X-search problem. The lower bound guarantees that the leftmost accepting path of $M_n^A(0^n)$ can not query one of the $y1$'s put into the oracle. Hence this path will keep accepting, but $0^n \notin L_A$. It remains to show that $NE \subseteq E/1$. The one bit of advice for strings of length n will code what action we took in stage n (i.e., whether we put in the strings $1y$ or not). Suppose we want to know whether $< x, e, l > \in K^A$. If we put in the strings $y1$ then K^A is coded correctly and we can query whether the code for $< x, e, l >$ is in A. On the other hand if we did not put in the $y1$'s the coding of K^A may be wrong. However for every oracle machine e there exists another oracle machine e' such that e' simulates e but whenever it queries a string of the form $y1$ (of length 2^n) it assumes that this string is not in the oracle. Morover we can generate e' from e in exponential time. Hence in this case we query whether the code for $< x, e', l >$ is in A. A complete proof will appear in the final version of this paper.

Since the proof of Lemma 3 relativizes, we conclude that, relative to the oracle constructed in Theorem 4, the first assumption in Lemma 3 cannot hold.

[3] E/lin is the class of sets that can be recognized in exponential time with a linear amount of advice for all strings of length n

Corollary 5. *There is an oracle relative to which*

1. NP *does not have a* $(\mathrm{coNP}, \mathrm{FP}_{\|}^{\mathrm{NP}})$ *proof-system.*
2. NP *does not have a* $(\mathrm{P}, \mathrm{FP}_{\|}^{\mathrm{NP}})$ *proof-system, and*

If one could compute satisfying assignments within $\mathrm{FP}_{\|}^{\mathrm{NP}}$, this would provide a $(\mathrm{P}, \mathrm{FP}_{\|}^{\mathrm{NP}})$ proof-system for NP. Hence, a (sloppy) way of putting Corollary 5 (2) is that *there is an oracle relative to which one cannot compute satisfying assignments within* $\mathrm{FP}_{\|}^{\mathrm{NP}}$.

Finally, we observe that in order to show that NP does not have a $(\mathrm{coNP}, \mathrm{FP}_{\|}^{\mathrm{NP}})$ proof-system, it suffices to show that $\mathrm{P}^{\mathrm{NP}[O(\log n)]}$ does not have a $(\mathrm{coNP}, \mathrm{FP}_{\|}^{\mathrm{NP}})$ proof-system.

Theorem 6. NP *has a* $(\mathrm{coNP}, \mathrm{FP}_{\|}^{\mathrm{NP}})$ *proof-system if and only if* $\mathrm{P}^{\mathrm{NP}[O(\log n)]}$ *has a* $(\mathrm{coNP}, \mathrm{FP}_{\|}^{\mathrm{NP}})$ *proof-system.*

Proof. Suppose that NP has coNP checkable proofs in $\mathrm{FP}_{\|}^{\mathrm{NP}}$. Let $L = L(M^A)$ for some P machine M and some set A in NP, and let $f \in \mathrm{FP}_{\|}^{\mathrm{NP}}$ be a proof generating function for A.

We define a function $g \in \mathrm{FP}_{\|}^{\mathrm{NP}}$ that generates proofs for L that are coNP-checkable. Let $x \in \Sigma^*$ and let $y_1, \ldots, y_k \in \Sigma^*$ be the queries of machine M on input x to its oracle A. Then we define

$$g(x) \quad = \quad ((y_1, w_1), \ldots, (y_k, w_k)),$$

where $w_i = f(y_i)$, if $y_i \in A$, and $w_i = 0$, if $y_i \notin A$, for $i = 1, \ldots, k$. (We assume w.l.o.g. that f is always different from 0.)

Since $f \in \mathrm{FP}_{\|}^{\mathrm{NP}}$, we also have $g \in \mathrm{FP}_{\|}^{\mathrm{NP}}$. Furthermore, for each $i = 1, \ldots, k$, we can check in coNP whether $y_i \in A$, if $w_i \neq 0$ by assumption, and also whether $y_i \notin A$, if $w_i = 0$. Therefore, the set $C = \{(x, w) \mid x \in L \text{ and } g(x) = w\}$ is in coNP. Thus C and g show that L has a $(\mathrm{coNP}, \mathrm{FP}_{\|}^{\mathrm{NP}})$ proof-system.

4 On the Completeness of UOCLIQUE

Optimization problems such as computing the shortest traveling salesman tour or the size of the largest clique are known to be complete for $\mathrm{FP}^{\mathrm{NP}}$ and $\mathrm{FP}^{\mathrm{NP}[O(\log n)]}$, respectively [Kr86]. Papadimitriou considered a decision version by asking whether those optimal solutions are *unique*. The motivation for this might be that in order to decide uniqueness there is no way around to solve, somehow implicit, the underlying optimization problem. Hence, these decision problems are expected to be hard for the corresponding complexity classes, i.e., P^{NP} and $\mathrm{P}^{\mathrm{NP}[O(\log n)]}$. In fact, Papdimitriou [P84] showed that the Unique Optimal Traveling Salesman Problem is complete for P^{NP}. Papadimitriou and Zachos [PZ83] asked whether the Unique Optimal Clique Problem (UOCLIQUE) is complete for $\mathrm{P}^{\mathrm{NP}[O(\log n)]}$. However, this is still an open problem. Since Unique Optimal Independent Set (UOIS) and Unique Optimal Vertice Cover (UOVC) are many-one equivalent to UOCLIQUE, this question can be extended to these two problems. In contrast, Krentel [Kr86] and Wagner [W86] showed that OddCLIQUE, i.e., the problem to decide whether the maximum clique of a graph has an odd number of vertices, is complete for $\mathrm{P}^{\mathrm{NP}[O(\log n)]}$.

Definition 7. UMaxSAT is the set of Boolean formulas in conjunctive normal form with the property that all assignments that satisfy the maximum number of clauses happen to satisfy the same set of clauses.

Kadin [Ka88] showed that UMaxSAT is complete for $\mathrm{P}^{\mathrm{NP}[O(\log n)]}$. Consider the standard reduction from SAT to CLIQUE (see for example [HU79]). Suppose, for some CNF formula $\varphi \in$ UMaxSAT, that at most k clauses are satisfiable at the same time. Note that there can be several assignments satisfying those k clauses. But each such assignment will give a different k clique in the constructed graph. Hence, this construction doesn't provide necessarily a unique optimal clique. When we restrict UMaxSAT even further by requiring that there is exactly one such optimal assignment, then the reduction works.

Definition 8. UMaxASAT is the set of Boolean formulas in conjunctive normal form that have one assignment that satisfies strictly more clauses than any other assignment.

We have already seen that UMaxASAT $\leq_m^P$ UOCLIQUE. But in fact, these two problems are equivalent.

Theorem 9. UMaxASAT $\equiv_m^P$ UOCLIQUE.

Hence, we are asking whether UMaxASAT is complete for $\mathrm{P}^{\mathrm{NP}[O(\log n)]}$. We will show that we can distinguish UMaxASAT and UMaxSAT by the complexity of the proof-systems for these sets. Let us start with UMaxASAT.

Definition 10.

$$f_{UMaxASat}(\varphi) = \begin{cases} \text{the assignment that satisfies} & \\ \text{most of the clauses of } \varphi, & \text{if } x \in \text{UMaxASAT}, \\ \perp, & \text{otherwise.} \end{cases}$$

First of all, we note that $f_{UMaxASat}$ is computable with parallel queries to NP, i.e., $f_{UMaxASat} \in \mathrm{FP}_{\|}^{\mathrm{NP}}$. Moreover, whether some given assignment for a formula φ is indeed the unique one satisfying most of the clauses of φ can be checked in coNP, i.e., the set $C = \{ (\varphi, a) \mid \varphi \in \text{UMaxASAT and } f_{UMaxASat}(\varphi) = a \}$ is in coNP, in fact it is coNP complete.

Proposition 11. UMaxASAT *and* UOCLIQUE *have a* $(\mathrm{coNP}, \mathrm{FP}_{\|}^{\mathrm{NP}})$ *proof-system.*

Recall that relative to the oracle constructed in Theorem 4, $\mathrm{P}^{\mathrm{NP}[O(\log n)]}$ does not have a $(\mathrm{coNP}, \mathrm{FP}_{\|}^{\mathrm{NP}})$ proof-system. Loosely speaking, it follows that UOCLIQUE *and* UMaxASAT *are not complete for* $\mathrm{P}^{\mathrm{NP}[O(\log n)]}$ *relative to that oracle.*

$F_{UMaxSat}$, the class of functions that give some assignment that satisfies the unique maximum number of clauses of a given formula, is the natural class for generating proofs for UMaxSAT. For UMaxASAT, we have

Proposition 12. UMaxSAT *has a* $(\mathrm{coNP}, F_{UMaxSat})$ *proof-system.*

It is not clear what the complexity of $F_{UMaxSat}$ is. Especially, it is not known whether there is some function in $F_{UMaxSat}$ that is computable with parallel queries to NP, i.e., whether $F_{UMaxSat} \cap \mathrm{FP}_{\|}^{\mathrm{NP}} \neq \emptyset$.

Since UMaxSAT is complete for $\mathrm{P}^{\mathrm{NP}[O(\log n)]}$ it follows that $F_{UMaxSat} \cap \mathrm{FP}_{\|}^{\mathrm{NP}} \neq \emptyset$ implies that $\mathrm{P}^{\mathrm{NP}[O(\log n)]}$, and hence NP, has a $(\mathrm{coNP}, \mathrm{FP}_{\|}^{\mathrm{NP}})$ proof-system. Therefore, relative to the oracle of Theorem 4, we have $F_{UMaxSat} \cap \mathrm{FP}_{\|}^{\mathrm{NP}} = \emptyset$.

It follows from the next theorem that $F_{UMaxSat} \cap \mathrm{FP}_{\|}^{\mathrm{NP}} \neq \emptyset$ not only implies a $(\mathrm{coNP}, \mathrm{FP}_{\|}^{\mathrm{NP}})$ proof-system for $\mathrm{P}^{\mathrm{NP}[O(\log n)]}$, but even a $(\mathrm{P}, \mathrm{FP}_{\|}^{\mathrm{NP}})$ proof-system for NP, and, by Theorem 6, for $\mathrm{P}^{\mathrm{NP}[O(\log n)]}$.

Theorem 13. $F_{UMaxSat} \cap \mathrm{FP}_{\|}^{\mathrm{NP}} \neq \emptyset \iff F_{sat} \cap \mathrm{FP}_{\|}^{\mathrm{NP}} \neq \emptyset$.

The above results indicate the difference between UMaxASAT (and UOCLIQUE) on one side and UMaxSAT on the other: it seems that we need a more powerful function class to compute proofs for UMaxSAT than for UMaxASAT in order to have these proofs coNP checkable. At least in some relativized world, this in fact holds. On the other hand, when we restrict the proof generating functions to be in $\mathrm{FP}_{\|}^{\mathrm{NP}}$, we will see below that there exist proof systems for which the checking can be done in D^{P}. In fact, the checking will be D^{P} complete.

If some Boolean formula φ is in UMaxSAT, it is not clear how to compute such a maximum assignment with parallel queries to NP. But the weaker information, which clauses are satisfied by such a maximum assignment can indeed be computed in $\mathrm{FP}_{\|}^{\mathrm{NP}}$. Let φ consist of m clauses C_i, i.e., $\varphi = \bigwedge_{i=1}^{m} C_i$. Then we define

$$h_{UMaxSAT}(\varphi) = \begin{cases} (i_1, \ldots, i_k), & \text{if } \varphi \in \mathrm{UMaxSAT},\ 1 \leq i_1 < \cdots < i_k \leq m, \text{ and} \\ & \text{some assignment that satisfies most of the} \\ & \text{clauses of } \varphi \text{ satisfies exactly clauses } C_{i_1}, \ldots, C_{i_k}, \\ \perp, & \text{otherwise.} \end{cases}$$

$h_{UMaxSAT}$ captures the whole power of $\mathrm{FP}_{\|}^{\mathrm{NP}}$: it is $\mathrm{FP}_{\|}^{\mathrm{NP}}$ complete. Furthermore, the proofs generated by $h_{UMaxSAT}$ can be checked in D^{P}, but not in coNP unless the Polynomial Hierarchy collapses. It follows that UMaxSAT has a $(\mathrm{D}^{\mathrm{P}}, \mathrm{FP}_{\|}^{\mathrm{NP}})$ proof-system.

Theorem 14. *(1)* $h_{UMaxSAT}$ *is* $\mathrm{FP}_{\|}^{\mathrm{NP}}$ *complete,*
(2) $C = \{ (\varphi, y) \mid \varphi \in \mathrm{UMaxSAT}$ *and* $h_{UMaxSAT}(\varphi) = y \}$ *is* D^{P} *complete.*

We will present similar results for other $\mathrm{P}^{\mathrm{NP}[O(\log n)]}$ complete sets in the full version of the paper. In summery, whenever some function in $\mathrm{FP}_{\|}^{\mathrm{NP}}$ generates proofs of membership for one of the known $\mathrm{P}^{\mathrm{NP}[O(\log n)]}$ complete sets, checking such a proof requires the computational power of D^{P}. On the other hand, proofs for UMaxASAT can be checked within coNP, we take this fact as some more evidence that UMaxASAT and UOCLIQUE are not complete for $\mathrm{P}^{\mathrm{NP}[O(\log n)]}$.

5 On the Completeness of USAT

We connect the question whether NP and $\mathrm{P}^{\mathrm{NP}[O(\log n)]}$ have a $(\mathrm{coNP}, \mathrm{FP}_{\|}^{\mathrm{NP}})$ proof-system with the question whether USAT is complete for D^{P}. We start by considering some classes that indeed have such proof-systems.

Definition 15. A set A is in UP if there exists a nondeterministic polynomial-time bounded Turing machine M that has at most one accepting path for each input and $L(M) = A$. A function f is in FUP if there exists a nondeterministic polynomial-time bounded Turing transducer that has at most one accepting path for each input x and outputs $f(x)$.

Proposition 16. *(1)* UP *has a* $(\mathrm{P}, \mathrm{FUP})$ *proof system,*
(2) USAT *has a* $(\mathrm{coNP}, \mathrm{FP}_{\|}^{\mathrm{NP}})$ *proof system.*

As already mentioned in Section 1.2, USAT is complete for D^{P} if SAT $\leq_m^P$ USAT. It follows from Proposition 16 (2) that SAT $\leq_m^P$ USAT implies that NP has coNP-checkable proofs in $\mathrm{FP}_{\|}^{\mathrm{NP}}$. Recall that the latter is not true relative to the oracle in Theorem 4.

We show below that the assumption that SAT $\leq_m^P$ USAT even implies that NP has coNP-checkable proofs in NPSV. Assume the stronger hypothesis on the reduction from SAT to USAT, that there is a function $h \in \mathrm{FP}$ that many-one reduces SAT to USAT in such a way that from any $\varphi \in \mathrm{SAT}$ and the unique satisfying assignment of $h(\varphi)$ one can compute in polynomial time some satisfying assignment of φ. We show that under this hypothesis NP = UP which, by Proposition 16, is equivalent with NP having a (P,FUP) proof system. Note that FUP $\subseteq$ NPSV.

Theorem 17. *(1)* SAT $\leq_m^P$ USAT $\implies$ NP *has a* $(\mathrm{coNP}, \mathrm{NPSV})$ *proof-system,*
(2) $F_{sat} \leq_{1\text{-}T}^{FP} f_{USat} \implies F_{sat} \cap \mathrm{FUP} \neq \emptyset \implies \mathrm{NP} = \mathrm{UP}$.

Proof. We show (2). The proof of (1) is an easy modification. We show the first implication, the second one is trivial. Let $t_1, t_2 \in \mathrm{FP}$ reduce F_{sat} to f_{USat}, i.e., the function $t_2(\varphi, f_{USat}(t_1(\varphi)))$ is in F_{sat}.

Consider the nondeterministic polynomial-time transducer M in Figure 1. We describe M on input φ.

We claim that M outputs exactly one satisfying assignment on some path, if $\varphi \in \mathrm{SAT}$, and makes no output, if $\varphi \notin \mathrm{SAT}$. To see this, let us first assume $\varphi \notin \mathrm{SAT}$. Since M will not find a satisfying assignment for φ, M will reject on all paths in line 4. Now, assume that $\varphi \in \mathrm{SAT}$. Then there exist satisfying assignments, say $\{\mathbf{a}_1, \ldots, \mathbf{a}_k\}$ for φ, for some $k \geq 1$. First, M will find all the $\mathbf{a}_i$'s on different computation paths and then compute $t_1(\varphi)$. Note that $t_1(\varphi)$ is in USAT, since otherwise $f_{USat}(t_1(\varphi)) = \perp$; but this is not possible since $t_2(\varphi, \perp)$ is not a satisfying assignment for φ as already checked in line 1. Hence, for each of the k paths where M found some $\mathbf{a}_i$, there will be exactly one path where M will find the unique satisfying assignment $\mathbf{b}$ of $t_1(\varphi)$. Now, by assumption, $t_2(\varphi, \mathbf{b}) = \mathbf{a}_j$ for some $j \in \{1, \ldots, k\}$. Finally, M will output $\mathbf{a}_j$ in line 9 on the unique path where $\mathbf{a}_j$ was found in line 3 and reject on all other paths in line 10. Hence, M is a FUP transducer for SAT. This proves the theorem.

```
M(φ)
if t_2(φ, ⊥) is a satisfying assignment of φ then output t_2(φ, ⊥)
else
    guess an assignment a for φ
    if a does not satisfy φ then reject
    else
        guess an assignment b for t_1(φ)
        if b does not satisfy t_1(φ) then reject
        else
            if t_2(φ, b) = a then output a
            else reject.
```

Fig. 1. FUP transducer computing satisfying assignments.

Since FUP $\subseteq$ NPSV, we conclude from [HNOS94] that the assumptions $F_{sat} \leq^{FP}_{1\text{-}T} f_{USat}$ and $F_{sat} \cap \mathrm{FUP} \neq \emptyset$ both imply that the Polynomial Hierarchy collapses. Note, however, that it is not known whether the assumption NP = UP implies a collapse of the Polynomial Hierarchy. Therefore it would be interesting to know whether some of the implications in Theorem 17 are in fact equivalences.

Corollary 18. *If $F_{sat} \leq^{FP}_{1\text{-}T} f_{USat}$, then the Polynomial Hierarchy collapses.*

Acknowledgments

We benefitted from discussions with Manindra Agrawal, Lance Fortnow, Toni Lozano, and Jacobo Tóran.

References

[BDG-I&II] J. Balcázar, J. Díaz, and J. Gabarró. Structural Complexity I & II. EATCS Monographs on Theoretical Computer Science, Springer-Verlag (1988, 1991)

[Be88] Beigel, R.: NP-hard sets are P-superterse unless R = NP. Technical Report 88-04, Dept. of Computer Science, The John Hopkins University (1988).

[BG82] Blass, A., Gurevich, Y.: On the unique satisfiability problem. Information and Control **55** (1982) 80-88

[BKT94] Buhrman, H., Kadin, J., Thierauf, T.: On functions computable with nonadaptive queries to NP. Proc. 9th Structure in Complexity Theory Conference (1994) 43-52

[CKR95] Chang, R., Kadin, J., Rohatgi, P.: On Unique Satisfiability and the threshhold behavior of randomized reductions. Journal of Computer and System Science **50** (1995) 359–373.

[Co71] Cook, S.: The Complexity of Theorem-Proving Procedures. Proc. 3rd ACM Symposium on Theory of Computing (1971) 151–158

[CT91] Chen, Z., Toda, S.: On the Complexity of Computing Optimal Solutions. International Journal of Foundations of Computer Science 2 (1991) 207-220

[CT93] Chen, Z., Toda, S.: An Exact Characterization of $FP_{\|}^{NP}$. Manuscript (1993)

[FHOS93] Fenner, S., Homer, S., Ogiwara, M., Selman, A.: On Using Oracles That Compute values. 10-th Annual Symposium on Theoretical Aspects of Computer Science, Springer Verlag LNCS **665** (1993) 398-407

[Fo94] Fortnow, L.: Personal Communication. In the plane to Madras (India) (December 7, 1994)

[H89] Hemachandra, L.: The strong exponential hierarchy collapses. Journal of Computer and System Sciences **39(3)** (1989) 299-322

[HNOS94] Hemaspaandra, L., Naik, A., Ogihara, M., Selman, A.: Finding Satisfying Assignments Uniquely Isn't so Easy: Unique Solutions Collapes the Polynomial Hierarchy. Algorithms and Compuatation, International Symposium ISAAC '94, Springer Verlag LNCS **834** (1994) 56-64

[HU79] Hopcroft, J., Ullman, J.: Introduction to Automata Theory, Languages, and Computation. Addison-Wesley (1979)

[IT89] Impagliazzo, R., Tardos, G.: Decision Versus Search Problems in Super-Polynomial Time. Proc. 30th IEEE Annual Symposium on Foundations of Computer Science (1989) 222-227

[Ka88] Kadin, J.: Restricted Turing Reducibilities and the Structure of the Polynomial Time Hierarchy. PhD thesis, Cornell University (1988)

[Kr86] Krentel, M.: The Complexity of Optimization Problems. Proc. 18th ACM Symposium on Theory of Computing (1986) 69-76

[Le73] Levin, L.: Universal Sorting Problems. Problems of Information Transmission **9** (1973) 265-266

[Og95] Ogihara, M.: Functions Computable with Limited Access to NP. Technical Report **538**, University of Rochester (1995)

[P84] Papadimitriou, C.: On the complexity of unique solutions. Journal of the ACM **31(2)** (1984) 392-400

[PY84] Papadimitriou, C., Yannakakis, M.: On the complexity of facets. Journal of Computer and System Sciences **28** (1984) 244-259

[PZ83] Papadimitriou, C., Zachos, D.: Two remarks on the power of counting. 6th GI Conference on TCS, Springer Verlag LNCS **145** (1983) 269-276

[Se94] Selman, A.: A taxonomy of complexity classes of functions. Journal of Computer and System Science **48** (1994) 357–381.

[To91] Toda S.: On polynomial-time truth-table reducibilities of intractable sets to P-selective sets. Mathematical Systems Theory **24** (1991) 69–82.

[VV86] Valiant, L., Vazirani, V.: NP is as easy as detecting unique solutions. Theoretical Computer Science **47(1)** (1986) 85-93

[W86] Wagner, K.: More complicated questions about maxima and minima and some closure properties of NP. Proc. 13th International Colloquium on Automata, Languages, and Programming (ICALP), Springer Verlag LNCS **226** (1986) 53-80

[W90] Wagner, K.: Bounded query classes. SIAM Journal on Computing **19(5)** (1990) 833-846

[WT93] Watanabe, O., Toda,S.: Structural Analysis on the Complexity of Inverse Functions. Mathematical Systems Theory **26** (1993) 203-214

This article was processed using the LaTeX macro package with LLNCS style