The Complexity of Generating and Checking Proofs of Membership

Harry Buhrman *
CWI Amsterdam

Thomas Thierauf †
Universität Ulm

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 investigate the complexity of generating proofs of membership for sets in NP and $\mathrm{P}^{\mathrm{NP}[O(\log n)]}$ and the complexity of checking these proofs. We show that such *proof-systems* can be used to distinguish the complexity of NP hard or $\mathrm{P}^{\mathrm{NP}[O(\log n)]}$ complete sets from that of USAT or UOCLIQUE, thereby giving some evidence that the answer to all the above questions is 'no'. Furthermore, we show the existence of an oracle relative to which $\mathrm{FP}_{\|}^{\mathrm{NP}}$ is not powerful enough to compute satisfying assignments for satisfiable Boolean formulas, answering an open question of Ogihara.

1 Introduction

NP is the class of sets that have easy to check proofs of membership. More formally, a set L is in NP if there is a polynomial time decidable set C (in symbols, $C \in \mathrm{P}$) and a polynomial p such that for all x

$$x \in L \iff \exists y\ (|y| \leq p(|x|))\ \ (x,y) \in C. \tag{1}$$

A string y witnessing $(x,y) \in C$ is called a proof of membership for $x \in L$. For a given y, one can check in polynomial time whether y is a proof of membership for x (since $C \in \mathrm{P}$). A fundamental question in computational complexity is the following: what is the computational difficulty to compute some polynomial-time checkable proof of membership for sets in NP?

As an upper bound, it is known that such proofs of membership 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 witness space using an appropriately chosen NP set as an oracle.

Is there a better way to compute a proof of membership? Consider the following subclasses of $\mathrm{FP}^{\mathrm{NP}}$.

*CWI. PO Box 94079, 1090 GB Amsterdam, The Netherlands. Part of this research was done while visiting the Univ. Politècnica de Catalunya in Barcelona. E-mail: buhrman@cwi.nl. Partially supported by the Dutch foundation for scientific research (NWO) through NFI Project ALADDIN, under contract number NF 62-376

†Abteilung Theoretische Informatik, Universität Ulm, Oberer Eselsberg, 89069 Ulm, Germany. E-mail: thierauf@informatik.uni-ulm.de

- $\mathrm{FP}_{\|}^{\mathrm{NP}}$, 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 in fact the same output.

Hence, we are especially asking whether it is possible to compute some polynomial-time checkable proof of membership for NP sets in $\mathrm{FP}_{\|}^{\mathrm{NP}}$ or in NPSV. Note that $\mathrm{NPSV} \subseteq \mathrm{FP}_{\|}^{\mathrm{NP}}$.

When considering the NP complete satisfiability problem SAT, a proof of membership for a given Boolean formula can for example be a satisfying assignment. Hemaspaandra et.al. [HNOS94] showed that one cannot compute satisfying assignments in NPSV, unless the Polynomial Hierarchy collapses, thereby answering the above question for one specific kind of membership proofs with respect to NPSV. Define the function class F_{sat} by

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

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

Theorem 1.1. [HNOS94] If $F_{sat} \cap \mathrm{NPSV} \neq \emptyset$, then the Polynomial Hierarchy collapses (to the second level).

However, it is not known whether one can extend Theorem 1.1 to other proofs of membership for SAT. More precisely, let C be any P set such that Equation (1) holds with $L = \mathrm{SAT}$, and define function class F_C analogously to F_{sat}. Then it is not known whether $F_C \cap \mathrm{NPSV} \neq \emptyset$ implies that the Polynomial Hierarchy collapses.

The question whether $\mathrm{FP}_{\|}^{\mathrm{NP}}$ is powerful enough to compute satisfying assignments is open as well. It is widely believed that this is not possibles. Fortnow [Fo94], extending a result of Watanabe and Toda [WT93], constructed an oracle relative to which $F_{sat} \cap \mathrm{FP}_{\|}^{\mathrm{NP}} \neq \emptyset$ and the Polynomial Hierarchy is infinite. On the other hand, we show that there is an oracle such that $F_{sat} \cap \mathrm{FP}_{\|}^{\mathrm{NP}} = \emptyset$, Hence, any answer with respect to $\mathrm{FP}_{\|}^{\mathrm{NP}}$ needs non-relativizing techniques.

Up to here, we asked for different function classes whether they can generate proofs of membership for NP sets that can be checked in polynomial time. But it is also interesting for some set L to fix some function class $\mathcal{F}$ and then to consider how difficult to check the proofs are that can be generated within $\mathcal{F}$ for L. This leads to the following definition.

Definition. 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.

In these terms, we asked above for the existence of $(\mathrm{P}, \mathrm{FP}_{||}^{\mathrm{NP}})$ or $(\mathrm{P}, \mathrm{NPSV})$ proof-systems for NP. However, it is even not known whether NP has a $(\mathrm{coNP}, \mathrm{FP}_{||}^{\mathrm{NP}})$ proof-system, that is whether proofs of membership for a NP set generated by a $\mathrm{FP}_{||}^{\mathrm{NP}}$ function can be checked in coNP.

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 but harder to generate. 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. This is the key to an application of the proof-systems as follows: let A and B be sets and let A have a $(\mathcal{C}, \mathcal{F})$ proof-systems. If B does not have a $(\mathcal{C}, \mathcal{F})$ proof-system, then we conclude that B is computational somehow more difficult than A.

In Section 3, we will apply this to 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 $\mathrm{P}^{\mathrm{NP}[O(\log n)]}$, i.e., it can be decided with logarithmically many queries to NP. Papadimitriou and Zachos [PZ83] asked whether UOCLIQUE is complete for $\mathrm{P}^{\mathrm{NP}[O(\log n)]}$. The problems whether a given graph as 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 open.

It is easy to see that UOCLIQUE, UOIS, and UOVC have $(\mathrm{coNP}, \mathrm{FP}_{||}^{\mathrm{NP}})$ proof-systems. On the other hand, it is not clear whether $\mathrm{P}^{\mathrm{NP}[O(\log n)]}$ complete sets have a $(\mathrm{coNP}, \mathrm{FP}_{||}^{\mathrm{NP}})$ proof-system. We show in Section 3 that for the known $\mathrm{P}^{\mathrm{NP}[O(\log n)]}$ complete sets, when fixing one of the two parameters of a proof-system to coNP and $\mathrm{FP}_{||}^{\mathrm{NP}}$, respectively, and taking a natural candidate for the other parameter, we either get a $(\mathrm{coNP}, \mathcal{F})$ proof-system, where $\mathcal{F}$ seems to be a more powerful class than $\mathrm{FP}_{||}^{\mathrm{NP}}$ (Theorem 3.8), or a $(\mathrm{D}^{\mathrm{P}}, \mathrm{FP}_{||}^{\mathrm{NP}})$ proof-system, where D^{P} [PY84] is the class of sets that are the difference of two NP sets. (Note that $\mathrm{coNP} \neq \mathrm{D}^{\mathrm{P}}$, unless NP = coNP.) We conjecture that no $\mathrm{P}^{\mathrm{NP}[O(\log n)]}$ complete set has a $(\mathrm{coNP}, \mathrm{FP}_{||}^{\mathrm{NP}})$ proof-system and hence, that UOCLIQUE , UOIS, and UOVC are *not* complete for $\mathrm{P}^{\mathrm{NP}[O(\log n)]}$.

As an example, let us consider the UMaxSAT problem, where, for a given set of clauses, it is asked whether there is a unique maximum subset of the clauses that is satisfiable by some assignment. UMaxSAT was shown to be $\mathrm{P}^{\mathrm{NP}[O(\log n)]}$ complete by Kadin. Let φ be a set of clauses (or a CNF formula). Whether a given assignment is indeed an assignment satisfying the unique maximum set of clauses of φ can be verified in coNP. Define the function class $F_{UMaxSat}$ by

$$f \in F_{UMaxSat} \iff f(\varphi) = \begin{cases} \text{some assignment that satisfies} & \\ \text{most of the clauses of } \varphi, & \text{if } \varphi \in \mathrm{UMaxSAT}, \\ \bot, & \text{otherwise.} \end{cases}$$

Hence, we have just argued that UMaxSAT has a $(\mathrm{coNP}, F_{UMaxSat})$ proof-system. Can some function from $F_{UMaxSat}$ be in $\mathrm{FP}_{||}^{\mathrm{NP}}$? We will see that this question hinges on our ability to compute satisfying assignments in $\mathrm{FP}_{||}^{\mathrm{NP}}$. Namely, we show that

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

This provides some evidence that no function from $F_{UMaxSat}$ might be in $\mathrm{FP}_{||}^{\mathrm{NP}}$. Moreover, relative to the above oracle where $F_{sat} \cap \mathrm{FP}_{||}^{\mathrm{NP}} = \emptyset$, also $F_{UMaxSat} \cap \mathrm{FP}_{||}^{\mathrm{NP}} = \emptyset$.

A proof of membership for φ computable in $\mathrm{FP}_{\|}^{\mathrm{NP}}$ is the sequence of indices of clauses that are in the unique maximum set of simultaneously satisfiable clauses, if $\varphi \in$ UMaxSAT. But it turns out that the problem whether some given sequence of indices in fact is the desired one is D^{P} complete, and hence not in coNP, unless NP = coNP.

We want mention that $F_{UMaxSat}$ can be used to characterize $\mathrm{FP}_{\|}^{\mathrm{NP}}$. Namely, combining Theorem 3.8 below with a result from [BKT94], it follows that $F_{UMaxSat}$ is *quasi-complete* [BKT94] for $\mathrm{FP}_{\|}^{\mathrm{NP}}$. That is, with respect to some reduction type, 1) every function in $\mathrm{FP}_{\|}^{\mathrm{NP}}$ is reducible to $F_{UMaxSat}$, and 2) if some function g reduces to $F_{UMaxSat}$, then g is in $\mathrm{FP}_{\|}^{\mathrm{NP}}$. However note that the reduction used in [BKT94] does not necessairily imply that $\mathrm{FP}_{\|}^{\mathrm{NP}} \cap F_{UMaxSat}$ is not empty. Moreover, relative to an oracle where $\mathrm{FP}_{\|}^{\mathrm{NP}} \cap F_{UMaxSat} = \emptyset$, we have that quasi-completeness differs from the standard completeness notion for $\mathrm{FP}_{\|}^{\mathrm{NP}}$.

As a side effect of our considerations we find some more examples of sets and functions complete for D^{P}, $\mathrm{P}^{\mathrm{NP}[O(\log n)]}$, and $\mathrm{FP}_{\|}^{\mathrm{NP}}$. Thereby, we continue work of Wagner [W86, W90], who studied the behaviour of NP complete sets when applying some predicate to the solution spaces of these sets.

In Section 4, we show that $\mathrm{P}^{\mathrm{NP}[O(\log n)]}$ has a $(\mathrm{coNP}, \mathrm{FP}_{\|}^{\mathrm{NP}})$ proof-system if and only if NP has a $(\mathrm{coNP}, \mathrm{FP}_{\|}^{\mathrm{NP}})$ proof-system (Theorem 4.2), enabling us to extend the above conjecture to NP. Furthermore, we connect the question whether NP has a $(\mathrm{coNP}, \mathrm{FP}_{\|}^{\mathrm{NP}})$ proof-system with the question whether the Unique Satisfiability (USAT) problem is complete for D^{P} which is a open question in computational complexity for a long time now.

2 Preliminaries

We follow the standard definitions and notations in computational complexity theory (see, e.g., [BDG88, BDG91, 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. FP is the class of polynomial-time computable functions. By taking 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 ask questions to. 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 all the queries in parallel, i.e., all queries have to be made before any computation that uses the answers to the queries. In fact, for the language classes these two restrictions give the same class, i.e., $\mathrm{P}^{\mathrm{NP}[O(\log n)]} = \mathrm{P}_{\|}^{\mathrm{NP}}$ [H89]. For the function classes, we only have a inclusion, namely

$\mathrm{FP}^{\mathrm{NP}[O(\log n)]} \subseteq \mathrm{FP}_{\|}^{\mathrm{NP}}$.

The *Polynomial Hierarchy* [St77] is defined as $\mathrm{NP} \cup \mathrm{NP}^{\mathrm{NP}} \cup \mathrm{NP}^{\mathrm{NP}^{\mathrm{NP}}} \cup \ldots$.

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. For two sets A and B, we say that A *many-one reduces to* B, denoted by $A \leq_m^P B$, if there exists a function $f \in \mathrm{FP}$ such that for all x, we have $x \in A \iff f(x) \in B$. B is *hard for some class of sets* $\mathcal{C}$, if every set $A \in \mathcal{C}$ is many-one reducible to B. B is *complete for* $\mathcal{C}$, if $B \in \mathcal{C}$ and B is hard for $\mathcal{C}$.

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

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

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.

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

This is sort of 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 have only to be some FP transducers that, no matter which function from G is used, compute *some* function in F.

3 Proof-Systems for Sets in $\mathrm{P}^{\mathrm{NP}[O(\log n)]}$

There are many examples for NP or PSPACE complete sets in the literature, but by far not so many examples of complete sets for classes between NP and PSPACE (in case NP $\neq$ PSPACE only!). Krentel [Kr86] showed some functions to be complete for $\mathrm{FP}^{\mathrm{NP}}$ and $\mathrm{FP}^{\mathrm{NP}[O(\log n)]}$, respectively. Examples of such functions are

- the length of the optimal Traveling Salesman tour,

which is complete for $\mathrm{FP}^{\mathrm{NP}}$,

- the size of the largest clique of a graph, and
- the maximum number of simultaneously satisfiable clauses of a CNF Boolean formula

which are complete for $\mathrm{FP}^{\mathrm{NP}[O(\log n)]}$.

Papadimitriou turned these functions into decision problems by asking whether those optimal solutions are *unique*. The intuition behind this might be that in order to decide uniqueness there is no way around to compute, somehow implicit, the underlying function. Hence, these decision problems are expected to be hard for the corresponding complexity classes. In fact, Papdimitriou [P84] showed that the

- Unique Optimal Traveling Salesman Problem (UOTSP) is complete for P^{NP}.

Papadimitriou and Zachos [PZ83] asked whether the Unique Optimal Clique problem (UOCLIQUE) is complete for $\mathrm{P}^{\mathrm{NP}[O(\log n)]}$, and this is still an open problem. Since Unique Optimal Independent Set (UOIS) and Unique Optimal Vertice Cover (UOVC) are clearly many-one equivalent to UOCLIQUE, this question can be extended to these two problems. In contrast, Krentel [Kr86] and Wagner [W86] showed that Odd-CLIQUE, i.e., the problem to determine whether the maximum clique of a graph has an odd number of vertices, is complete for $\mathrm{P}^{\mathrm{NP}[O(\log n)]}$.

On the other hand, Kadin [Ka88] showed that

- the problem whether the largest set of simultaneously satisfiable clauses of a CNF Boolean formula is unique (UMaxSAT) is complete for $\mathrm{P}^{\mathrm{NP}[O(\log n)]}$.

Definition. 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.

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. Clearly, when we restrict UMaxSAT even further by requiring that there is only one such assignment, then the reduction works.

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

Hence, we have seen that UMaxASAT $\leq_m^P$ UOCLIQUE. But in fact, these two problems are equivalent. We introduce the following notation.

Notation. For some formula α and some integer $k \geq 0$, let $\alpha^k = \bigwedge_{i=1}^{k} \alpha$, that is, α^k is the conjunction of k times α.

Theorem 3.1. UMaxASAT $\equiv_m^P$ UOCLIQUE.

Proof. We show UOCLIQUE $\leq_m^P$ UMaxASAT. Let G be graph with n vertices ($n \geq 1$). Our first step is to construct an NP machine M that finds all cliques in G (of any size). That is, on input G, M first guesses a nonempty subset C of the vertices of G and accepts if C is a clique in G, and rejects otherwise.

Now, we apply Cook's reduction on M and G, together with some appropriate time bound for M, getting a CNF formula φ. Since any (nonempty) graph as cliques of size one, M accepts G in the usual nondeterministic sense, and hence $\varphi \in$ SAT. Recall that in Cook's reduction any nondeterministic computation path of M corresponds to an assignment of the variables of φ and furthermore the path is accepting if and only if the assignment satisfies φ.

It is not hard to see that we can assume that M works in such a way that φ will contain variables, say $v_1, \ldots, v_n$, that correspond to the set C of vertices chosen by M in the sense that

the corresponding assignment sets $v_i = 1$ if vertice $i \in C$ and $v_i = 0$ otherwise. Define

$$f(G) \quad = \quad \varphi^n \ \wedge \ \bigwedge_{i=1}^{n} v_i.$$

We claim that f reduces UOCLIQUE to UMaxASAT. Let **a** be an assignment that satisfies most of the clauses of $f(G)$. **a** will clearly satisfy φ, because otherwise at least n clauses of $f(G)$ are not satisfied, but by satisfying φ at most $n-1$ clauses of $f(G)$ are not satisfied. Hence, **a** will set the maximum number of the variable v_i to one such that φ is still satisfiable. Therefore, every maximum clique in G corresponds to exactly one assignment satisfying most of the clauses of $f(G)$, and thus, we have $G \in$ UOCLIQUE if and only if $f(G) \in$ UMaxASAT. □

Therefore, also for UMaxASAT we don't know whether it is complete for $\mathrm{P}^{\mathrm{NP}[O(\log n)]}$. Note that UMaxASAT is sort of related to USAT [BG82], the set ot Boolean formulas having *exactly one* satisfying assignment. UMaxASAT can be seen as a generalization of USAT to unsatisfiable formulas. USAT is in D^{P}, since it can be written as the difference of the two NP sets requiring at least one and at least two satisfying assignments of a Boolean formula, respectively. USAT is coNP hard, but it is neither known to be complete for D^{P} nor known to belong to coNP. It is widely conjectured that USAT is kind of an "intermediate" problem with respect to coNP and D^{P}, i.e., that it is indeed not complete for D^{P} and does not belong to coNP. It is easy to see that USAT is complete for D^{P} if and only if it is hard for NP. Also for UMaxASAT it is an open problem whether it is NP hard. For later reference, we show the just mentioned hardness results.

Theorem 3.2. [BG82, Ka88]

(1) USAT, UMaxASAT, and UMaxSAT are hard for coNP,

(2) UMaxSAT is hard for NP.

Proof. (1) We reduce $\overline{\mathrm{SAT}}$ to the three sets. Let $\varphi = \varphi(x_1, \ldots, x_n)$ be some CNF Boolean formula and let z be a new variable. Define

$$\psi' \quad = \quad (\bigwedge_{i=1}^{n} x_i \wedge z) \ \vee \ (\varphi \wedge \overline{z}).$$

ψ' is satisfiable and has exactly one satisfying assignment if and only if $\varphi \in \overline{\mathrm{SAT}}$. By applying de Morgan's law, we can transform ψ' into an equivalent CNF formula ψ. Then we have $\varphi \in \overline{\mathrm{SAT}} \iff \psi \in \mathrm{USAT} \iff \psi \in \mathrm{UMaxASAT}$.

To reduce $\overline{\mathrm{SAT}}$ to UMaxSAT, we further extend ψ to Φ

$$\Phi \quad = \quad \psi \ \wedge \ \bigwedge_{i=1}^{n} x_i \ \wedge \ \bigwedge_{i=1}^{n} \overline{x_i}.$$

For every assignment, from the last $2n$ clauses of Φ there are exactly n satisfied, and each assignment is reflected by a different set of those clauses. Thus, $\varphi \in \overline{\mathrm{SAT}} \iff \Phi \in$ UMaxSAT.

(2) We reduce SAT to UMaxSAT. Let φ be some CNF Boolean formula having m clauses $C_1, \ldots, C_m$, i.e., $\varphi = \bigwedge_{i=1}^{m} C_i$, and let z be a new variable. We will show that the following function h is a many-one reduction from SAT to UMaxSAT.

$$h(\varphi) \quad = \quad (\varphi \vee z) \ \wedge \ (\varphi \vee \overline{z}) \quad = \quad (\bigwedge_{i=1}^{m} (C_i \vee z)) \ \wedge \ (\bigwedge_{i=1}^{m} (C_i \vee \overline{z})).$$

If φ is in SAT, then also $h(\varphi)$ is in SAT and therefore in UMaxSAT. On the other hand, if φ is not in SAT, then also $h(\varphi)$ is not in SAT, and furthermore, if at most, say, k clauses of φ are satisfiable for some $k < m$, then at most $m + k$ clauses of $h(\varphi)$ are satisfiable, but, by changing the value of variable z, two different sets of $m + k$ clauses. Therefore, $h(\varphi)$ is not in UMaxSAT. □

It is not hard to see that for any set $L \in \mathrm{P}^{\mathrm{NP}[O(\log n)]}$ there is a set $A \in \mathrm{D}^{\mathrm{P}}$ such that L is *disjunctively reducible to* A, i.e., for any $x \in \Sigma^*$ one can compute in polynomial time strings $x_1, \ldots, x_m$ such that $x \in L$ if and only if there exists an $i \in \{1, \ldots, m\}$ such that $x_i \in A$. Furthermore, considering the known examples, when L is complete for $\mathrm{P}^{\mathrm{NP}[O(\log n)]}$, then A is complete for D^{P}. However, we observe that UMaxASAT disjunctively reduces to USAT. Since USAT is conjectured to be not complete for D^{P}, the above observation motivates our conjecture that *UOCLIQUE, UOIS, UOVC, and UMaxASAT are not complete for* $\mathrm{P}^{\mathrm{NP}[O(\log n)]}$. Below, we will give more arguments to justify our conjecture.

Theorem 3.3. UMaxASAT disjunctively reduces to USAT.

Proof. We define the following NP A set. For any CNF formula φ and any integer $k \geq 0$,

$$(\varphi, k) \in A \quad \Longleftrightarrow \quad \text{there is an assignment satisfying at least k clauses of } \varphi.$$

Since A is in NP, we can reduce A to SAT via the Cook reduction [Co71], say $c \in \mathrm{FP}$.[1]

Note that the Cook reduction is parsimonious, i.e., for any (φ, k), the number of assignments satisfying at least k clauses of φ will be equal to the number of satisfying assignments of $c(\varphi, k)$. Therefore, we have $\varphi \in$ UMaxASAT if and only if there is a k such that $c(\varphi, k) \in$ USAT. □

In the rest of this section, 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.

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

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, coNP complete.

[1] More precisely, for any polynomial-time nondeterministic machine M Cook's reduction maps a given string x to some Boolean formula $\gamma_{M,x}$ such that $x \in L(M) \iff \gamma_{M,x} \in$ SAT. Hence, by saying that A is reduced to SAT via the Cook reduction, we implicitly refer to some machine that accepts A.

Proposition 3.4. UMaxASAT and UOCLIQUE have a $(\mathrm{coNP}, \mathrm{FP}_{\|}^{\mathrm{NP}})$ proof-system.

By essentially the same argument as above for UMaxASAT, we have

Proposition 3.5. UMaxSAT has a $(\mathrm{coNP}, F_{UMaxSat})$ proof-system.

However, the complexity of $F_{UMaxSat}$ is not at all clear. Especially, it is not known whether there is some function in $F_{UMaxSat}$ that is computable with parallel queries to NP, i.e., whether $F_{UMaxSat} \cap \mathrm{FP}_{\|}^{\mathrm{NP}} \neq \emptyset$. We will show that this question is equivalent with the problem of whether one can compute satisfying assignments in $\mathrm{FP}_{\|}^{\mathrm{NP}}$.

In [BKT94], the function class F_{zero} is introduced.

$$f \in F_{zero} \iff f(\varphi) = \begin{cases} \text{some satisfying assignment of } \varphi \text{ having} & \\ \text{the maximum number of zeros,} & \text{if } \varphi \in \mathrm{SAT}, \\ \bot, & \text{otherwise.} \end{cases}$$

The idea of F_{zero} is to filter out satisfying assignments that might be easier to compute than others. This intuition is somehow justified by the following result.

Theorem 3.6. [BKT94] $F_{sat} \cap \mathrm{FP}_{\|}^{\mathrm{NP}} \neq \emptyset \iff F_{zero} \cap \mathrm{FP}_{\|}^{\mathrm{NP}} \neq \emptyset$.

We show that $F_{UMaxSat}$ is equivalent to F_{zero}, and therefore, Theorem 3.6 holds also with F_{zero} replaced $F_{UMaxSat}$. We interpret this result as some evidence that it is indeed *not* possible to compute some function in $F_{UMaxSat}$ in $\mathrm{FP}_{\|}^{\mathrm{NP}}$, and hence, that $F_{UMaxSat}$ is a more powerful class than $\mathrm{FP}_{\|}^{\mathrm{NP}}$.

We need the following lemma stating that UMaxSAT have kind of an OR-function: one can combine several instances $\varphi_1, \ldots, \varphi_k$ that have some additional property into one instance ψ such that $\exists i : \varphi_i \in \mathrm{UMaxSAT} \iff \psi \in \mathrm{UMaxSAT}$. The additional property we need requires that if some φ_i is in UMaxSAT then there is also φ_{i_0} in UMaxSAT that is the unique best one with respect to the minimum number of clauses remaining unsatisfied by a any assignment.

Lemma 3.7. **[Ka88]** Let $\varphi_1, \ldots, \varphi_k$ be CNF formulas and let u_i be the minimum number of simultaneously *unsatisfiable* clauses of φ_i. One can construct in polynomial time a CNF formula ψ with the following properties.

(i) If there is an i_0 such that $u_{i_0} < u_i$ for all $i \neq i_0$, then $\varphi_{i_0} \in \mathrm{UMaxSAT} \iff \psi \in \mathrm{UMaxSAT}$,

(ii) otherwise, ψ will not be in UMaxSAT.

Furthermore, in case of (i), one can easily obtain an assignment satisfying most of the clauses of φ_{i_0} from such a one of ψ.

Proof. Let $y_1, \ldots, y_k$ be new variables and let $\mathrm{Exact}(y_1, \ldots, y_k)$ be the formula that is true if and only if exactly one of its variables is true. That is,

$$\mathrm{Exact}(y_1, \ldots, y_k) = (\bigvee_{i=1}^{k} y_i) \wedge \bigwedge_{1 \leq i < j \leq k} (\overline{y_i} \vee \overline{y_j}).$$

Let m be the maximum of the number of clauses of $\varphi_1, \ldots, \varphi_k$. We will show that the following formula ψ has the claimed properties.

$$\psi \quad = \quad (\bigwedge_{i=1}^{k} (\varphi_i \vee \overline{y_i})) \ \wedge \ \text{Exact}^{m+1}.$$

(Recall that the exponent at a formula means that we conjunct that formula so many times.) Let i_0 be the index required to exist in (i) and let $\mathbf{a}$ be an assignment that sets $y_{i_0} = 1$, $y_i = 0$, for all $i \neq i_0$, coincides with a maximal assignment for φ_{i_0} on those variables, and arbitrarily assigns all other variables. Then $\mathbf{a}$ satisfies Exact and hence, all clauses of ψ up to u_{i_0} many are satisfied by $\mathbf{a}$. We claim that no assignment can satisfy more clauses of ψ than $\mathbf{a}$. This is because if an assignment $\mathbf{b}$ does not satisfy Exact, then at least one clause of Exact is not satisfied and therefore, at least $m+1$ clauses of ψ are not satisfied by $\mathbf{b}$. Since $u_{i_0} < m+1$, $\mathbf{a}$ satisfies more clauses of ψ than $\mathbf{b}$. Moreover, of all assignments satisfying Exact, $\mathbf{a}$ is clearly optimal. □

Theorem 3.8. $F_{UMaxSat} \equiv^{FP}_{1\text{-}T} F_{zero}$.

Proof. We first show $F_{UMaxSat} \leq^{FP}_{1\text{-}T} F_{zero}$. For any given formula φ, we will construct in polynomial time a formula $\psi \in$ SAT such that from any satisfying assignment of ψ having the maximum number of zeros, we can decide whether $\varphi \in$ UMaxSAT, and furthermore, if this is the case, compute an assignment satisfying most of the clauses of φ.

We define two NP sets A and B as follows. For a CNF formula φ and some integer $i \geq 0$,

$$\begin{aligned}
(\varphi, i) \in A \quad &\Longleftrightarrow \quad \text{there is an assignment satisfying at least } i \text{ clauses of } \varphi, \\
(\varphi, i) \in B \quad &\Longleftrightarrow \quad \text{there are two assignments, each satisfying at least } i \text{ clauses of } \varphi, \\
&\qquad\quad \text{but different sets of clauses.}
\end{aligned}$$

Since A and B are in NP, we can reduce them to (CNF) SAT using the Cook reduction, say via c_A, $c_B \in$ FP (see footnote on page 8). Let us fix some φ and suppose φ has m clauses.

We define formulas $\varphi_0, \ldots, \varphi_{2m+1}$ as follows. For $i = 0, \ldots, m$,

$$\begin{aligned}
\varphi_{2i} \quad &= \quad c_A(\varphi, i), \\
\varphi_{2i+1} \quad &= \quad c_B(\varphi, i).
\end{aligned}$$

We assume that the φ_j's have pairwise disjoint sets of variables. Note the following property of the Cook reduction: from any satisfying assignment of some φ_{2i}, one can compute in polynomial time an assignment of φ that satisfies at least i clauses of φ. Note furthermore that when $\varphi_j \in$ SAT for some $j \geq 0$, then also $\varphi_k \in$ SAT for all $k \leq j$, and φ is in UMaxSAT if and only if the largest j such that φ_j is in SAT is even.

Now, let n_j be the number of variables of φ_j for $j = 0, \ldots, 2m+1$. Let furthermore $n = 1 + \max\{\, n_j \mid j = 0, \ldots, 2m+1 \,\}$ and $M = 2m+1$. We define

$$\psi \quad = \quad \bigvee_{j=0}^{M} (\varphi_j \ \wedge \ \bigwedge_{k=1}^{(M-j)n} z_k),$$

where $z_1, \ldots, z_{nM}$ are new variables.

We have $\psi \in$ SAT, since $\varphi_0 \in$ SAT. Since ψ is a disjunction, it is satisfied as soon as one of the terms $\tau_j = \varphi_j \wedge \bigwedge_{k=1}^{(M-j)n} z_k$ is satisfied. Let t_j be the maximum number of zeros a satisfying assignment of ψ can have when term τ_j is satisfied.

Claim. $\varphi_j \in \text{SAT} \implies t_{j-1} < t_j$, for all $j \geq 1$.

To prove the claim let us fix some j and assume $\varphi_j \in$ SAT. Let m_j denote the maximum number of zeros a satisfying assignment of φ_j can have.

In order to satisfy τ_j, we can get m_j zeros from the variables of φ_j and jn zeros from variables z_k, because we have to assign 1 to variables $z_1, \ldots, z_{(M-j)n}$. To all other variables, we can assign zero. Let $N = \sum_{k=0}^{M} n_k$, then we have

$$t_j \quad = \quad (N - n_j) + m_j + jn.$$

Since $0 \leq n_j - m_j < n$, we have $N + (j-1)n < t_j \leq N + jn$. Thus, we can conclude that $t_{j-1} < t_j$. This proves the claim.

It follows that any satisfying assignment of ψ having the maximum number of zeros will achieve this by satisfying term τ_{j_0}, where j_0 is the largest j such that $\varphi_j \in$ SAT. By the definition of sets A and B, we have $\varphi \in$ UMaxSAT if and only if j_0 is even, and furthermore, in this case, from an assignment of ψ having the maximum number of zeros, we can compute an assignment of φ satisfying most of its clauses.

For the reverse direction, we show $F_{zero} \leq_{1\text{-}T}^{FP} F_{UMaxSat}$. For any given formula φ, we will construct in polynomial time a formula ψ such that $\varphi \in$ SAT if and only if $\psi \in$ UMaxSAT, and furthermore, if $\varphi \in$ SAT, from any assignment satisfying most of the clauses of ψ, we can compute a satisfying assignment of φ having the maximum number of zeros.

Let h be the reduction from SAT to UMaxSAT as in Theorem 3.2. That is,

$$h(\varphi) \quad = \quad (\varphi \vee z) \ \wedge \ (\varphi \vee \overline{z}),$$

for some new variable z. We will use the following property of h: when φ is in SAT, then from any given assignment that satisfies most of the clauses of $h(\varphi)$, we can read off a satisfying assignment of φ.

We define an NP set A as follows. For a formula φ and some integer $i \geq 0$,

$$(\varphi, i) \in A \quad \iff \quad \text{there is a satisfying assignment of } \varphi \text{ having at least } i \text{ zeros.}$$

Since A is in NP, we can reduce it to (CNF) SAT using the Cook reduction, say via $c \in$ FP. Let us fix some $\varphi = \varphi(x_1, \ldots, x_n)$. For all $i \in \{0, \ldots, n\}$, we first reduce the pairs (φ, i) to CNF SAT via c, and then further to UMaxSAT via h defined above. That is, we consider the formulas $\varphi_i = h \circ c(\varphi, i)$. W.l.o.g., we can assume that all formulas φ_i have the same number m of clauses.

Suppose that φ is in SAT and let i_0 be the maximum number of zeros any satisfying assignment of φ can have. By definition of the formulas φ_i, it follows that i_0 is the maximum $i \in \{0, \ldots, n\}$ such that $\varphi_i \in$ UMaxSAT, and furthermore, by our reduction functions c and h, from any given assignment that satisfies most of the clauses of φ_{i_0}, we can easily get a satisfying assignment of φ having i_0 zeros. In what follows, we will combine the formulas φ_i, for

$i = 0, \ldots, n$, into one formula $\psi \in$ UMaxSAT in such a way that from an assignment that satisfies most of the clauses of ψ, we can read off an assignment that satisfies most of the clauses of φ_{i_0}. This accomplishes the reduction.

For $i = 0, \ldots, n$ and new variables $z_0, \ldots, z_n$ and $y_0, \ldots, y_n$, and Exact $=$ Exact$(y_0, \ldots, y_n)$ as defined in the proof of Lemma 3.7, we define

$$\begin{aligned} \gamma_i &= \varphi_i^{n+1} \ \wedge \ z_i^{n-i+1} \ \wedge \ \overline{z_i}^{n-i}, \\ \psi &= (\bigwedge_{i=0}^{n} (\gamma_i \vee \overline{y_i})) \ \wedge \ \text{Exact}^c, \end{aligned}$$

where $c = m(n+1) + 2n + 2$.

It remains to show that the assumptions of Lemma 3.7 are fullfilled. Note first that γ_i has $m(n+1) + 2(n-i) + 1$ clauses, so that c is strictly greater than the number of clauses of γ_i, for $i = 0, \ldots, n$. Now, let u_i be the minimum number of simultaneously *unsatisfiable* clauses of γ_i. For i_0 as defined above, we claim that u_{i_0} is strictly less than any other u_i.

Claim. $u_{i_0} < u_i$, for all $i \neq i_0$.

To prove the claim, assume first that $i \leq i_0$. Then $\varphi_i \in$ SAT and hence, any assignment can satisfy at most $m(n+1) + n - i + 1$ clauses of γ_i by taking a satisfying assignment for φ_i and setting $z_i = 1$. Therefore, we have $u_i = n - i$ and hence, $u_{i_0} < u_i$, for all $i < i_0$.

Now, suppose $i > i_0$. Then $\varphi_i \notin$ SAT. Hence, at least one clause of φ_i is not satisfied by any assignment and therefore, at least $(n+1) + n - i$ clauses of γ_i. Thus, we have $u_i \geq n + 1 > u_{i_0}$. This proves the claim.

By Lemma 3.7, if $\varphi \in$ SAT then $\psi \in$ UMaxSAT and, from an assignment satisfying most of the clauses of ψ, we can get a satisfying assignment of φ having the maximum number of zeros. Finally, we note that if φ is not in SAT, then all formulas φ_i are not in UMaxSAT, and therefore ψ is not in UMaxSAT. Thus, $F_{UMaxSat}$ will return $\perp$ in this case which is also the value of F_{zero}. □

Corollary 3.9. $F_{UMaxSat} \cap \text{FP}_{\|}^{\text{NP}} \neq \emptyset \iff F_{sat} \cap \text{FP}_{\|}^{\text{NP}} \neq \emptyset$.

In view of these results, it seems as we need a more powerful function class to compute proofs for UMaxSAT than for UMaxASAT in order to have these proofs coNP checkable. On the other hand, when we restrict the proof generating functions to be in $\text{FP}_{\|}^{\text{NP}}$, how hard are those proofs to check? We cannot give an absolute answer to this question, however, when taking a natural proof generating function in $\text{FP}_{\|}^{\text{NP}}$ for any of the currently known $\text{P}^{\text{NP}[O(\log n)]}$ complete sets, then checking these proofs turns out to be D^{P} complete. Below, we give several examples.

Clearly, the existence of a $(\text{D}^{\text{P}}, \text{FP}_{\|}^{\text{NP}})$ proof-system for one $\text{P}^{\text{NP}[O(\log n)]}$ complete set implies that all sets $L \in \text{P}^{\text{NP}[O(\log n)]}$ have such a proof-system, since one can reduce L to the complete set and then use its proof-system for L. Similarly, if one can find a $(\mathcal{C}, \text{FP}_{\|}^{\text{NP}})$ proof-system for a $\text{P}^{\text{NP}[O(\log n)]}$ complete set, for some subclass $\mathcal{C}$ of D^{P}, then this carries over to the other sets in $\text{P}^{\text{NP}[O(\log n)]}$. In our examples below, we take several sets complete for $\text{P}^{\text{NP}[O(\log n)]}$ and demonstrate for each of them *independently* that a natural proof generating function in $\text{FP}_{\|}^{\text{NP}}$ can be checked in D^{P}, but not in NP or coNP unless NP $=$ coNP.

Example 1. 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

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

$h_{UMaxSAT}$ 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. We conclude that UMaxSAT has a $(\mathrm{D}^{\mathrm{P}}, \mathrm{FP}_{\|}^{\mathrm{NP}})$ proof-system.

Theorem 3.10. (1) $h_{UMaxSAT}$ is $\mathrm{FP}_{\|}^{\mathrm{NP}}$ complete,

(2) $C = \{\, (\varphi, y) \mid \varphi \in \text{UMaxSAT} \text{ and } h_{UMaxSAT}(\varphi) = y \,\}$ is D^{P} complete.

Proof. (1). $h_{UMaxSAT}$ is in $\mathrm{FP}_{\|}^{\mathrm{NP}}$ via some transducer M that, for some given CNF formula φ, first finds out the maximum number k of simultaneously satisfiable clauses of φ, then checks whether φ is in UMaxSAT, and , if yes, M can find out for each clause C whether there exists an assignment satisfying k clauses of φ, including C. Note that in fact all the queries of M can be asked in parallel. We leave the details to the reader.

Next, we show that $h_{UMaxSAT}$ is hard for $\mathrm{FP}_{\|}^{\mathrm{NP}}$. Let $f \in \mathrm{FP}_{\|}^{\mathrm{NP}}$ via some polynomial-time transducer T that, on a given input x, asks polynomially many queries $\varphi_1, \ldots, \varphi_k$ to its oracle SAT. As a first step, we show how to reduce a single query to UMaxSAT.

Lemma 3.11. For any CNF formula φ there is a formula $\psi \in$ UMaxSAT such that from $h_{UMaxSAT}(\psi)$ one can decide in polynomial time whether φ is in SAT.

Proof. We combine the reductions from SAT and $\overline{\text{SAT}}$ to UMaxSAT from Theorem 3.2. That is, for $\varphi = \varphi(x_1, \ldots, x_n)$ and new variables y and z, we define

$$\begin{aligned} \varphi_0 &= \left((\bigwedge_{i=1}^{n} x_i \wedge z) \ \vee \ (\varphi \wedge \overline{z}) \right) \ \wedge \ \bigwedge_{i=1}^{n} x_i \ \wedge \ \bigwedge_{i=1}^{n} \overline{x_i}, \\ \varphi_1 &= (\varphi \vee y) \ \wedge \ (\varphi \vee \overline{y}). \end{aligned}$$

Exactly one of φ_0 and φ_1 will be in UMaxSAT. Let u_0 and u_1 be the minimum number of simultaneously unsatisfiable clauses of φ_0 and φ_1, respectively. We have $u_0 = n$ and $u_1 = 0$ if $\varphi \in$ SAT, and $u_1 \ge 1$ if $\varphi \notin$ SAT.

For new variables y_0, y_1, $\text{Exact} = \text{Exact}(y_0, y_1) = (y_0 \vee y_1) \wedge (\overline{y_0} \vee \overline{y_1})$, and large enough c let

$$\psi \quad = \quad (\varphi_0 \vee \overline{y_0}) \ \wedge \ (\varphi_1^{n+1} \vee \overline{y_1}) \ \wedge \ \text{Exact}^c.$$

Now, by Lemma 3.7, any assignment satisfying most of the clauses of ψ will set $y_0 = 1$ if $\varphi \in$ SAT, and $y_1 = 1$ if $\varphi \notin$ SAT. Thus, ψ has the properties we claimed. □

We apply Lemma 3.11 to all the queries $\varphi_1, \ldots, \varphi_k$ of transducer T on input x, getting formulas $\psi_1, \ldots, \psi_k \in$ UMaxSAT, respectively. Now, define $\Phi = \bigwedge_{i=1}^{k} \psi_i$. Clearly, $\Phi \in$ UMaxSAT

and, getting $h_{UMaxSAT}(\Phi)$, one can decide membership of all queries $\varphi_1, \ldots, \varphi_k$ in SAT, and hence, compute $f(x)$ in polynomial time. This shows $f \leq^{FP}_{1\text{-}T} h_{UMaxSAT}$.

(2). We define two NP sets L_1 and L_2 such that $C = L_1 - L_2$. For any CNF formula $\varphi = \bigwedge_{i=1}^{m} C_i$ and any sequence $1 \leq i_1 < \cdots < i_k \leq m$,

$$\begin{aligned}
(\varphi, (i_1, \ldots, i_k)) \in L_1 \quad &\Longleftrightarrow \quad \text{there exist an assignment satisfying } C_{i_1}, \ldots, C_{i_k}, \\
(\varphi, (i_1, \ldots, i_k)) \in L_2 \quad &\Longleftrightarrow \quad \text{there exist an assignment satisfying a set of } k \text{ clauses of } \varphi \\
&\qquad\qquad \text{different from } \{C_{i_1}, \ldots, C_{i_k}\}.
\end{aligned}$$

To show that C is hard for D^{P}, we reduce (SAT,UNSAT) to C, i.e., the set of all pairs of (CNF) formulas (φ_1, φ_2) such that $\varphi_1 \in$ SAT and $\varphi_2 \notin$ SAT, which is known to be complete for D^{P}. Let us fix a pair (φ_1, φ_2) with disjoint sets of variables. We use the reductions from SAT and $\overline{\text{SAT}}$ to UMaxSAT as in Theorem 3.2. That is, for $\varphi_2 = \varphi_2(x_1, \ldots, x_n)$ and new variables y and z, we define

$$\begin{aligned}
\psi_1 &= (\varphi_1 \vee y) \ \wedge \ (\varphi_1 \vee \overline{y}), \\
\psi_2 &= \left((\bigwedge_{i=1}^{n} x_i \wedge z) \ \vee \ (\varphi_2 \wedge \overline{z}) \right) \ \wedge \ \bigwedge_{i=1}^{n} x_i \ \wedge \ \bigwedge_{i=1}^{n} \overline{x_i}.
\end{aligned}$$

Let m_1 and m_2 be the number of clauses of ψ_1 and ψ_2, respectively when being in CNF and let $\bigwedge_{i=1}^{n} x_i \ \wedge \ \bigwedge_{i=1}^{n} \overline{x_i}$ be the last $2n$ clauses of ψ_2 (in this ordering). Then we have $(\varphi_1, \varphi_2) \in$ (SAT,UNSAT) $\Longleftrightarrow$ $(\psi_1 \wedge \psi_2, (1, \ldots, m_1 + m_2 - n)) \in C$. □

Example 2. Chen and Toda [CT93] showed that a class of functions complete for $\mathrm{FP}^{\mathrm{NP}}_{\|}$ can be derived by using the supremum function. For a nonempty set $S \subseteq \Sigma^n$, the *supremum of* S is the string $s \in \Sigma^n$, where the ith bit of s is a one if and only if there is a string in S having a one at position i. The supremum of the empty set is undefined.

Roughly speaking, Chen and Toda showed that for any of the known NP complete sets, the function mapping a given instance x to the supremum of the solution space of x (with respect to some fixed relation for the NP set) is complete for $\mathrm{FP}^{\mathrm{NP}}_{\|}$. An example is $\sup(\varphi)$, for a Boolean formula $\varphi = \varphi(x_1, \ldots, x_n)$, the supremum of the satisfying assignments of φ.

We define a set OddSup that is complete for $\mathrm{P}^{\mathrm{NP}[O(\log n)]}$ as follows.

$$\text{OddSup} = \{\, \varphi \in \text{SAT} \mid \text{the number of ones in } \sup(\varphi) \text{ is odd} \,\}.$$

Then, the supremum itself is a proof for OddSup that can be checked in D^{P}. We conclude that OddSup has a $(\mathrm{D}^{\mathrm{P}}, \mathrm{FP}^{\mathrm{NP}}_{\|})$ proof-system.

Theorem 3.12. (1) OddSup is complete for $\mathrm{P}^{\mathrm{NP}[O(\log n)]}$,

(2) $C = \{\, (\varphi, s) \mid \varphi \in \text{OddSup and } \sup(\varphi) = s \,\}$ is D^{P} complete.

Proof. (1). First, we show that SAT can be reduced to OddSup. Let $\varphi = \varphi(x_1, \ldots, x_n)$ be some Boolean formula. We assume that n is even and that the assignment 1^n is not a satisfying

assignment of φ (otherwise we conjunct one or two new negated variables to φ). Now, for a new variable x_{n+1}, we define

$$h(\varphi) \quad = \quad (\bigwedge_{i=1}^{n} x_i \vee x_{n+1}) \ \wedge \ (\varphi \vee \overline{x_{n+1}}).$$

Then $h(\varphi) \in$ SAT and, if φ is in SAT, then $\sup(h(\varphi)) = 1^{n+1}$, which contains an odd number of ones. If φ is not in SAT, then $\sup(h(\varphi)) = 1^n 0$, which contains an even number of ones. This shows that h reduces SAT to OddSup.

Now, let $L \in \mathrm{P}^{\mathrm{NP}[O(\log n)]} = \mathrm{P}_{\|}^{\mathrm{NP}}$ via some polynomial-time machine M asking parallel queries to SAT. We show that L can be reduced to OddSup.

We define two NP sets A and B as follows. For $x \in \Sigma^*$ let $\varphi_1, \ldots, \varphi_l$ be the queries of M on input x to SAT, for some polynomially bounded l, and let $k \geq 0$,

$$\begin{aligned}
(x,k) \in A \quad &\Longleftrightarrow \quad \text{at least } k \text{ of } \varphi_1, \ldots, \varphi_l \text{ are in SAT,} \\
(x,k) \in B \quad &\Longleftrightarrow \quad (x, k+1) \in A \text{ or} \\
&\qquad \text{there are } i_1 < \cdots < i_k \text{ such that } \varphi_{i_1}, \ldots, \varphi_{i_k} \in \text{SAT and } M \text{ accepts when} \\
&\qquad \text{answer yes is given to } \varphi_{i_1}, \ldots, \varphi_{i_k} \text{ and answer no to the other queries.}
\end{aligned}$$

Suppose that exactly k_0 of the queries $\varphi_1, \ldots, \varphi_l$ are in SAT. Then we have

- $(x,1), \ldots, (x,k_0) \in A$ and $(x,k_0+1), \ldots, (x,l) \notin A$,
- $(x,0), \ldots, (x,k_0-1) \in B$ and $(x,k_0+1), \ldots, (x,l) \notin B$, and
- $(x,k_0) \in B \iff x \in L$.

Let f_A and f_B be reductions from A and B to SAT, respectively, and h be the reduction from SAT to OddSup from above. Now, with $\alpha_k = h \circ f_A(x,k)$ and $\beta_k = h \circ f_B(x,k)$, where all the formulas have different sets of variables, we define

$$g(x) \quad = \quad \beta_0 \ \wedge \ \bigwedge_{k=1}^{l} (\alpha_k \wedge \beta_k).$$

We claim that g reduces L to OddSup. Assume first that x is in L. Then, of all the formulas α_i and β_j occuring in $g(x)$, precisely $\alpha_1, \ldots, \alpha_{k_0}$ and $\beta_0, \ldots, \beta_{k_0}$ are in OddSup, i.e., an odd number of formulas, so that $g(x)$ is in OddSup. If x is not in L, then precisely $\alpha_1, \ldots, \alpha_{k_0}$ and $\beta_0, \ldots, \beta_{k_0-1}$ are in OddSup, i.e., an even number of formulas, so that $g(x)$ is not in OddSup.

(2). We define two NP sets L_1 and L_2 such that $C = L_1 - L_2$. For a formula $\varphi = \varphi(x_1, \ldots, x_n)$ and a string $s = s_1 \cdots s_n$ that has an odd number of ones,

$$\begin{aligned}
(\varphi, s) \in L_1 \quad &\Longleftrightarrow \quad \forall i: \ \ s_i = 1 \implies \text{there exists a satisfying assignment of } \varphi \\
&\qquad \text{setting variable } x_i \text{ to one,} \\
(\varphi, s) \in L_2 \quad &\Longleftrightarrow \quad \exists i: \ \ s_i = 0 \text{ and there exists a satisfying assignment of } \varphi \\
&\qquad \text{setting variable } x_i \text{ to one.}
\end{aligned}$$

To show that C is hard for D^{P}, we reduce (SAT,UNSAT) to C. Let us fix two formulas $\varphi_1 = \varphi(x_1, \ldots, x_n)$ and $\varphi_2 = \varphi(y_1, \ldots, y_m)$. We assume again that φ_1 and φ_2 have an even number of variables and that 1^n and 1^m are not satisfying assignments for φ_1 and φ_2, respectively. Let

$$(\psi, s) \quad = \quad (h(\varphi_1) \wedge h(\varphi_2), 1^{n+m+1}0).$$

By the above mentioned properties of h, we have that (φ_1, φ_2) is in (SAT,UNSAT) if and only if (ψ, s) is in C. □

Analogous results to Theorem 3.12 hold with respect to the other supremum functions considered by Chen and Toda. For example they considered the supremum of Hamiltonian Cycles, Longest Paths, or of Minimum Dominating Set and showed those functions to be complete for $\mathrm{FP}_{\|}^{\mathrm{NP}}$.

Example 3. For a deterministic oracle transducer T, a string $z \in \Sigma^*$, and a $t \geq 0$, where T on input z makes at most t steps and queries are made in parallel, define

$$\begin{aligned} u(T, z, 1^t) &= \text{the output of } T^{\mathrm{SAT}} \text{ on input } z, \text{ and} \\ U &= \{\, x \mid \text{the number of ones in } u(x) \text{ is odd} \,\}. \end{aligned}$$

It is easy to see that u is complete for $\mathrm{FP}_{\|}^{\mathrm{NP}}$. Furthermore, U is complete for $\mathrm{P}^{\mathrm{NP}[O(\log n)]}$. We could use u as the proof generating function for U. But note that the set $C_u = \{\, (x, y) \mid x \in U \text{ and } u(x) = y \,\}$ is complete for $\mathrm{P}^{\mathrm{NP}[O(\log n)]}$.

Instead of u, we take the following function v. For T, z, and t as above, let

$$v(T, z, 1^t) \quad = \quad \text{the answers to the queries of } T^{\mathrm{SAT}}(z).$$

It is not hard to see that v is complete for $\mathrm{FP}_{\|}^{\mathrm{NP}}$. Now, v generates proofs for U that can be checked within D^{P}. We conclude that U has a $(\mathrm{D}^{\mathrm{P}}, \mathrm{FP}_{\|}^{\mathrm{NP}})$ proof-system.

Theorem 3.13. (1) U is complete for $\mathrm{P}^{\mathrm{NP}[O(\log n)]}$,

(2) $C_v = \{\, (x, a) \mid x \in U \text{ and } v(x) = a \,\}$ is complete for D^{P}.

Proof. (1). Let $L \in \mathrm{P}^{\mathrm{NP}[O(\log n)]} = \mathrm{P}_{\|}^{\mathrm{NP}}$ via some polynomial-time machine M asking parallel queries to SAT. We show that L can be reduced to U.

Buss and Hay [BH88] (see also Beigel [B91]) have shown that there exists a function $f \in \mathrm{FP}$ such that for any x, $f(x)$ is a sequence of Boolean formulas $\varphi_1, \ldots, \varphi_l$ such that

$$x \in L \iff \text{an odd number of the } \varphi_i\text{'s is in SAT.}$$

Consider the following transducer T with oracle SAT. On input x, T computes $f(x) = (\varphi_1, \ldots, \varphi_l)$ and for each φ_i in SAT, T writes a one on its output tape, for $i = 1, \ldots, l$. Then we have $x \in L \iff (T, x, 1^t) \in U$, for some appropriate choice of t.

(2). To show that C_v is in D^{P}, we define two NP sets L_1 and L_2 such that $C_v = L_1 - L_2$. For $x = (T, z, t)$ according to the definition of U, where T on input z makes at most t steps and the queries made in parallel are $\varphi_1, \ldots, \varphi_l$, for some $l \leq t$, and a string $a = a_1 \cdots a_l$,

$$(x,a) \in L_1 \iff \forall i: \ a_i = 1 \implies \varphi_i \in \text{SAT and } T\text{'s output has an odd number of ones,}$$
$$\text{when answers to the queries } \varphi_1, \ldots, \varphi_l \text{ are given according to } a,$$
$$(x,a) \in L_2 \iff \exists i: \ a_i = 0 \text{ and } \varphi_i \in \text{SAT}.$$

To show that C_v is hard for D^{P}, we reduce (SAT,UNSAT) to C_v. Let us define a transducer M that, on input of two formulas φ_1 and φ_2, asks both formulas to its oracle and then outputs a one. Then we have $(\varphi_1, \varphi_2) \in (\text{SAT}, \text{UNSAT}) \iff ((M, (\varphi_1, \varphi_2), 1^t), 10) \in C_v$, for some appropriate choice of t. □

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}. Since, 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 in fact not complete for $\mathrm{P}^{\mathrm{NP}[O(\log n)]}$.

The function class $\mathrm{FP}^{\mathrm{NP}[O(\log n)]}$ is strictly included in $\mathrm{FP}_{\|}^{\mathrm{NP}}$, unless the Polynomial Hierarchy collapses [S, T91]. Therefore, proofs generated by functions from $\mathrm{FP}^{\mathrm{NP}[O(\log n)]}$ can contain less information than from $\mathrm{FP}_{\|}^{\mathrm{NP}}$, and hence might be harder to check. However, proofs for $\mathrm{P}^{\mathrm{NP}[O(\log n)]}$ complete sets generated by functions from $\mathrm{FP}^{\mathrm{NP}[O(\log n)]}$ can still be checked in D^{P}. This comes from the fact that a $\mathrm{FP}^{\mathrm{NP}[O(\log n)]}$ function can compute enough information such that an NP machine can compute the value of $\mathrm{FP}_{\|}^{\mathrm{NP}}$ function for itself. Since $\mathrm{NP} \subseteq \mathrm{D}^{\mathrm{P}}$, this holds for D^{P} as well. Namely, in Example 1, define $h'_{UMaxSAT}(\varphi)$ to give just *the number of clauses that can be satisfied at most simultaneously*, in Example 2, define $\sup'(\varphi)$ to just give the number of ones in $\sup(\varphi)$, and, in Example 3, define $v'(x)$ to give *the number of yes-answers the transducer gets to its oracle queries.* Then it is not hard to see that $h'_{UMaxSAT}$, $\sup'$, and v' are in $\mathrm{FP}^{\mathrm{NP}[O(\log n)]}$. Furthermore, the corresponding sets C to check the proofs generated by $h'_{UMaxSAT}$, $\sup'$, and v' for UMaxSAT, OddSup, and U, respectively, are D^{P} complete. To see this, one has to modify the proofs of the above theorems appropriately.

4 Proof-Systems for Sets in NP

In this section, we address the question whether NP has a $(\text{coNP}, \mathrm{FP}_{\|}^{\mathrm{NP}})$ proof-system. We observe first that NP cannot have a $(\text{coNP}, \mathrm{FP}^{\mathrm{NP}[O(\log n)]})$ 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 4.1. If NP has a $(\text{coNP}, \mathrm{FP}^{\mathrm{NP}[O(\log n)]})$ proof-system then NP = coNP.

The following theorem shows that if NP has coNP-checkable proofs in $\mathrm{FP}_{\|}^{\mathrm{NP}}$ then so has $\mathrm{P}^{\mathrm{NP}[O(\log n)]}$. Therefore, motivated by the results of Section 3, *we conjecture that NP does not have a* $(\text{coNP}, \mathrm{FP}_{\|}^{\mathrm{NP}})$ *proof-system.*

Theorem 4.2. NP has coNP-checkable proofs in $\mathrm{FP}_{\|}^{\mathrm{NP}}$ if and only if $\mathrm{P}^{\mathrm{NP}[O(\log n)]}$ has coNP-checkable proofs in $\mathrm{FP}_{\|}^{\mathrm{NP}}$.

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 NP set A, 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 proof 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. □

Note that even for the more specific question whether one can compute satisfying assignments with parallel queries to NP, an absolute answer or an answer in the sense of Theorem 1.1 cannot be given using relativizing techniques. Namely, Watanabe and Toda [WT93] showed the existence of an oracle such that relative to that oracle $F_{sat} \cap \mathrm{FP}_{\|}^{\mathrm{NP}} \neq \emptyset$. Moreover, Fortnow [Fo94] constructed an oracle with the same property, and in addition, the (relativized) Polynomial Hierarchy is infinite. On the other hand, we show that there is an oracle such that $F_{sat} \cap \mathrm{FP}_{\|}^{\mathrm{NP}} = \emptyset$, so that there are relativizations in both ways.

We will see that relative to an oracle constructed by Impagliazzo and Tardos [IT89], we have $F_{sat} \cap \mathrm{FP}_{\|}^{\mathrm{NP}} = \emptyset$. They showed a relativized world where E = NE and NE does not have a (P,FE) proof-system, where E, NE, and FE are the linear exponential-time (function) classes. This is in contrast to the polynomial-time setting where P = NP clearly implies that NP has a (P,FP) proof-system. However, we show in Lemma 4.3 below that if $F_{sat} \cap \mathrm{FP}_{\|}^{\mathrm{NP}} \neq \emptyset$ then E = NE implies that NE has a (P,FE) proof-system. Since the lemma relativizes, satisfying assignments cannot be computed within $\mathrm{FP}_{\|}^{\mathrm{NP}}$ relative to the above oracle.

Lemma 4.3. If $F_{sat} \cap \mathrm{FP}_{\|}^{\mathrm{NP}} \neq \emptyset$ and E = NE then NE has a (P,FE) proof-system.

Proof. Let $L \in \mathrm{NE}$ and $A \in \mathrm{P}$ such that for some constant c and all x,

$$x \in L \iff \exists y\ (|y| \leq 2^{c|x|}) :\ (x, y) \in A.$$

We show that for every $x \in L$ a witness y with respect to A can be computed in $\mathrm{FE}_{\|}^{\mathrm{NP}}$, i.e., in exponential time with parallel queries to NP. Note that E = NE implies that $\mathrm{FE}_{\|}^{\mathrm{NP}} = \mathrm{FE}$, so that this suffices to prove the lemma.

Consider the tally versions of L and A, i.e., the sets $T(L) = \{\, 1^{1x} \mid x \in L \,\}$ and $T(A) = \{\, (1^{1x}, y) \mid (x, y) \in A \,\}$, where the exponent $1x$ is treated as a binary number. Clearly, $T(L) \in \mathrm{NP}$ and $T(A) \in \mathrm{P}$.

Let $T(L) \leq_m^P \mathrm{SAT}$ via Cook's reduction such that for all $x \in L$, if 1^{1x} is reduced to φ then one can generate in polynomial time from a satisfying assignment of φ a witness y such that

$(1^{1x}, y) \in T(A)$, and hence, $(x, y) \in A$. Since $F_{sat} \cap \mathrm{FP}_{\|}^{\mathrm{NP}} \neq \emptyset$, it follows that there exists a function $f \in \mathrm{FP}_{\|}^{\mathrm{NP}}$ that, on input 1^{1x}, generates such a witness y with respect to $T(A)$ for every $x \in T(L)$ Now, let $g(x) = f(1^{1x})$. Then $g \in \mathrm{FE}_{\|}^{\mathrm{NP}}$ and furthermore, g computes a witness with respect to A for every $x \in L$. □

Corollary 4.4. There exists an oracle relative to which $F_{sat} \cap \mathrm{FP}_{\|}^{\mathrm{NP}} = \emptyset$.

Finally, we want to connect the question of whether NP or $\mathrm{P}^{\mathrm{NP}[O(\log n)]}$ have a $(\mathrm{coNP}, \mathrm{FP}_{\|}^{\mathrm{NP}})$ proof-system with some other open problems in computational complexity. We start by considering some classes that indeed have such proof-systems.

Definition. UP and FewP are the classes of sets that are accepted by some nondeterministic polynomial-time bounded Turing machine that has at most one, resp. polynomially many accepting paths for each input. FUP is the class of functions computed by a UP type of machine, i.e., there is at most one path that makes an output. Few is defined as FewP, but here a P predicate getting the number of accepting paths of the machine on some input x finally decides membership of x.

We have $\mathrm{UP} \subseteq \mathrm{FewP} \subseteq \mathrm{NP}$, $\mathrm{Few} \subseteq \mathrm{P}^{\mathrm{NP}[O(\log n)]}$, and $\mathrm{FUP} \subseteq \mathrm{NPSV}$.

Proposition 4.5. (1) UP has P-checkable proofs in FUP,

(2) FewP has P-checkable proofs in $\mathrm{FP}_{\|}^{\mathrm{NP}}$,

(3) Few and USAT have coNP-checkable proofs in $\mathrm{FP}_{\|}^{\mathrm{NP}}$.

As already mentioned in Section 3, USAT is complete for D^{P} if it is NP hard. It follows from Proposition 4.5 (3) that SAT $\leq_m^P$ USAT implies that NP has coNP-checkable proofs in $\mathrm{FP}_{\|}^{\mathrm{NP}}$. We show below that this assumption even implies that NP as coNP-checkable proofs in NPSV. Thus, showing that NP does not have such a proof-system would also show that USAT is not complete for D^{P}, thereby solving an old open problem.

Suppose even stronger 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 this implies that NP = UP which, by Proposition 4.5, is equivalent with NP having a P-checkable proofs generated in FUP.

Theorem 4.6. (1) SAT $\leq_m^P$ USAT $\Longrightarrow$ NP has a (coNP,NPSV) proof-system,

(2) $F_{sat} \leq_{1\text{-}T}^{FP} f_{USat} \Longrightarrow F_{sat} \cap \mathrm{FUP} \neq \emptyset \Longrightarrow \mathrm{NP} = \mathrm{UP}$.

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 following nondeterministic polynomial-time transducer M. We describe M on input φ by using some PASCAL-like language.

$M(\varphi)$

1 **if** $t_2(\varphi, \perp)$ is a satisfying assignment of φ **then output** $t_2(\varphi, \perp)$
2 **else**
3 **guess** an assignment **a** for φ
4 **if a** does not satisfy φ **then reject**
5 **else**
6 **guess** an assignment **b** for $t_1(\varphi)$
7 **if b** does not satisfy $t_1(\varphi)$ **then reject**
8 **else**
9 **if** $t_2(\varphi, \mathbf{b}) = \mathbf{a}$ **then output a**
10 **else reject.**

We claim that M outputs exactly one satisfying assignment on some path, if $\varphi \in$ SAT, and makes no output, if $\varphi \notin$ SAT. To see this, let us first assume $\varphi \notin$ SAT. Since M will not find a satisfying assignment for φ, M will reject on all paths in line 4. Now, assume that $\varphi \in$ 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 **b** of $t_1(\varphi)$. Now, by assumption, $t_2(\varphi, \mathbf{b}) = \mathbf{a_{i_0}}$ for some $i_0 \in \{1, \ldots, k\}$. Finally, M will output $\mathbf{a_{i_0}}$ in line 9 on the unique path where $\mathbf{a_{i_0}}$ 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. □

Since FUP $\subseteq$ NPSV, we conclude from Theorem 1.1 that the assumptions $F_{sat} \leq^{FP}_{1\text{-}T} f_{USat}$ and $F_{sat} \cap$ 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 4.6 are in fact equivalences.

Corollary 4.7. If $F_{sat} \leq^{FP}_{1\text{-}T} f_{USat}$, then the Polynomial Hierarchy collapses to the second level.

References

[AW90] E. Allender and O. Watanabe. Kolmogorov Complexity and degrees of Tally sets. *Information and Computation 86(2)*, pages 160–178, 1990

[BDG88] J. Balcázar, J. Díaz, and J. Gabarró. *Structural Complexity I.* EATCS Monographs on Theoretical Computer Science, Springer-Verlag, 1988.

[BDG91] J. Balcázar, J. Díaz, and J. Gabarró. *Structural Complexity II.* EATCS Monographs on Theoretical Computer Science, Springer-Verlag, 1991.

[B91] R. Beigel. Bounded queries to SAT and the Boolean hierarchy. *Theoretical Computer Science 84*, pages 199-223, 1991.

[BG82] A. Blass and Y. Gurevich. On the unique satisfiability problem. *Information and Control 55*, pages 80–88, 1982.

[BKT94] H. Buhrman, J. Kadin, and T. Thierauf. On functions computable with nonadaptive queries to NP. In *Proc. 9th Structure in Complexity Theory Conference*, pages 43–52, 1994.

[BH88] S. Buss and L. Hay. On truth table reducibilities to SAT and the difference hierrachy over NP. In *Proc. 3rd Structure in Complexity Theory Conference*, pages 224–233, 1988.

[CKR91] R. Chang, J. Kadin, and P. Rohatgi. Connections between the complexity of Unique Satisfiability and the threshhold behavior of randomized reductions. In *Proc. 6th Structure in Complexity Theory Conference*, pages 255–269, 1991.

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

[CT91] Z. Chen and S. Toda. On the Complexity of Computing Optimal Solutions. In *International Journal of Foundations of Computer Science 2*, pages 207-220, 1991.

[CT93] Z. Chen and S. Toda. An Exact Characterization of $\mathrm{FP}_{\|}^{\mathrm{NP}}$. Manuscript, 1993.

[FHOS93] S. Fenner, S. Homer, M. Ogiwara, and A. Selman. On Using Oracles That Compute values. In *10-th Annual Symposium on Theoretical Aspects of Computer Science*, pages 398–407, 1993.

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

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

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

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

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

[JT93] B. Jenner and J. Torán. Computing Functions With Parallel Queries to NP. In *Proc. 8th Structure in Complexity Theory Conference*, pages 280–291, 1993.

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

[Kr86] M. Krentel. The Complexity of Optimization Problems. In *Proc. 18th ACM Symposium on Theory of Computing*, pages 69–76, 1986.

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

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

[PZ83] C. Papadimitriou and D. Zachos. Two remarks on the power of counting. In *6th GI Conference on TCS*, Springer Verlag LNCS 145, pages 269–276, 1983

[S] A. Selman. A taxonomy of complexity classes of functions. *Journal of Computer and System Science*, to appear.

[T91] S. Toda. On polynomial-time truth-table reducibilities of intractable sets to P-selective sets. *Mathematical Systems Theory 24*, pages 69–82, 1991.

[St77] L. Stockmeyer, The Polynomial-Time Hierarchy. *Theoretical Computer Science* 3, pages 1–22, 1977.

[Va76] L. Valiant. The Relative Complexity of Checking and Evaluating. *Information Processing Letters 5*, pages 20–23, 1976.

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

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

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

[WT93] O. Watanabe and S. Toda. Structural Analysis on the Complexity of Inverse Functions. In *Mathematical Systems Theory 26*, 203–214, 1993.