Electronic Colloquium on Computational Complexity, Report No. 31 (2014)

On the Query Complexity of Selecting Few Minimal Sets

Joao Marques-Silva[1,2] and Mikoláš Janota[2]
[1,2] *CASL, University College Dublin, Ireland*
[2] *IST/INESC-ID, Technical University of Lisbon, Portugal*

Abstract

Propositional Satisfiability (SAT) solvers are routinely used for solving many function problems. A natural question that has seldom been addressed is: what is the *best* worst-case number of calls to a SAT solver for solving some target function problem? This paper develops tighter upper bounds on the query complexity of solving several function problems defined on propositional formulas. These include computing the backbone of a formula and computing the set of independent variables of a formula. For the general case, the paper develops tighter upper bounds on the query complexity of computing a minimal set when the number of minimal sets is constant. This applies for example to the computation of a minimal unsatisfiable subset (MUS) for CNF formulas, but also to the computation of prime implicants and implicates.

1 Introduction

The practical success of Boolean Satisfiability (SAT) solvers is demonstrated by an ever increasing number of applications. While some of these applications are naturally formulated as decision problems; others are not. In many settings SAT solvers are used for solving function problems. This is the case for example with computing a minimal unsatisfiable subset (MUS) of a CNF formula, a minimal correction subset (MCS) of a CNF formula, the backbone of a propositional formula, a prime implicant or implicate of a propositional formula, the largest autark assignment of a CNF formula, among many other function problems. Some of these function problems find important practical applications. For example MUSes are routinely used in abstraction refinement loops in software model checking (e.g. [11]), prime implicates are also used in model checking [2, 3], and backbones are used for example in configuration and in post-silicon fault localization [23].

Despite the vast number of settings in which SAT solvers are used for solving function problems, the worst-case number of times a SAT solver is called for solving these problems is in general not known with sufficient accuracy. For example, it is not known what is the *best* worst-case number of calls to a SAT

ISSN 1433-8092

solver for computing the backbone of a propositional formula, or for computing the maximum autarky of a CNF formula, or for computing a minimal unsatisfiable subset, or for computing a maximally satisfiable subset, or for computing a prime implicant or implicate, among many other function problems. Moreover, the exact query complexity of computing an MUS has been an open research topic for around two decades [5].

This paper extends recent work on unifying the approaches used for solving a large number of function problems related with propositional formulas [16, 15], namely function problems that can be modeled as computing a minimal set subject to a monotone predicate. More concretely, this paper investigates the worst-case number of times SAT solvers are called for solving function problems, and refines known upper bounds for a number of problems.

Most algorithms for solving function problems can be modeled with polynomial time algorithms that call a SAT solver a polynomial number of times. A measure of the hardness of these function problems is the number of times the SAT solver is called given the problem instance size, i.e. the query complexity of the function problem. Following standard approaches in computational complexity, the SAT solver can be viewed as an *oracle*: given an instance the SAT solver answers positively (for satisfiable instances) or negatively (for unsatisfiable instances). However, the operation of SAT solvers differs from the traditional NP oracle (e.g. [6, 17]). In the case of positive answers, the SAT solver also returns a satisfying truth assignment (i.e. a *witness*). The ability of the SAT solver to return a witness on positive answers can have a profound impact on query complexity characterizations of function problems (related results have actually been investigated in the past, e.g. [7, 8]). This paper demonstrates that in some function problems, the availability of witnesses enables reducing the worst case number of oracle queries from polynomial to logarithmic. As a result, this paper proposes to use witness oracles [4, 12] to model the operation of a SAT solver, and to develop query complexity characterizations of function problems in terms of witness oracles. These query complexity characterizations can then serve to compare available algorithms against the best known theoretical results.

The contributions of this paper can be summarized as follows. For function problems that can be represented as computing a minimal set subject to a monotone predicate in certain general forms [16, 15], and have a constant number of minimal sets, the paper shows that a minimal set can be computed with a logarithmic number of calls to a witness oracle. This result has a number of consequences which the paper also analyzes. For example, the backbone of a propositional formula or the independent variables of a propositional formula can be computed with a logarithmic number of calls to a witness oracle. In addition, a few additional results are included in the paper, related to special cases of other function problems when the number of minimal sets is constant. Observe that, although the paper addresses function problems solved with polynomial number of calls to a SAT solver, other *oracles* could be considered, including SMT, CSP, and QBF solvers.

The paper is organized as follows. Section 2 introduces the notation used in

the paper. Section 3 develops query complexity results for function problems that have exactly one minimal set. Section 4 extends the results of the previous section for function problems that have exactly a constant number of minimal sets. The paper concludes in Section 5.

2 Preliminaries

This section introduces the notation and definitions used throughout the paper.

2.1 Propositional Logic

Standard propositional logic definitions are used throughout the paper (e.g. [10, 1]), including propositional formulas, truth assignments, etc. Some definitions are briefly reviewed in this section.

Sets are represented in calligraphic font, e.g. $\mathcal{R}, \mathcal{W}, \ldots$ Propositional formulas are also represented in calligraphic font, e.g. $\mathcal{F}, \mathcal{H}, \mathcal{T}, \ldots$ Propositional variables are represented with letters from the end of the alphabet, e.g. x, y, z, and indices can be used, e.g. $x_1, y_1, \ldots$ A literal is a variable x_i or its negation $\neg x_i$.

The variables of a propositional formula $\mathcal{F}$ are represented by $\mathsf{var}(\mathcal{F})$. For simplicity, the set of variables of a formula will be denoted by $X \triangleq \mathsf{var}(\mathcal{F})$. A clause c is a non-tautologous set of literals, interpreted as a disjunction of literals. A CNF formula $\mathcal{F}$ is set clauses, interpreted as a conjunction of clauses.

When the set of variables of a formula $\mathcal{F}$ is relevant, the notation $\mathcal{F}[X]$ is used, meaning that F is defined in terms of variables from X. Substitutions of variables will be used. The notation $\mathcal{F}[x_i/y_i]$ represents formula $\mathcal{F}$ with variable x_i replaced with y_i. This definition can be extended to more than one variable. For the general case, if $X = \{x_1, \ldots, x_n\}$ and $Y = \{y_1, \ldots, y_n\}$, then the notation $\mathcal{F}[X/Y]$ represents formula $\mathcal{F}$ with x_1 replaced by y_1, x_2 replaced by y_2, ..., and x_n replaced by y_n, simultaneously. Alternatively, one could write $\mathcal{F}[x_1/y_1, x_2/x_2, \ldots, x_n/y_n]$.

2.2 Computational Complexity

Standard computational complexity definitions are used throughout the paper [6, 17]. The notation is adapted from [17] and more recent papers (e.g. [19]). Besides the well-known complexity classes P, NP, coNP, D^{p}, as well as $\mathrm{P}^{\mathrm{NP}} = \Delta_2^{\mathrm{p}}$, $\mathrm{NP}^{\mathrm{NP}} = \Sigma_2^{\mathrm{p}}$, etc., the following classes of function problems are used in the paper (see [17] for a definition of function problem):

- FP: class of function problems solvable in deterministic polynomial time.
- $\mathrm{FP}^{\mathrm{NP}}$: class of function problems solvable in deterministic polynomial time by executing a polynomial number of calls to an NP oracle.
- $\mathrm{FP}^{\mathrm{NP}}[\mathsf{log}]$: class of function problems solvable in deterministic polynomial time by executing a logarithmic number of calls to an NP oracle.

Observe that, although $\mathrm{P}^{\mathrm{NP}}[\mathsf{log}] = \mathrm{P}_{||}^{\mathrm{NP}}$ (e.g. [17]), it is believed that $\mathrm{FP}^{\mathrm{NP}}[\mathsf{log}] \neq \mathrm{FP}_{||}^{\mathrm{NP}}$ (e.g. [20]).

As indicated above, a SAT solver computes witnesses for the positive outcomes. Thus, the standard NP oracle model is inadequate given that retrieving a witness requires more than a logarithmic number of calls to an NP oracle, unless P = NP [7, Theorem 5.4]. Nevertheless, researchers have looked into a similar issue in the past, and proposed the use of a *witness oracle*, i.e. an NP oracle that returns a witness for the positive outcomes. The definition of a witness oracle is taken from [4, page 4] and [12, Definition 6.3.1]. Observe that the difference between the default NP oracle is of interest only when a better than polynomial (e.g. logarithmic) number of calls can be used for solving a given function problem. Hence, this paper considers the following additional complexity class:

- $\text{FP}^{\text{NP}}[\mathsf{wit},\mathsf{log}]$: Class of function problems that can be solved with a logarithmic number of (adaptive) calls to a witness oracle.

Witness oracles can also be defined for other levels of the polynomial hierarchy [4, 12].

2.3 Minimal Sets over Monotone Predicates

Monotone predicates have recently been proposed as a unifying approach for computing minimal sets [2, 3, 16, 15]. A predicate $P : 2^{\mathcal{R}} \rightarrow \{0, 1\}$, defined on a reference set $\mathcal{R}$, with $|\mathcal{R}| = m$, is said to be *monotone* if whenever $P(\mathcal{R}_0)$ holds, with $\mathcal{R}_0 \subseteq \mathcal{R}$, then $P(\mathcal{R}_1)$ also holds, with $\mathcal{R}_0 \subseteq \mathcal{R}_1 \subseteq \mathcal{R}$. Observe that $P(\mathcal{R})$ can be assumed, but this is not required. Also, $P(\mathcal{R})$ can be tested with a single predicate test. Moreover, observe that, if there exists a set $\mathcal{R}_0 \subseteq \mathcal{R}$ such that $P(\mathcal{R}_0)$ holds, and P is monotone, then $P(\mathcal{R})$ also holds.

Definition 1 *Let P be a monotone predicate, and let $\mathcal{M} \subseteq \mathcal{R}$ such that $P(\mathcal{M})$ holds. $\mathcal{M}$ is minimal iff $\forall_{\mathcal{M}' \subsetneq \mathcal{M}}, \neg P(\mathcal{M}')$.*

Definition 2 (MSMP Problem) *Given a monotone predicate P, the* Minimal Set over a Monotone Predicate *(MSMP) problem consists in finding a minimal subset $\mathcal{M}$ of $\mathcal{R}$ such that $P(\mathcal{M})$ holds.*

The MSMP problem was shown to model a number of function problems related with computing minimal sets on propositional formulas in [16]. This work was extended in [15] to show that a much larger set of function problems can be reduced to the MSMP problem. Nevertheless, and as shown in [15] for the considered problems, the monotone predicates end up being of one of three forms. Let $\mathcal{R}$ be a reference set, and let element $u_i \in \mathcal{R}$ represent either a literal or a clause. Moreover, $\sigma(u_i)$ represents a Boolean formula built from u_i, where new variables may be used, but such that u_i is the only element from $\mathcal{R}$ used in $\sigma(u_i)$. For example, $\sigma(u_i)$ can represent a clause, a literal, or the negation of a literal or of a clause, etc. Let $\mathcal{G}$ denote a propositional formula, and let $\mathcal{W} \subseteq \mathcal{R}$. Then, from [15] the following predicate forms are defined.

Table 1: Examples of monotone predicates

Problem	$\mathcal{R}$	Form	u_i	$\sigma(u_i)$	$\mathcal{G}$	Predicate, P, with $\mathcal{W} \subseteq \mathcal{R}$
FMUS	$\mathcal{F}$	$\mathscr{P}$	cl. c	c	$\emptyset$	$\neg\mathsf{SAT}(\wedge_{c\in\mathcal{W}}(c))$
FMCS	$\mathcal{F}$	$\mathscr{L}$	cl. c	c	$\emptyset$	$\mathsf{SAT}(\wedge_{c\in\mathcal{R}\setminus\mathcal{W}}(c))$
FBB	X	$\mathscr{B}$	var. x	$(x \wedge \neg y)$	$\mathcal{F}^{\text{BB}}$	$\neg\mathsf{SAT}(\mathcal{F}^{\text{BB}} \wedge (\vee_{x\in\mathcal{R}\setminus\mathcal{W}}(x \wedge \neg y)))$
FVInd	X	$\mathscr{P}$	var. x	$(x \leftrightarrow y)$	$\mathcal{F}^{\text{VInd}}$	$\neg\mathsf{SAT}(\mathcal{F}^{\text{VInd}} \wedge \wedge_{x\in\mathcal{W}}(x \leftrightarrow y))$

Definition 3 (Predicates of Form $\mathscr{L}$) *A predicate P is of* form $\mathscr{L}$ *iff its general form is given by,*

$$P(\mathcal{W}) \triangleq \mathsf{SAT}(\mathcal{G} \wedge \wedge_{u_i\in\mathcal{R}\setminus\mathcal{W}}(\sigma(u_i))) \tag{1}$$

Definition 4 (Predicates of Form $\mathscr{B}$) *A predicate P is of* form $\mathscr{B}$ *iff its general form is given by,*

$$P(\mathcal{W}) \triangleq \neg\mathsf{SAT}(\mathcal{G} \wedge (\vee_{u_i\in\mathcal{R}\setminus\mathcal{W}}(\sigma(u_i)))) \tag{2}$$

Definition 5 (Predicates of Form $\mathscr{P}$) *A predicate P is of* form $\mathscr{P}$ *iff its general form is given by,*

$$P(\mathcal{W}) \triangleq \neg\mathsf{SAT}(\mathcal{G} \wedge \wedge_{u_i\in\mathcal{W}}(\sigma(u_i))) \tag{3}$$

Example 1 *The following function problems are studied in this paper: (i) extracting an MUS (FMUS); extracting an MCS (FMCS); (iii) computing the backbone of a formula (FBB); and (iv) identifying the set of independent variables of a formula (FVInd). The monotone predicates for FMUS and FMCS are studied in detail in [16, 15]. For FBB, subformula $\mathcal{G}$ is given by, $\mathcal{F}^{\text{BB}} \triangleq \mathcal{F}[X/X] \wedge \mathcal{F}[X/Y]$, where Y is a fresh set of variables. For FBB, observe that the elements in the complement of the minimal set $\mathcal{R} \setminus \mathcal{M}$ are the variables for which $\mathcal{F}$ cannot be satisfied with $x = 0$ and $x = 1$. For FVInd, subformula $\mathcal{G}$ is given by, $\mathcal{F}^{\text{VInd}} \triangleq (\mathcal{F}[X/Y] \wedge \neg\mathcal{F}[X/X] \vee \neg\mathcal{F}[X/Y] \wedge \mathcal{F}[X/X])$, where Y is again a fresh set of variables. For FVInd, observe that the elements in the minimal set are the variables that must take the same value in the two copies of $\mathcal{F}$ for those copies to remain equivalent.*

2.4 Related Work

The query complexity of selecting minimal/maximal sets has been studied in the past [13, 5]. The hereditary property studied in [5] resembles monotonicity properties studied in recent work [2, 3, 16, 15]. Nevertheless, the approaches have important differences, the most significant of which is that hereditarity is defined in terms of possible solutions; that is not the case with monotonicity.

Of the function problems studied in this paper, the best studied is the selection of an MUS (e.g. [18, 5]). The actual query complexity of computing an

MUS is not known. The best bounds were obtained in [5], with a lower bound in $\mathrm{FP}_{||}^{\mathrm{NP}}$ and a (trivial) upper bound in $\mathrm{FP}^{\mathrm{NP}}$. Observe that this makes it very unlikely that selecting an MUS could be in $\mathrm{FP}^{\mathrm{NP}}[\mathsf{log}]$, otherwise FewP = P, NP = R and coNP = US [20]. For computing the backbone of a formula, to the best of our knowledge, the only complexity analysis is [9], but restricted to the decision problem. For computing the set of independent variables of a formula, the best upper bound is $\mathrm{FP}^{\mathrm{NP}}$ [14].

3 Complexity of Selecting 1 Minimal Set

This section investigates the query complexity of selecting the minimal set for instances of the MSMP problem when it is known that there exists exactly one minimal set.

3.1 Preliminary Results

Definition 6 (Partial Maximum Satisfiability (MaxSAT)) *Let $\mathcal{F}$ denote a set of soft clauses, i.e. some may be falsified. Let $\mathcal{H}$ denote a set of hard clauses, i.e. all must be satisfied. Partial MaxSAT problem is the function problem of computing the largest set $\mathcal{S} \subseteq \mathcal{F}$ such that $\mathcal{H} \cup \mathcal{S}$ is satisfiable.*

If $\mathcal{H} = \emptyset$, then the problem is referred to as (plain) MaxSAT. Observe that this definition differs somewhat from the standard definition (e.g. [13]) in that the actual set of satisfied clauses is to be computed, not just the number. However, in many practical applications of MaxSAT, this is exactly what is to be computed. However, with an NP oracle, the identification of the actual set of satisfied clauses requires more than a logarithmic number of calls to an NP oracle; otherwise P = NP [7, Theorem 5.4].

Proposition 1 *Partial MaxSAT is in* $\mathrm{FP}^{\mathrm{NP}}[\mathsf{wit},\mathsf{log}]$.

Proof. It is well-known that (unweighted) (partial) MaxSAT is in $\mathrm{FP}^{\mathrm{NP}}[\mathsf{log}]$ when the function problem is to compute the *number* of simultaneously satisfied clauses [13]. Thus, we can use a straightforward binary search algorithm (e.g. [13]), but using a witness-oracle instead of an NP oracle, which serves to identify the satisfied clauses. Thus the number of witness oracle calls is $\mathcal{O}(\log m)$, where m is the number of clauses. Hence, MaxSAT is in $\mathrm{FP}^{\mathrm{NP}}[\mathsf{wit},\mathsf{log}]$. □

Definition 7 (MaxSet) *Let $\{\mathcal{F}_1(X), \ldots, \mathcal{F}_n(X)\}$ be a set of Boolean functions defined on a set of variables X. The function problem of computing the set of Boolean functions $\mathcal{F}_i$ that are satisfiable is the* MaxSet *problem for $\{\mathcal{F}_1, \ldots, \mathcal{F}_n\}$.*

Proposition 2 *MaxSet is in* $\mathrm{FP}_{||}^{\mathrm{NP}}$.

Proof. The satisfiability of each function $\mathcal{F}_i$ can be tested independently of the other functions. Thus, n non-adaptive calls can be used for solving the MaxSet problem and so MaxSet is in $\mathrm{FP}^{\mathrm{NP}}_{||}$. □

Proposition 3 *MaxSet is in* $\mathrm{FP}^{\mathrm{NP}}[\mathsf{wit},\mathsf{log}]$.

Proof. We reduce MaxSet to partial MaxSAT which, by 1, is in $\mathrm{FP}^{\mathrm{NP}}[\mathsf{wit},\mathsf{log}]$. Consider $n+1$ sets of distinct variables $X_1,\ldots,X_n$, and Z, with $|X_i| = |X|$, $1 \leq i \leq n$, and $|Z| = n$. Define $z_i \leftrightarrow \mathcal{F}_i[X/X_i]$ and let,

$$\mathcal{T} \triangleq \bigwedge_{i=1}^{n} (z_i \leftrightarrow \mathcal{F}_i[X/X_i]) \tag{4}$$

It is plain to conclude that the largest number of z_i variables that can be set to 1 such that $\mathcal{T}$ is satisfied identifies the largest set of satisfiable Boolean functions. Formula $\mathcal{T}$ can be encoded into an equisatisfiable CNF formula in polynomial space by introducing additional variables (e.g. [22]). The resulting clauses are marked as hard. Finally, define the set of soft clauses to be (z_i), $1 \leq i \leq n$. Therefore, the largest set of soft clauses that can be simultaneously satisfied such that $\mathcal{T}$ is satisfied represents all of the Boolean functions that are satisfiable. Thus, MaxSet is in $\mathrm{FP}^{\mathrm{NP}}[\mathsf{wit},\mathsf{log}]$. □

Proposition 4 *Let* Γ *be a function problem represented by monotone predicate* P *of form* $\mathscr{B}$*, and let* $u_j \in \mathcal{R}$*. The formula* $\mathcal{G} \land (\sigma(u_j))$ *is satisfiable iff* $u_j \in \mathcal{M}$ *for every minimal set* $\mathcal{M} \subseteq \mathcal{R}$*. Hence, for* Γ *there is a unique minimal set.*

Proof.

- $\Leftarrow$ By definition of form $\mathscr{B}$, for minimal set $\mathcal{M} \subseteq \mathcal{R}$, $\mathcal{G} \land (\vee_{u_i \in \mathcal{R}\setminus\mathcal{M}}\sigma(u_i))$ is unsatisfiable. By hypothesis, $\mathcal{M}$ is minimal, and so for any $u_j \in \mathcal{M}$, $\mathcal{G} \land (\vee_{u_i \in \mathcal{R}\setminus(\mathcal{M}\setminus\{u_j\})}\sigma(u_i)) \equiv \big(\mathcal{G} \land (\vee_{u_i \in \mathcal{R}\setminus\mathcal{M}}\sigma(u_i)) \lor \mathcal{G} \land (\sigma(u_j))\big)$ is satisfiable. Thus, the formula $\mathcal{G} \land (\sigma(u_j))$ is satisfiable.
- $\Rightarrow$ A minimal set $\mathcal{M} \subseteq \mathcal{R}$ given predicate P of form $\mathscr{B}$ is such that $\mathcal{G} \land (\vee_{u_i \in \mathcal{R}\setminus\mathcal{M}}\sigma(u_i))$ is unsatisfiable and for any $\mathcal{M}' \subsetneq \mathcal{M}$, $\mathcal{G} \land (\vee_{u_i \in \mathcal{R}\setminus\mathcal{M}'}\sigma(u_i))$ is satisfiable. Let $u_j \in \mathcal{R}\setminus\mathcal{M}$, where $\mathcal{M}$ is a minimal set for which P holds. Then, $\mathcal{G} \land (\sigma(u_j))$ is unsatisfiable. Therefore, if $\mathcal{G} \land (\sigma(u_j))$ is satisfiable, then $u_j \in \mathcal{M}$ where $\mathcal{M} \subseteq \mathcal{R}$ is a minimal set given P. □

Therefore, any element $u_j \in \mathcal{R}$ not in a minimal set is such that $\mathcal{G} \land (\sigma(u_j))$ is unsatisfiable. Add any such element to a set $\mathcal{U} = \mathcal{R} \setminus \mathcal{S}$. Then, by construction, $\mathcal{U}$ is maximal, unique, and $\mathcal{G} \land (\vee_{u_i \in \mathcal{R}\setminus\mathcal{U}}\sigma(u_i))$ is unsatisfiable. Thus, $\mathcal{S} = \mathcal{R}\setminus\mathcal{U}$ is a minimal set and it is unique. □

Proposition 5 *Let* Γ *be a function problem represented by monotone predicate* P *of form* $\mathscr{B}$*. Then* Γ *is in* $\mathrm{FP}^{\mathrm{NP}}_{||}$.

Proof. Given 4, for any element $u_j \in \mathcal{M}$, for minimal set $\mathcal{M} \subseteq \mathcal{R}$, formula $\mathcal{G} \land (\sigma(u_j))$ is satisfiable. Otherwise, it is unsatisfiable. Thus, consider $|R|$ non-adaptive oracle calls, where for each u_j check whether $\mathcal{G} \land (\sigma(u_j))$ is satisfiable. If it is, then u_j is in the minimal set $\mathcal{M}$. Otherwise u_j is not in the minimal set. □

Finally, even for predicates of form $\mathscr{P}$ there are examples of function problems which have a unique minimal set.

Proposition 6 *The function problem of computing the set of independent variables of $\mathcal{F}$ (FVInd) [15] has a unique minimal set.*

Proof. By definition, Boolean function $\mathcal{F}$ is independent of x_i iff $\mathcal{F} \equiv \mathcal{F}[x_i/0] \equiv \mathcal{F}[x_i/1]$. Thus, we can check each variable separately for independence. Any variable x_i of which $\mathcal{F}$ is independent is added to a set $\mathcal{I}$. No other variable can be added to $\mathcal{I}$ and $\mathcal{I}$ is unique. □

3.2 Main Results

Theorem 1 *Let Γ be a function problem represented by a monotone predicate P of form $\mathscr{L}$. Then Γ is in $\mathrm{FP}^{\mathrm{NP}}[\mathsf{wit},\mathsf{log}]$.*

Proof. We reduce the problem of computing the smallest minimal set to MaxSAT. Let $\mathcal{M} \subseteq \mathcal{R}$ denote any minimal set. Then, by definition of predicate of form $\mathscr{L}$, there exists an assignment that satisfies $\mathcal{G} \land \land_{u_i \in \mathcal{R} \setminus \mathcal{M}} (\sigma(u_i))$. Let $z_i \leftrightarrow \sigma(u_i)$, where z_i is a new propositional variable. Any minimal set $\mathcal{M}$ corresponds to a maximal set $\mathcal{R} \setminus \mathcal{M}$ of variables z_i assigned value 1. We can compute the largest minimal set as follows. Define the formula,

$$\mathcal{G} \land \land_{u_i \in \mathcal{R}} (z_i \leftrightarrow \sigma(u_i)) \tag{5}$$

which is encoded to a set of hard clauses. Moreover, let the soft clauses be (z_i). This is a MaxSAT formulation, and by 1 the largest set of satisfied z_i variables can be found with a logarithmic number of calls to a witness oracle. The non-satisfied z_i variables denote a smallest minimal set. Thus, any function problem Γ represented with a monotone predicate P of form $\mathscr{L}$ is in $\mathrm{FP}^{\mathrm{NP}}[\mathsf{wit},\mathsf{log}]$. □

Theorem 2 *Let Γ be a function problem represented by a monotone predicate P of form $\mathscr{B}$. Then, $\Gamma \in \mathrm{FP}^{\mathrm{NP}}[\mathsf{wit},\mathsf{log}]$.*

Proof. By 4, any predicate of form $\mathscr{B}$ has exactly one minimal set. Thus, the proof considers that there exists a unique minimal set. We reduce the problem of computing the minimal set to MaxSAT (using the construction in the proof of 3). Let $\mathcal{M} \subseteq \mathcal{R}$ denote the minimal set. By 4, $u_j \in \mathcal{M}$ iff $\mathcal{G} \land (\sigma(u_i))$ is unsatisfiable. Consider the following new sets of variables $X_1, \ldots, X_m$ and Z, and define the formula,

$$\mathcal{T} \triangleq \bigwedge_{u_i \in \mathcal{R}} z_i \leftrightarrow (\mathcal{G} \land (\sigma(u_i)))[X/X_i] \tag{6}$$

with $z_i \in Z$. Hence, the variables z_i that can be assigned value 1, such that $\mathcal{T}$ is satisfied, denote the elements of $\mathcal{R}$ that are *not* in the minimal set. Let the the hard clauses be given by $\mathcal{T}$ and the soft clauses be (z_i). The MaxSAT solution gives the elements that are not in the minimal set. The complement represents the elements that are in the minimal set. Thus, $\Gamma \in \mathrm{FP}^{\mathrm{NP}}[\mathsf{wit},\mathsf{log}]$. □

Theorem 3 *Let Γ be a function problem represented by a monotone predicate P of form $\mathscr{P}$, such that Γ has exactly one minimal set. Then, $\Gamma \in \mathrm{FP}^{\mathrm{NP}}[\mathsf{wit},\mathsf{log}]$.*

Proof. We reduce the problem of computing a minimal set for Γ to the MaxSet problem. Let $\mathcal{M} \subseteq \mathcal{R}$ be the unique minimal set of Γ. Consider the propositional formula representing the argument of predicate P for some $\mathcal{W} \subseteq \mathcal{R}$:

$$\mathcal{G} \land \land_{u_j \in \mathcal{W}} \sigma(u_i) \tag{7}$$

Clearly, if $\mathcal{M} \subseteq \mathcal{W}$, then (7) is unsatisfiable; otherwise it is satisfiable. Next, consider the following $m = |\mathcal{R}|$ sets for $\mathcal{W}$: $\mathcal{R} \setminus \{u_i\}$. Clearly, if $u_i \in \mathcal{M}$, then (7) is satisfiable with $\mathcal{W} = \mathcal{R} \setminus \{u_i\}$; otherwise it is unsatisfiable. Thus, the set of elements $u_i \in \mathcal{R}$ that satisfy (7) when $\mathcal{W} = \mathcal{R} \setminus \{u_i\}$ represents the minimal set of Γ. Next, consider a fresh set of variables for each of the considered sets, $\{X_1, \ldots, X_m\}$:

$$\left(\mathcal{G} \land \land_{u_j \in \mathcal{R} \setminus \{u_i\}} \sigma(u_i)\right)[X/X_i] \tag{8}$$

Thus, we have created $m = |\mathcal{R}|$ propositional formulas, one for each i, $1 \leq i \leq m$, given by (8).
This concludes the reduction to MaxSet, and thus $\Gamma \in \mathrm{FP}^{\mathrm{NP}}[\mathsf{wit},\mathsf{log}]$. □

Observe that, for the concrete case of computing prime implicates, Theorem 3 refines the claims in [2, Theorem 3] and in [3, Theorem 4.2] regarding the selection of a minimal set of Γ when Γ has a unique minimal set. The claim in [2, 3] states that the worst-case number oracle calls is necessarily linear in the size of the problem representation. If the oracle calls use a witness oracle then, as Theorem 3 shows, a better worst-case is achieved.

3.3 Consequences

This section highlights some of the number of consequences from the results in the previous section, namely Theorem 1, Theorem 2, and Theorem 3. An immediate consequence of Theorem 2 is that the backbone of a Boolean formula can be computed with a logarithmic number of calls to a witness oracle. From a query complexity perspective, this represents an exponential improvement over the best known algorithms [23].

Corollary 1 *The function problem of computing the backbone of a propositional formula* FBB *(but also* FBBr*) [15] is in* $\mathrm{FP}^{\mathrm{NP}}[\mathsf{wit},\mathsf{log}]$.

Moreover, and using Theorem 3 and 6 we also get the following:

Corollary 2 FVInd *is in* $\mathrm{FP}^{\mathrm{NP}}[\mathit{wit},\mathit{log}]$.

2 refines the query complexity of $\mathrm{FP}^{\mathrm{NP}}$ suggested in [14]. If the oracle computes witness (e.g. a SAT solver), then a logarithmic number of calls suffices. The main result for predicates of form $\mathscr{P}$ is more restricted, due to the requirement of the problem having a unique minimal set. Nevertheless, and in those cases, one can compute a minimal set with a logarithmic number of calls to a witness oracle. One concrete example is the extraction of an MUS.

Corollary 3 FMUS *for formulas with a unique MUS is in* $\mathrm{FP}^{\mathrm{NP}}[\mathit{wit},\mathit{log}]$.

4 Complexity of Selecting k Minimal Sets

This section extends the results of the previous Section 3 to the case when it is known that there are exactly k minimal sets, for some constant k. Given the results of previous sections, form $\mathscr{P}$ is the only predicate form of interest, since function problems represented with predicates of either form $\mathscr{L}$ or $\mathscr{B}$ can be solved with a logarithmic number of witness oracle calls. We begin by a lemma that will let us relate the general case with the case $k = 1$ (i.e. Theorem 3).

Lemma 1 *Let P be a predicate of the form $\mathscr{P}$ such that $\mathcal{R}$ contains exactly k distinct minimal sets $\mathcal{M}_1, \ldots, \mathcal{M}_k$. For any $i \in 1..k$ there exists a set $\mathcal{D} \subseteq \mathcal{R}$ with $|\mathcal{D}| = k-1$ such that $\mathcal{R} \setminus \mathcal{D}$ contains one and only one minimal set, which is the set M_i.*

Proof. The sets $\mathcal{M}_j$ are such that the formula $\mathcal{G} \land \land_{u \in \mathcal{M}_j} \sigma(u)$ is unsatisfiable, for $j \in 1..k$. Further, removing any element from $\mathcal{M}_j$ makes this formula satisfiable, due to minimality. Hence, there are no $\mathcal{M}_j \subseteq \mathcal{M}_{j'}$ with $j \neq j'$. Put into $\mathcal{D}$ an element u_j from each $\mathcal{M}_j$, $j \neq i$, such that $u_j \notin \mathcal{M}_i$. (Observe that once $\mathcal{D}$ is removed from $\mathcal{R}$, unsatisfiability cannot be due to any $\mathcal{M}_j$, $j \neq i$.) Now $\mathcal{D}$ has at most $k-1$ elements. If $\mathcal{R} \setminus \mathcal{M}_i$ does not have enough elements to do so, add dummy elements to $\mathcal{R}$ in order to ensure that it has. (For instance always add $k-1$ elements to $\mathcal{R}$ before the construction.) □

Theorem 4 *Consider a function problem Γ, and let P denote the monotone predicate representing Γ, of form $\mathscr{P}$. Let Γ have exactly k minimal sets. Then $\Gamma \in \mathrm{FP}^{\mathrm{NP}}[\mathit{wit},\mathit{log}]$. Moreover, computing the k minimal sets is also in $\mathrm{FP}^{\mathrm{NP}}[\mathit{wit},\mathit{log}]$.*

Proof. (Sketch)
The proof is motivated by 1. It constructs a formula that examines the removal of all possible subsets of $\mathcal{R}$ of size $k-1$. Some of these removals leave a single minimal set in $\mathcal{R}$; gadgets of Theorem 3 are used to calculate this set.
Consider $\binom{m}{k-1} = \mathcal{O}(m^{k-1})$ subsets of $\mathcal{R}$, where for each set, $k-1$ elements are removed. Let $\mathcal{D}_t$, with $1 \leq t \leq \binom{m}{k-1}$, denote each of the subsets with $k-1$ elements of $\mathcal{R}$, and let $\mathcal{R}_t \triangleq \mathcal{R} \setminus \mathcal{D}_t$. For each $\mathcal{R}_t$ construct the gadget used in the proof of Theorem 3, where a set Z^t of z_i^t variables are associated with the

formulas created for each instance t. In addition, use an extra z variable, i.e. z^t_{m+1}, as follows:

$$z^t_{m+1} \leftrightarrow (\mathcal{G} \land \land_{u_i \in \mathcal{R}} \sigma(u_i))\,[X/X^t_{m+1}] \tag{9}$$

This variable serves to indicate whether $\mathcal{R}^t$ contains a minimal set. For each $\mathcal{R}_t$, one the following four cases can occur:

1. If $\mathcal{R}_t$ has at least two disjoint minimal sets, then none of the z^t_i variables can take value 1, and so no minimal set will be identified.
2. If $\mathcal{D}_t$ intersects $k-1$ of the k minimal sets, then $\mathcal{R}_t$ has exactly one minimal set. Thus, the z^t_i variables will identify the minimal set for instance t. This situation is guaranteed to occur by 1.
3. If there are no minimal sets in $\mathcal{R}_t$, then the predicate does not hold and so $\mathcal{G} \land \land_{u_i \in \mathcal{R}} \sigma(u_i)$ is satisfiable. Thus, variable z^t_{m+1} can be assigned value 1.
4. If all minimal sets in $\mathcal{R}_t$ intersect, then the z^t_i variables that take value 1 will be the ones associated with elements in the intersection of the minimal sets. This set of variables will necessarily be a subset of some other set covered by case 2.

We use MaxSAT to find all the z^t_i variables that can take value 1. Afterwards we discard the sets of z^t_i variables for those instances t that are contained in any other instance; this is a polynomial time procedure. We must also discard any z^t_{m+1} variables that take value 1, and the associated z^t_i variables. The total number of z^t_i variables, over all $\binom{m}{k-1}$ instances is $(m+1) \times \binom{m}{k-1} = \mathcal{O}(m^k)$. Therefore, binary search for computing the largest set of z^t_i variables that can take value 1 requires $\mathcal{O}(\log m^k) = \mathcal{O}(\log m)$ (for constant k) witness oracle calls. Thus, $\Gamma \in \mathrm{FP}^{\mathrm{NP}}[\mathsf{wit},\mathsf{log}]$. To conclude the proof, observe that the set of z^t_i variables gives all the k minimal sets. Thus, computing the k minimal sets is also in $\mathrm{FP}^{\mathrm{NP}}[\mathsf{wit},\mathsf{log}]$. □

5 Conclusions & Research Directions

The practical success of SAT solvers motivates their use in solving both decision and function problems. Many of these function problems are in $\mathrm{FP}^{\mathrm{NP}}$ but a more accurate query complexity characterization is not known. For propositional formulas, this is the case, for example, with computing an MUS (for CNF formulas) (e.g. [18, 5]), computing the backbone (e.g. [9, 23]), identifying the set of independent variables (e.g. [14]), or computing prime implicants/implicates, among many others. This paper studies the query complexity of function problems with a constant number of minimal sets, and shows that, if these problems can be represented with monotone predicates in certain general forms, then they are in $\mathrm{FP}^{\mathrm{NP}}[\mathsf{wit},\mathsf{log}]$, i.e. the class of function problems solved in polynomial time by using at most a logarithmic number of calls to a witness oracle [4, 12]. The main consequence of this result is that a large number of well-known function problems are shown to be solved with a logarithmic number of calls to a witness oracle. These include computing the backbone of a propositional formula,

the set of independent variables of a propositional formula, computing an MUS when there exist a constant number of MUSes, and also computing prime implicants (given a term) and prime implicates (given a clause), again when there exist a constant number.

A number of future research directions can be envisioned. First, it is open whether the results for monotone predicates of form $\mathscr{P}$ can be improved upon. This would have key consequences in long-standing open problems, e.g. computing an MUS [5]. Second, although the worst-case query complexity results in this paper improve significantly what practical algorithms have achieved for several function problems, it would be important to also investigate lower bounds on query complexity.

Acknowledgements

This work is partially supported by SFI PI grant BEACON (09/IN.1/I2618), FCT grants ATTEST (CMU-PT/ELE/0009/2009), POLARIS (PTDC/EIA-CCO/123051/2010), and by INESC-ID multiannual funding from the PIDDAC program funds.

References

[1] A. Biere, M. Heule, H. van Maaren, and T. Walsh, editors. *Handbook of Satisfiability*, volume 185 of *Frontiers in AI and Applications*. IOS Press, 2009.

[2] A. R. Bradley and Z. Manna. Checking safety by inductive generalization of counterexamples to induction. In *FMCAD*, pages 173–180. IEEE Press, 2007.

[3] A. R. Bradley and Z. Manna. Property-directed incremental invariant generation. *Formal Asp. Comput.*, 20(4-5):379–405, 2008.

[4] S. R. Buss, J. Krajíček, and G. Takeuti. Provably total functions in the bounded arithmetic theories R^i_3, U^i_2, and V^i_2. In P. Clote and J. Krajíček, editors, *Arithmetic, Proof Theory, and Computational Complexity*, pages 116–161. OUP, 1995.

[5] Z.-Z. Chen and S. Toda. The complexity of selecting maximal solutions. *Inf. Comput.*, 119(2):231–239, 1995.

[6] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1979.

[7] G. Gottlob and C. G. Fermüller. Removing redundancy from a clause. *Artif. Intell.*, 61(2):263–289, 1993.

[8] B. Jenner and J. Torán. The complexity of obtaining solutions for problems in NP and NL. In *Complexity theory retrospective II*, pages 155–178. Springer, 1997.

[9] P. Kilby, J. K. Slaney, S. Thiébaux, and T. Walsh. Backbones and backdoors in satisfiability. In M. M. Veloso and S. Kambhampati, editors, *AAAI*, pages 1368–1373. AAAI Press / The MIT Press, 2005.

[10] H. Kleine Büning and T. Letterman. *Propositional Logic: Deduction and Algorithms*. Cambridge University Press, 1999.

[11] A. Komuravelli, A. Gurfinkel, S. Chaki, and E. M. Clarke. Automatic abstraction in SMT-based unbounded software model checking. In Sharygina and Veith [21], pages 846–862.

[12] J. Krajíček. *Bounded Arithmetic, Propositional Logic and Complexity Theory*. Cambridge University Press, 1995.

[13] M. W. Krentel. The complexity of optimization problems. *J. Comput. Syst. Sci.*, 36(3):490–509, 1988.

[14] J. Lang, P. Liberatore, and P. Marquis. Propositional independence: Formula-variable independence and forgetting. *J. Artif. Intell. Res.*, 18:391–443, 2003.

[15] J. Marques-Silva and M. Janota. Computing minimal sets on propositional formulae I: Problems & reductions. *CoRR*, arXiv:1402.3011, 2014. Available at `http://arxiv.org/abs/1310.2491`.

[16] J. Marques-Silva, M. Janota, and A. Belov. Minimal sets over monotone predicates in Boolean formulae. In Sharygina and Veith [21], pages 592–607.

[17] C. H. Papadimitriou. *Computational Complexity*. Addison Wesley, 1994.

[18] C. H. Papadimitriou and D. Wolfe. The complexity of facets resolved. *J. Comput. Syst. Sci.*, 37(1):2–13, 1988.

[19] M. Schaefer and C. Umans. Completeness in the polynomial-time hierarchy: A compendium. *SIGACT news*, 33(3):32–49, 2002.

[20] A. L. Selman. A taxonomy of complexity classes of functions. *J. Comput. Syst. Sci.*, 48(2):357–381, 1994.

[21] N. Sharygina and H. Veith, editors. *Computer Aided Verification (CAV)*, 2013.

[22] G. S. Tseitin. On the complexity of derivation in propositional calculus. *Studies in Constructive Mathematics and Mathematical Logic, Part II*, pages 115–125, 1968.

[23] C. S. Zhu, G. Weissenbacher, and S. Malik. Post-silicon fault localisation using maximum satisfiability and backbones. In P. Bjesse and A. Slobodová, editors, *FMCAD*, pages 63–66. FMCAD Inc., 2011.

ECCC ISSN 1433-8092

http://eccc.hpi-web.de