Counting Maximal Satisfiable Subsets

Jaroslav Bendík,[1] **Kuldeep S. Meel**[2]
[1] Masaryk University, Brno, Czech Republic
[2] National University of Singapore, Singapore

Abstract

Given an unsatisfiable set of constraints F, a maximal satisfiable subset (MSS) is a maximal subset of constraints $C \subseteq F$ such that C is satisfiable. Over the past two decades, the steady improvement in runtime performance of algorithms for finding MSSes has led to increased adoption of MSS-based techniques in a wide variety of domains. Motivated by the progress in finding an MSS, the past decade has witnessed a surge of interest in the design of algorithmic techniques to enumerate all the MSSes, which has subsequently led to a discovery of new applications utilizing enumeration of MSSes.

The development of techniques for finding and enumeration of MSSes mirrors a similar phenomenon of finding and enumeration of SAT solutions in the early 2000s, which subsequently motivated the design of algorithmic techniques for model counting. In a similar spirit, we undertake a study to investigate the feasibility of MSS counting techniques. In particular, the focus point of our investigation is to answer whether one can design efficient MSS counting techniques that do not rely on explicit MSS enumeration. The primary contribution of this work is an affirmative answer to the above question in the form of a novel algorithm. The algorithm uses a novel architecture of a wrapper $\mathcal{W}$ and a remainder $\mathcal{R}$ such that the desired MSS count can be expressed as $|\mathcal{W}| - |\mathcal{R}|$. To efficiently compute $|\mathcal{W}|$ and $|\mathcal{R}|$, the algorithm relies on the advances in projected model counting. Our empirical evaluation demonstrates that our approach can scale to instances clearly beyond the reach of enumeration-based techniques.

1 Introduction

Logical constraints have emerged as a prominent representation language to model environments and agents. A set of constraints is called satisfiable if there exists an assignment to variables such that all the constraints are satisfied. Similarly, a set of constraints is unsatisfiable (also referred to as *over-constrained*) if there does not exist an assignment that would satisfy the constraints (Meseguer et al. 2003). If we are given an unsatisfiable set of constraints, the goal is often to analyze the unsatisfiability. In such cases, two notions are of particular interest: a Minimal Unsatisfiable Subset (MUS), which is a minimal subset of constraints that is unsatisfiable, and a Maximal Satisfiable Subset (MSS), which is a maximal subset of constraints that is satisfiable (Liffiton and Sakallah 2008; Belov, Lynce, and Marques-Silva 2012; Reiter 1987). A dual notion to an MSS is that of a Minimal Correction Subset (MCS), which is a minimal subset of constraints that need to be relaxed. Formally, given a set of constraints F, a set $C \subseteq F$ is a MSS of F iff $F \setminus C$ is a MCS of F.

The design of modern AI systems often encounters over-constrained systems and in such scenarios, MCSes and MSSes often play a major role (Bailey and Stuckey 2005; Liffiton and Sakallah 2008). In particular, in the field of diagnosis of systems, an MCS represents the constraints that need to be relaxed for the system to be conflict-free. Similarly, in the context of a belief update and argumentation, an MSS plays a key role in the update of belief in the presence of an incoming contradictory belief. We refer the reader to (Besnard, Grégoire, and Lagniez 2015) for further discussion. Often, finding a single MSS (resp. MCS) is not sufficient, and we seek to enumerate all MSSes or to count the number of MSSes. Especially, the MSS count serves as a good diagnostic metric (Thimm 2018; Hunter and Konieczny 2008), or as an indicator of the feasibility of the complete MSS enumeration.

Our interest in counting the number of MSSes is motivated by the rise of Beyond NP paradigm wherein the progress in the design of efficient techniques for satisfiability paved the way for interest in the design of efficient techniques for problems such as counting, sampling, optimization, and the like. In particular, the past two decades have witnessed a proliferation of efficient techniques for model counting, also denoted as #SAT. It is worth remarking that initial studies into model counting were motivated by applications in Bayesian inference but the subsequent availability of efficient model counting techniques have now led to several new applications ranging from neural network verification (Baluta et al. 2019), quantified information flow (Biondi et al. 2018), computational biology (Sashittal and El-Kebir 2020), network reliability, and the like. In this regard, we view that given the availability of efficient techniques for finding an MSS/MCS, it is the right time to pursue an investigation into the design of efficient counting techniques for MSSes/MCSes, and the availability of efficient counting techniques for MCSes/MSSes would lead to a discovery of a diverse set of applications for MSS counting.

Similarly to the early years of research into #SAT, the best-known technique, as of now, to perform the MSS counting is to employ state of the art techniques for complete MSS enumeration. While MSS enumeration techniques have improved over the years, the complete MSS enumeration is often practically intractable since there can be up to exponentially many MSSes w.r.t. the size of the input constraint set. In this context, the primary research question that we seek to investigate is *whether we can design MSS counting techniques that do not necessarily rely on enumeration?*. We envision development of MSS counting techniques to take advantage of the progress in the model counting techniques. Given that the problem of finding an MSS is in $\mathrm{FP}^{\mathrm{NP}[\log n]}$ (i.e., harder than the classical SAT problem), a natural target problem is projected model counting, a generalization of the classical model counting problem; the projected model counting is known to be #NP-hard in contrast to #P-completeness of classical model counting.

The primary contribution of this paper is an affirmative answer to the above question. In particular, we design a new algorithmic framework that uses a novel architecture of a wrapper $\mathcal{W}$ and a remainder $\mathcal{R}$ such that the desired MSS count corresponding to the formula F is $|\mathcal{W}| - |\mathcal{R}|$. We encode the wrapper $\mathcal{W}$ and the remainder $\mathcal{R}$ via Boolean formulas $\mathbb{W}$ and $\mathbb{R}$ with suitable projection sets such that the projected model count of $\mathbb{W}$ and $\mathbb{R}$ is equal to $|\mathcal{W}|$ and $|\mathcal{R}|$ respectively. We present four different strategies for the construction of wrappers (and their corresponding remainders) and observe the soundness of a composition of different wrappers. The reduction to projected model counting allows us to build on recent advances in the design of efficient component caching-based projected model counting techniques. To demonstrate the empirical effectiveness of our approach, we implemented a Python-based prototype and performed a detailed empirical analysis. Out of 1200 benchmarks, the enumeration-based techniques can solve 353 benchmarks while our approach can solve 510 benchmarks.

2 Prelimilaries and Problem Formulation

We use standard definitions for propositional logic. A propositional formula F is built over a set of *literals*, where a *literal* is either a Boolean variable or its negation. $Vars(F)$ denotes the set of variables used in F. A *valuation* π of a finite set A of variables is a mapping from A to $\{1, 0\}$. $Vals(F)$ denotes the set of all valuations of $Vars(F)$. Given a valuation π and a formula F, we write $F[\pi]$ to denote the substitution of each variable x in the domain of π by the value $\pi(x)$; furthermore, we apply trivial simplifications, e.g., $G \vee 0 = G$, $G \wedge 0 = 0$, etc. Observe that if $A \supseteq Vars(F)$, then $F[\pi]$ is simplified either to 1 or to 0. If $A \supseteq Vars(F)$ and $F[\pi] = 1$, we write $\pi \models F$ and we call π a *model* of F; otherwise, if $F[\pi] = 0$, we write $\pi \not\models F$. A formula is *satisfiable* iff it has a model, and, otherwise, it is *unsatisfiable*. We write M_F to denote the set of all models of a formula F. Furthermore, for a set A of variables such that $A \subseteq Vars(F)$, we write $M_{F\downarrow A}$ to denote the projection of M_F on A, and for $\pi \in M_F$, we write $\pi_{\downarrow A}$ to denote the projection of π on A. Two formulas F and G are *equivalent*, denoted $F \equiv G$, iff $M_F = M_G$.

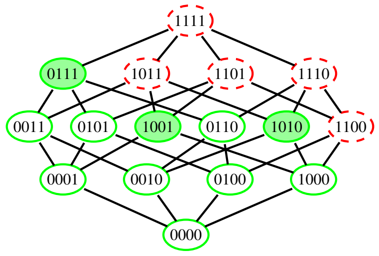

Figure 1: Illustration of $\mathcal{P}(F)$ from the Example 1. We denote individual subsets of F as bit-vectors, e.g., $\{f_1, f_3\}$ is written as 1010. The subsets with a dashed border are the unsatisfiable subsets, and the others are satisfiable subsets. The MSSes are filled with a background color.

A CNF formula is a conjunction of disjunctions (*clauses*) of literals. A CNF formula can be also viewed as a multiset of clauses where a clause is a set of literals; the two representations are used interchangeably and are clear from the context. Throughout the whole text, we use F to denote the input CNF formula of interest. We use capital letters, e.g., S, K, N, or blackboard bold letters, e.g., $\mathbb{W}$, $\mathbb{R}$, to denote other formulas, small letters, e.g., f, f_1, f_i, to denote clauses, and small letters, e.g., x, x', y, to denote variables. Finally, given a set X, we write $\mathcal{P}(X)$ to denote the powerset of X, and $|X|$ to denote the cardinality of X.

Definition 1 (MSS)**.** *A set N, $N \subseteq F$, is a* maximal satisfiable subset (MSS) *of F iff N is satisfiable and every N', such that $N \subsetneq N' \subseteq F$, is unsatisfiable.*

Definition 2 (MCS)**.** *A set N, $N \subseteq F$, is a* minimal correction subset (MCS) *of F iff $F \setminus N$ is satisfiable and for all N', such that $N' \subsetneq N$, the set $F \setminus N'$ is unsatisfiable. Equivalently, N is a MCS of F iff $F \setminus N$ is a MSS of F.*

Note that the maximality is the set maximality and not the maximum cardinality as e.g. in the MaxSAT problem. Consequently, there can be MSSes with different cardinalities, and in total, there can be up to exponentially many MSSes of F w.r.t. $|F|$ (see the Sperner's theorem (Sperner 1928)). The same applies also to MCSes. We write $\mathtt{MSS}_F$ to denote the set of all MSSes of F, $\mathtt{MCS}_F$ to denote the set of all MCSes of F, $\mathtt{SS}_F$ to denote the set of all satisfiable subsets of F, and $\mathtt{nonMSS}_F$ to denote the set $\mathtt{SS}_F \setminus \mathtt{MSS}_F$ of all satisfiable subsets of F that are not MSSes.

Example 1. We demonstrate the concepts of MSSes and MCSes on an example, illustrated in Fig. 1. Assume that $F = \{f_1 = \{x_1\}, f_2 = \{\neg x_1\}, f_3 = \{x_2\}, f_4 = \{\neg x_1, \neg x_2\}\}$. There are 3 MSSes: $\mathtt{MSS}_F = \{\{f_2, f_3, f_4\}, \{f_1, f_4\}, \{f_1, f_3\}\}$, and thus also 3 MCSes: $\mathtt{MCS}_F = \{\{f_1\}, \{f_2, f_3\}, \{f_2, f_4\}\}$.

Problem Definitions

In this paper, we are concerned with the following three problems.

Name: #MSS
Input: A formula F.
Output: The number $|\mathtt{MSS}_F|$ of MSSes of F.

Name: #MCS
Input: A formula F.
Output: The number $|\mathtt{MCS}_F|$ of MCSes of F.

Name: proj-#SAT
Input: A formula F and a set of variables $S \subseteq \mathit{Vars}(F)$.
Output: The number $M_{F\downarrow S}$ of models of F projected on S.

Our goal is to solve the #MCS and #MSS problems. Since MCSes are complements of MSSes, the two problems are equivalent. Thus, in the rest of the paper, we focus only on the #MSS problem. Finally, we do not focus on solving the proj-#SAT problem; instead, we propose several reductions of #MSS to proj-#SAT.

3 Related Work

MSS Counting As far as we know, there is no algorithm dedicated to counting MSSes. A straightforward approach to determine the count is to enumerate all the MSSes via an MSS enumeration algorithm, e.g., (Bailey and Stuckey 2005; Stern et al. 2012; Liffiton et al. 2016; Marques-Silva et al. 2013; Bendík et al. 2016; Narodytska et al. 2018; Previti et al. 2018; Bendík and Černá 2020), and then simply count the enumerated MSSes. However, the complete MSS enumeration is often practically intractable, since there can be exponentially many MSSes w.r.t. $|F|$ and thus, the MSSes just cannot be explicitly enumerated in a reasonable time.

Another possible solution how to count MSSes (MCSes) is based on a well-known duality between MCSes and so-called *minimal unsatisfiable subsets* (MUSes) of F. A set N is a MUS of F iff N is unsatisfiable and $\forall f \in N$ the set $N \setminus \{f\}$ is satisfiable. The *hitting set duality* (Reiter 1987; de Kleer and Williams 1987) between MUSes and MCSes says that every MCS is a *minimal hitting set* of the set of all MUSes of F. Consequently, one can first use a MUS enumeration tool, e.g., (Bailey and Stuckey 2005; Stern et al. 2012; Liffiton et al. 2016; Bacchus and Katsirelos 2015, 2016; Narodytska et al. 2018; Bendík, Černá, and Beneš 2018; Liu and Luo 2018; Bendík and Černá 2020a,b), to identify all MUSes of F, and then count the number of minimal hitting sets of the MUSes. The problem is that there can be also exponentially many MUSes w.r.t. $|F|$, which makes the MUS enumeration also often practically intractable.

It is also worth remarking a recent hashing-based approach, called AMUSIC, to approximate counting of MUSes of a given formula (Bendík and Meel 2020). AMUSIC identifies an approximate MUS count via polynomially many calls of a Σ_3^P oracle. In contrast, we focus on the exact counting and while the notions of MUSes and MSSes share a close relationship, we are unaware of any efficient reduction of the MSS counting problem to the MUS counting problem.

Model Counting In his seminal paper (Valiant 1979), Valiant showed that the problem of propositional model counting (i.e., proj-#SAT when $S = \mathit{Vars}(F)$) is #P-complete. Durand, Hermann, and Kolaitis (2005) showed that the general problem of proj-#SAT is #NP-hard. From a practical perspective, the earliest work on model counting dates to Birnbaum and Lozinskii (1999), which sought to rely on *smarter* enumeration strategies of partial solutions. Subsequently, Bayardo and Pehoushek (Bayardo Jr and Pehoushek 2000) introduced the notion of component caching, wherein a residual formula after substituting the current partial assignment can be partitioned into different subsets of clauses such that these subsets do not share variables. Each of these subsets is called a component, and the model count of the formula is obtained by multiplying the corresponding counts for each of the components. Therefore, the model count is often determined by explicitly identifying only a fraction of all models. Caching scheme is used to avoid recomputation as similar components appear in different parts of the search space. Over the past two decades, there has been a series of algorithmic and system-driven improvements of component caching-based model counting techniques (Sang et al. 2004; Sang, Beame, and Kautz 2005; Thurley 2006; Muise et al. 2012; Sharma et al. 2019). There has been also an extensive research on compilation-based model counting including e.g. C2D (Darwiche 2004), SDD (Darwiche 2011), and D4 (Lagniez and Marquis 2017).

Over the past 5 years, there has been a concentrated effort on developing efficient projected model counting techniques (Chakraborty et al. 2014; Aziz et al. 2015; Chakraborty, Meel, and Vardi 2016; Möhle and Biere 2018; Sharma et al. 2019; Lagniez and Marquis 2019). These techniques rely on appropriate modifications of standard propositional model counters (as described above). In this work, we rely on the state of the art projected model counter GANAK (Sharma et al. 2019), which was part of the system that won the Projected model counting track at the recently organized model counting competition.

4 Counting the Number of MSSes

4.1 Basic Idea

Our approach for finding the MSS count $|\mathtt{MSS}_F|$ is based on a simple observation: one can count the number $|\mathtt{SS}_F|$ of all satisfiable subsets of F, the number $|\mathtt{nonMSS}_F|$ of satisfiable subsets that are not MSSes, and then do the math: $|\mathtt{MSS}_F| = |\mathtt{SS}_F| - |\mathtt{nonMSS}_F|$. In fact, even a more general observation holds:

Definition 3 (wrapper and remainder)**.** *A set $\mathcal{W}$ of subsets of F is a* wrapper *iff $\mathtt{MSS}_F \subseteq \mathcal{W} \subseteq \mathtt{SS}_F$. Futhermore, the* remainder *of $\mathcal{W}$ is the set $\mathcal{R} = \mathcal{W} \cap \mathtt{nonMSS}_F$.*

Proposition 1. Let $\mathcal{W}$ be a wrapper and $\mathcal{R}$ its remainder. Then $|\mathtt{MSS}_F| = |\mathcal{W}| - |\mathcal{R}|$.

Proof. $\mathtt{SS}_F = \mathtt{MSS}_F \cup \mathtt{nonMSS}_F$, and $\mathtt{MSS}_F \cap \mathtt{nonMSS}_F = \emptyset$, hence $\mathtt{MSS}_F = \mathcal{W} \setminus \mathtt{nonMSS}_F = \mathcal{W} \setminus \mathcal{R}$, and since $\mathcal{R} \subseteq \mathcal{W}$, it holds $|\mathtt{MSS}_F| = |\mathcal{W}| - |\mathcal{R}|$. $\square$

Our approach for counting MSSes consists of the following steps. First, we find a *suitable* wrapper $\mathcal{W}$ and the corresponding remainder $\mathcal{R}$. Then, we encode the wrapper $\mathcal{W}$ and the corresponding remainder $\mathcal{R}$ with two formulas, $\mathbb{W}$ and $\mathbb{R}$, such that each *projected model* of $\mathbb{W}$ and $\mathbb{R}$ corresponds to an element of $\mathcal{W}$ and $\mathcal{R}$, respectively. Finally, we use a projected model counting tool to count the models of $\mathbb{W}$ and $\mathbb{R}$, and hence to determine $|\mathcal{W}|$ and $|\mathcal{R}|$ which yields also the MSS count $|\texttt{MSS}_F|$ (Proposition 1). In the following section, we provide details on *what is* and *how to find* a suitable wrapper, how to build the formulas $\mathbb{W}$ and $\mathbb{R}$, and what is the projection set.

4.2 Wrappers and Remainders

In this section, we gradually present 4 different wrappers $\mathcal{W}_1, \ldots, \mathcal{W}_4$ and their corresponding remainders $\mathcal{R}_1, \ldots, \mathcal{R}_4$. For each wrapper $\mathcal{W}_i$ and its remainder $\mathcal{R}_i$, we also build the corresponding formulas $\mathbb{W}_i$ and $\mathbb{R}_i$. Subsequently, we show how to combine multiple wrappers into a single, possibly more efficient, wrapper.

Wrapper $\mathcal{W}_1$ Our first wrapper $\mathcal{W}_1$ is simply the set $\texttt{SS}_F$ of all satisfiable subsets of F, and the corresponding remainder $\mathcal{R}_1$ is thus $\texttt{SS}_F \cap \texttt{nonMSS}_F = \texttt{nonMSS}_F$. We build the formula $\mathbb{W}_1$ using the variables $\mathit{Vars}(F)$ of F and an additional set of *activation variables* $A = \{a_f \,|\, f \in F\}$:

$$\mathbb{W}_1 = \bigwedge_{f \in F} (f \vee \neg a_f) \tag{1}$$

Intuitively, by setting a variable a_f to 1, we *activate* the sub-clause f in the clause $f' = (f \vee \neg a_f)$ of $\mathbb{W}_1$ since satisfying f is then the only way to satisfy f'. Let us denote by $AF(\pi)$ the one-to-one mapping (bijection) between a valuation π of A and the corresponding set of activated clauses of F, i.e., $AF(\pi) = \{f \in F \,|\, \pi(a_f) = 1\}$.

Proposition 2. For every valuation π of A, $\pi \in M_{\mathbb{W}_1 \downarrow A}$ iff $AF(\pi) \in \mathcal{W}_1$. Consequently, $|M_{\mathbb{W}_1 \downarrow A}| = |\mathcal{W}_1|$.

Proof. $\Rightarrow$: Let π' be a model of $\mathbb{W}_1$ such that $\pi'_{\downarrow A} = \pi$. We show that $\pi' \models AF(\pi)$, hence $AF(\pi)$ is satisfiable and belongs to $\mathcal{W}_1$. For every $f \in AF(\pi)$, we know that $\pi' \models (f \vee \neg a_f)$. Moreover, by the definition of $AF(\pi)$, $\pi(a_f) = 1 = \pi'(a_f)$, i.e., $\pi' \not\models \neg a_f$, and thus $\pi' \models f$.
$\Leftarrow$: Let N be an element of $\mathcal{W}_1$ and ϕ its model. We define a valuation π' of $\mathbb{W}_1$ as $\pi'(x) = \phi(x)$ if $x \in \mathit{Vars}(F)$, $\pi'(a_f) = 0$ if $f \notin N$, and $\pi'(a_f) = 1$ if $f \in N$. Observe that $AF(\pi'_{\downarrow A}) = N$. Furthermore, $\pi' \models \mathbb{W}_1$: every clause $(f \vee \neg a_f) \in \mathbb{W}_1$ such that $f \in N$ is inherently satisfied by ϕ, and every clause $(f \vee \neg a_f) \in \mathbb{W}_1$ such that $f \notin N$ is satisfied by $\pi'(a_f) = 0$. Hence, $\pi = \pi'_{\downarrow A} \in M_{\mathbb{W}_1 \downarrow A}$ $\square$

To build the formula $\mathbb{R}_1$ that encodes the remainder $\mathcal{R}_1 = \texttt{SS}_F \cap \texttt{nonMSS}_F$, i.e., the set of all satisfiable subsets of F that are not MSSes, we introduce another set B of *activation variables* $B = \{b_f \,|\, f \in F\}$. Similarly as in the case of A, given a valuation π of B, let us by $BF(\pi)$ denote the subset $\{f \in F \,|\, \pi(b_f) = 1\}$ of F. By the definition of a MSS, a satisfiable subset N of F is not a MSS iff there exists a satisfiable N' such that $N \subsetneq N' \subseteq F$. We use $AF(\pi)$ and $BF(\pi)$ to encode such N and N', respectively, in $\mathbb{R}_1$:

$$\mathbb{R}_1 = \mathbb{W}_1 \wedge \bigwedge_{f' \in F'} (f' \vee \neg b_f) \wedge \bigwedge_{f \in F} (a_f \implies b_f) \wedge \bigvee_{f \in F} (\neg a_f \wedge b_f) \tag{2}$$

Intuitively, the first conjunct $\mathbb{W}_1$ encodes that $AF(\pi)$ is satisfiable, the second conjunct encodes that $BF(\pi)$ is satisfiable, and the last two conjuncts express that $AF(\pi) \subsetneq BF(\pi)$. Note that since both the first conjuncts reason about satisfiability of subsets of F, we use in the second conjunct a primed version F' of F, i.e. a copy of F where each literal is substituted by its primed version.

Proposition 3. For every valuation π of A, $\pi \in M_{\mathbb{R}_1 \downarrow A}$ iff $AF(\pi) \in \mathcal{R}_1$. Consequently, $|M_{\mathbb{R}_1 \downarrow A}| = |\mathcal{R}_1|$.

Proof. $\Rightarrow$: Let π' be a model of $\mathbb{R}_1$ such that $\pi'_{\downarrow A} = \pi$. Since $\mathbb{R}_1$ subsumes $\mathbb{W}_1$, $\pi' \models \mathbb{W}_1$, and hence by Proposition 2 $AF(\pi)$ is satisfiable. The set $BF(\pi'_{\downarrow B})$ is defined and constrained in $\mathbb{R}_1$ analogously to $AF(\pi)$, thus $BF(\pi'_{\downarrow B})$ is also satisfiable. Furthermore, $AF(\pi) \subsetneq BF(\pi'_{\downarrow B})$ since $\pi' \models \bigwedge_{f \in F}(a_f \implies b_f) \wedge \bigvee_{f \in F}(\neg a_f \wedge b_f)$.
$\Leftarrow$: Let N be an element of $\mathcal{R}_1$, N' a satisfiable superset of N, ϕ a model of N, and ϕ' a model of N'. We build a model π' of $\mathbb{R}_1$ as $\pi'(x) = \phi(x)$ if $x \in \mathit{Vars}(F)$, $\pi'(a_f) = 0$ if $f \notin N$, $\pi'(a_f) = 1$ if $f \in N$, $\pi'(x) = \phi'(x')$ if $x' \in \mathit{Vars}(F')$, $\pi'(b_f) = 0$ if $f \notin N'$, and $\pi'(b_f) = 1$ if $f \in N'$. Analogously to the proof of Proposition 2, we know that $\pi' \models \mathbb{W}_1 \wedge \bigwedge_{f' \in F'}(f' \vee \neg b_f)$. As for the remaining part of $\mathbb{R}_1$, since $N \subsetneq N'$, we have that $\pi'(a_f) = 1$ implies $\pi'(b_f) = 1$ for every $f \in F$, and that there is at least one $f \in F$ such that $\pi'(a_f) = 0$ and $\pi'(b_f) = 1$. $\square$

Based on the above observations, we can use a projected model counter to determine the cardinalities $|\mathcal{W}_1|$ and $|\mathcal{R}_1|$, and then employ Proposition 1 to deduce the MSS count. However, due to the complexity of projected model counting, determining the model counts for $\mathbb{W}_1$ and $\mathbb{R}_1$ can be practically intractable. In general, there are two main criteria that affect the practical efficiency of the projected model counting. One criterion is the cardinality of the projection set, which is in the case of $\mathcal{W}_1$ $|A| = |F|$. The other criterion is the number of the models, i.e., $|\mathcal{W}_1|$, which can be exponential w.r.t. $|F|$ since there are $2^{|F|}$ subsets of F and all of them (excluding the whole F) can be satisfiable. In the following, we propose three other wrappers (and corresponding remainders) that tend to optimize these two criteria by providing a better description of MSSes. All formulas that encode the following wrappers and their remainders use the same variables as in the case of $\mathcal{W}_1$ and $\mathcal{R}_1$, i.e., $\mathit{Vars}(F) \cup \mathit{Vars}(F') \cup A \cup B$. We also use the notation $AF(\pi)$ and $BF(\pi)$ to map valuations of A and B, respectively, to subsets of F.

Wrapper $\mathcal{W}_2$ Our second wrapper, $\mathcal{W}_2$, exploits the intersection $\texttt{IMSS}_F$ of all MSSes of F. Clearly, for every MSS $N \in \texttt{MSS}_F$ it holds that $\texttt{IMSS}_F \subseteq N$. Thus, we could define the next wrapper as the set of all satisfiable subsets of F that are supersets of $\texttt{IMSS}_F$. Unfortunately, computing the intersection can be very expensive (see below), thus, we exploit a more general observation:

Observation 1. For every MSS $N \in \texttt{MSS}_F$ and every underapproximation I of $\texttt{IMSS}_F$, i.e., $I \subseteq \texttt{IMSS}_F$, it holds that $I \subseteq N$.

Given an under-approximation I of $\texttt{IMSS}_F$, we define the second wrapper as $\mathcal{W}_2 = \{N \in \texttt{SS}_F \,|\, I \subseteq N\}$. The formulas $\mathbb{W}_2$ and $\mathbb{R}_2$ that encode $\mathcal{W}_2$ and $\mathcal{R}_2$, respectively, are defined as follows:

$$\mathbb{W}_2 = \mathbb{W}_1 \wedge \bigwedge_{f \in I} a_f \quad (3)$$

$$\mathbb{R}_2 = \mathbb{W}_2 \wedge \mathbb{R}_1 \quad (4)$$

Proposition 4. For every valuation π of A, $\pi \in M_{\mathbb{W}_2 \downarrow A}$ iff $AF(\pi) \in \mathcal{W}_2$. Consequently, $|M_{\mathbb{W}_2 \downarrow A}| = |\mathcal{W}_2|$.

Proof. $\Rightarrow$: $\mathbb{W}_2$ subsumes $\mathbb{W}_1$, thus for every $\pi \in M_{\mathbb{W}_2 \downarrow A}$ the set $AF(\pi)$ is satisfiable (Proposition 2), and since $f \in AF(\pi)$ iff $\pi \models a_f$, $\bigwedge_{f \in I} a_f$ ensures that $I \subseteq AF(\pi)$.
$\Leftarrow$: Given $N \in \mathcal{W}_2$ and a model ϕ of N, we build a valuation π' of $\mathbb{W}_2$ as $\pi'(x) = \phi(x)$ if $x \in \mathit{Vars}(F)$, $\pi'(a_f) = 0$ if $f \notin N$, and $\pi'(a_f) = 1$ if $f \in N$. Similarly as in the proof of Proposition 2, observe that $AF(\pi'_{\downarrow A}) = N$ and that $\pi' \models \mathbb{W}_2$, thus $\pi = \pi'_{\downarrow A} \in M_{\mathbb{W}_2 \downarrow A}$. □

Proposition 5. For every valuation π of A, $\pi \in M_{\mathbb{R}_2 \downarrow A}$ iff $AF(\pi) \in \mathcal{R}_2$. Consequently, $|M_{\mathbb{R}_2 \downarrow A}| = |\mathcal{R}_2|$.

Proof. $M_{\mathbb{R}_2 \downarrow A} = M_{(\mathbb{W}_2 \wedge \mathbb{R}_1) \downarrow A} = M_{\mathbb{W}_2 \downarrow A} \cap M_{\mathbb{R}_1 \downarrow A} = \{\pi \,|\, AF(\pi) \in \mathcal{W}_2\} \cap \{\pi \,|\, AF(\pi) \in \mathcal{R}_1\} = \{\pi \,|\, AF(\pi) \in \mathcal{W}_2 \cap \mathcal{R}_1\} = \{\pi \,|\, AF(\pi) \in \mathcal{R}_2\}$. The other direction holds since AF is a one-to-one mapping (bijection). □

Note that by enforcing the variables $\{a_f \,|\, f \in I\}$ to be set to 1, we effectively reduce the size of the projection set A since all models of $\mathbb{W}_2$ (and $\mathbb{R}_2$) agree on the assignment to these variables. In other words, $|M_{\mathbb{W}_2 \downarrow A}| = |M_{\mathbb{W}_2 \downarrow (A \setminus \{a_f \,|\, f \in I\})}|$ (and $|M_{\mathbb{R}_2 \downarrow A}| = |M_{\mathbb{R}_2 \downarrow (A \setminus \{a_f \,|\, f \in I\})}|$). Based on our practical experience, the $\texttt{IMSS}_F$ is often relatively large, i.e., I can be also relatively large, and thus we can significantly reduce the projection set.

The remaining question is how to compute either exactly $\texttt{IMSS}_F$ or at least its under-approximation I. We are not aware of any work that would be dedicated to computing the intersection $\texttt{IMSS}_F$ of all MSSes of F. Yet, as shown in (Kullmann 2000a), it holds that $\texttt{IMSS}_F = F \setminus \texttt{UMUS}_F$ where $\texttt{UMUS}_F$ is the union of all minimal unsatisfiable subsets (MUSes) of F. Unfortunately, based on a recent study (Mencía et al. 2019), computing $\texttt{UMUS}_F$, and hence also $\texttt{IMSS}_F$, is often practically intractable even for relatively small formulas. On the other hand, it is often possible to cheaply compute a good over-approximation of $\texttt{UMUS}_F$ via the concepts of *autark variables* and *lean kernel*. A set $A \subseteq \mathit{Vars}(F)$ is an *autark* of F iff there exists a truth assignment to A such that every clause of F that contains a variable from A is satisfied by the assignment (Monien and Speckenmeyer 1985). It is known (Kleine Büning and Kullmann 2009; Kullmann 2000b) that the union of two autark sets is also an autark set, and thus there exist a unique largest autark set of F. The *lean kernel* K of F is the set of all clauses that do not contain any variable from the largest autark set. It holds (Kleine Büning and Kullmann 2009; Kullmann 2000b) that the lean kernel K of F is an over-approximation of $\texttt{UMUS}_F$. Consequently, $F \setminus K$ is an under-approximation of $\texttt{IMSS}_F$. We employ an approach from (Marques-Silva et al. 2014) to compute the lean kernel K and then use $I = F \setminus K$ to build the wrapper $\mathcal{W}_2$.

Similarly as we used the intersection $\texttt{IMSS}_F$, one might think of exploiting the union $\texttt{UMSS}_F$ of all MSSes; clearly, every MSS of F is contained in $\texttt{UMSS}_F$. Thus, at first glance, it makes sense to build a wrapper that contains all satisfiable subsets of F that are subsets of $\texttt{UMSS}_F$. Yet, we observe that $\texttt{UMSS}_F = F$ and thus the use of the union would not bring any benefit compared to $\mathcal{W}_1$:

Proposition 6. For every formula F (with no empty clause) and the union $\texttt{UMSS}_F$ of all MSSes of F, it holds that $F = \texttt{UMSS}_F$.

Proof. By contradiction, assume a clause $f \in F$ that is not contained in any MSS of F. Since f is non-empty, it is necessarily satisfiable, and hence either $\{f\}$ is a MSS of F or there exists an MSS N of F such that $N \supsetneq \{f\}$. □

Wrapper $\mathcal{W}_3$ Our next wrapper, $\mathcal{W}_3$, is based on the following characterization of MSSes:

Proposition 7. For every MSS N of F there exists a valuation ϕ of $\mathit{Vars}(F)$ such that $\phi \models N$ and for every $f \in F \setminus N$ it holds that $\phi \not\models f$.

Proof. N is satisfiable, thus it has a model. For every model ϕ of N, if $\phi \models f$ for some $f \in F \setminus N$, then $\phi \models N \cup \{f\}$ which contradicts that N is a MSS. □

The corresponding wrapper for the property stated in Propositon 7 is $\mathcal{W}_3 = \{N \in \texttt{SS}_F \,|\, \exists \phi \in \mathit{Vals}(F). \phi \models N \wedge \bigwedge_{f \in F \setminus N} \neg f\}$. The formulas $\mathbb{W}_3$ and $\mathbb{R}_3$ encoding $\mathcal{W}_3$ and $\mathcal{R}_3$, respectively, are the following:

$$\mathbb{W}_3 = \mathbb{W}_1 \wedge \bigwedge_{f \in F} (\neg a_f \rightarrow \neg f) \quad (5)$$

$$\mathbb{R}_3 = \mathbb{W}_3 \wedge \mathbb{R}_1 \quad (6)$$

Proposition 8. For every valuation π of A, $\pi \in M_{\mathbb{W}_3 \downarrow A}$ iff $AF(\pi) \in \mathcal{W}_3$. Consequently, $|M_{\mathbb{W}_3 \downarrow A}| = |\mathcal{W}_3|$.

Proof. $\Rightarrow$: $\mathbb{W}_3$ subsumes $\mathbb{W}_1$, thus for every model π' of $\mathbb{W}_3$ such that $\pi = \pi'_{\downarrow A}$ it holds that $\pi' \models AF(\pi)$ (Proposition 2). Furthermore, by the definition of AF, $f \notin AF(\pi)$ iff $\pi \models \neg a_f$ (i.e., $\pi(a_f) = 0$). Thus, the clauses $\bigwedge_{f \in F} (\neg a_f \rightarrow \neg f)$ of $\mathbb{W}_3$ ensure that for all $f \in F \setminus AF(\pi)$ it holds that $\pi' \not\models f$. Hence, π' is the model ϕ from the definition of $\mathcal{W}_3$.
$\Leftarrow$: Given $N \in \mathcal{W}_3$ and a model ϕ of $N \wedge \bigwedge_{f \in F \setminus N} \neg f$ (by the definition of $\mathcal{W}_3$), we build a valuation π' of $\mathbb{W}_3$ as $\pi'(x) = \phi(x)$ if $x \in \mathit{Vars}(F)$, $\pi'(a_f) = 0$ if $f \notin N$, and $\pi'(a_f) = 1$ if $f \in N$. Similarly as in the proof of Proposition 2, observe that $AF(\pi'_{\downarrow A}) = N$ and that $\pi' \models \mathbb{W}_3$, thus $\pi = \pi'_{\downarrow A} \in M_{\mathbb{W}_3 \downarrow A}$. □

Proposition 9. For every valuation π of A, $\pi \in M_{\mathbb{R}_3 \downarrow A}$ iff $AF(\pi) \in \mathcal{R}_3$. Consequently, $|M_{\mathbb{R}_3 \downarrow A}| = |\mathcal{R}_3|$.

Proof. $M_{\mathbb{R}_3\downarrow A} = M_{(\mathbb{W}_3\wedge\mathbb{R}_1)\downarrow A} = M_{\mathbb{W}_3\downarrow A} \cap M_{\mathbb{R}_1\downarrow A} = \{\pi \,|\, AF(\pi) \in \mathcal{W}_3\} \cap \{\pi \,|\, AF(\pi) \in \mathcal{R}_1\} = \{\pi \,|\, AF(\pi) \in \mathcal{W}_3 \cap \mathcal{R}_1\} = \{\pi \,|\, AF(\pi) \in \mathcal{R}_3\}$. The other direction holds since AF is a one-to-one mapping (bijection). □

Wrapper $\mathcal{W}_4$ Our next wrapper, $\mathcal{W}_4$, is based on another property of MSSes. Intuitively, assume an MSS N of F and a model ϕ of $Vars(F)$. We observe that for every clause $f \in F \setminus N$ and every literal l of the clause, there is a clause $g \in N$ that *forces* l to be falsified. If there would be l that is not forced to be falsified, then the model ϕ can be relaxed to a model ϕ' that would satisfy $N \cup \{f\}$ (which is not possible since N is a MSS). Formally:

Proposition 10. Let N be an MSS of F and ϕ a valuation of $Vars(F)$. If $\phi \models N$, then $\phi \models \bigwedge_{f\in F\setminus N} \mathbb{P}$, where $\mathbb{P} = \bigwedge_{l\in f} \bigvee_{g\in\{g\in N \,|\, \neg l\in g\}} \bigwedge_{k\in g\setminus\{\neg l\}} \neg k$.

Proof. By contradiction, let ϕ be a model of N such that $\phi \not\models \bigwedge_{f\in F\setminus N} \mathbb{P}$. Hence, there exists $f \in F \setminus N$ and $l \in f$ such that $\phi \not\models \bigvee_{g\in\{g\in N \,|\, \neg l\in g\}} \bigwedge_{k\in g\setminus\{\neg l\}} \neg k$. In other words, for every clause $g \in G = \{g \in N \,|\, \neg l \in g\}$ there is a literal $k \in g$, $k \neq \neg l$, with $\phi \models k$. Now, assume that we turn ϕ into a valuation ϕ' by only flipping the assignment to l, i.e., $\phi' \models l$. Clearly, $\phi' \models f$ since $l \in f$. Furthermore, $\phi' \models N \setminus G$ since these clauses do not contain $\neg l$ and thus the change of the assignment to l does not affect them. Finally, $\phi' \models G$ since every $g \in G$ contains a literal k, $k \neq \neg l$, with $\phi \models k$ (and ϕ' agrees with ϕ on k). Hence, $N \cup \{f\}$ is satisfiable, which contradicts that N is a MSS. □

Unfortunately, the proposition reasons about all models of a MSS (i.e., a universal property), which is expensive to encode with a propositional formula. Yet, since every MSS has at least a single model, we can relatively cheaply encode a weaker, existential, variant of Proposition 10. We define $\mathcal{W}_4$ as $\mathcal{W}_4 = \{N \in \mathtt{SS}_F \,|\, \exists\phi \in Vals(F).\phi \models N \wedge \bigwedge_{f\in F\setminus N} \mathbb{P}\}$, where $\mathbb{P} = \bigwedge_{l\in f} \bigvee_{g\in\{g\in N \,|\, \neg l\in g\}} \bigwedge_{k\in g\setminus\{\neg l\}} \neg k$. We encode $\mathcal{W}_4$ and its reminder $\mathcal{R}_4$ via $\mathbb{W}_4$ and $\mathbb{R}_4$ as follows:

$$\begin{aligned} \mathbb{W}_4 &= \mathbb{W}_1 \wedge \bigwedge_{f\in F} \neg a_f \implies \mathbb{P}', \;\; where \\ \mathbb{P}' &= \bigwedge_{l\in f} \bigvee_{g\in\{g\in F \,|\, \neg l\in g\}} (a_g \wedge \bigwedge_{k\in g\setminus\{\neg l\}} \neg k) \end{aligned} \tag{7}$$

$$\mathbb{R}_4 = \mathbb{W}_4 \wedge \mathbb{R}_1 \tag{8}$$

Proposition 11. For every valuation π of A, $\pi \in M_{\mathbb{W}_4\downarrow A}$ iff $AF(\pi) \in \mathcal{W}_4$. Consequently, $|M_{\mathbb{W}_4\downarrow A}| = |\mathcal{W}_4|$.

Proof. $\Rightarrow$: Let π' be a model of $\mathbb{W}_4$ such that $\pi = \pi'_{\downarrow A}$. We show that π' and $AF(\pi)$ comply with the conditions on ϕ and N, respectively, from the definition of $\mathcal{W}_4$. $\mathbb{W}_4$ subsumes $\mathbb{W}_1$, thus $\pi' \models AF(\pi)$ (Proposition 2). Furthermore, since $\pi' \models \mathbb{W}_4$ and $\pi = \pi'_{\downarrow A}$, it holds that $\pi' \models \mathbb{W}_4[\pi]$. Finally, as $f \in AF(\pi)$ iff $\pi \models a_f$, observe that $\mathbb{W}_4[\pi] \equiv \bigwedge_{f\in F\setminus AF(\pi)} \bigwedge_{l\in f} \bigvee_{g\in\{g\in AF(\pi) \,|\, \neg l\in g\}} \bigwedge_{k\in g\setminus\{\neg l\}} \neg k$ (which is the condition on ϕ).

$\Leftarrow$: Given $N \in \mathcal{W}_4$ and a valuation ϕ such that $\phi \models N \wedge \bigwedge_{f\in F\setminus N} \mathbb{P}$ (as in the definition of $\mathcal{W}_4$) , we build a model π' of $\mathbb{W}_4$ same as we did in the proof of Proposition 2, i.e., $\pi'(x) = \phi(x)$ if $x \in Vars(F)$, $\pi'(a_f) = 0$ if $f \notin N$, and $\pi'(a_f) = 1$ if $f \in N$. Like in the proof of Proposition 2, it holds that $AF(\pi'_{\downarrow A}) = N$ and $\pi' \models \mathbb{W}_1$. As for the rest of $\mathbb{W}_4$, it generally holds that $\pi' \models (\bigwedge_{f\in F} \neg a_f \implies \mathbb{P}')$ iff $\pi' \models (\bigwedge_{f\in F} \neg a_f \implies \mathbb{P}')[\pi'_{\downarrow A}]$. Furthermore, $(\bigwedge_{f\in F} \neg a_f \implies \mathbb{P}')[\pi'_{\downarrow A}] \equiv \bigwedge_{f\in F\setminus N} \mathbb{P}$, since $\pi'_{\downarrow A} \models a_f$ iff $f \in N$. Finally, $\phi \models \bigwedge_{f\in F\setminus N} \mathbb{P}$ and π' agrees with ϕ on $Vars(\bigwedge_{f\in F\setminus N} \mathbb{P})$, hence $\pi' \models \bigwedge_{f\in F\setminus N} \mathbb{P}$. □

Proposition 12. For every valuation π of A, $\pi \in M_{\mathbb{R}_4\downarrow A}$ iff $AF(\pi) \in \mathcal{R}_4$. Consequently, $|M_{\mathbb{R}_4\downarrow A}| = |\mathcal{R}_4|$.

Proof. $M_{\mathbb{R}_4\downarrow A} = M_{(\mathbb{W}_4\wedge\mathbb{R}_1)\downarrow A} = M_{\mathbb{W}_4\downarrow A} \cap M_{\mathbb{R}_1\downarrow A} = \{\pi \,|\, AF(\pi) \in \mathcal{W}_4\} \cap \{\pi \,|\, AF(\pi) \in \mathcal{R}_1\} = \{\pi \,|\, AF(\pi) \in \mathcal{W}_4 \cap \mathcal{R}_1\} = \{\pi \,|\, AF(\pi) \in \mathcal{R}_4\}$. The other direction holds since AF is a one-to-one mapping (bijection). □

4.3 Combining The Wrappers

Proposition 13. For every two wrappers $\mathcal{W}_i, \mathcal{W}_j \in \{\mathcal{W}_1, \ldots, \mathcal{W}_4\}$ and their remainders $\mathcal{R}_i, \mathcal{R}_j$, it holds:

1. $\mathcal{W}_i \cap \mathcal{W}_j$ is a wrapper, and $\mathcal{R}_i \cap \mathcal{R}_j$ is its remainder.
2. For every valuation π of A, $\pi \in M_{(\mathbb{W}_i\wedge\mathbb{W}_j)\downarrow A}$ iff $AF(\pi) \in \mathcal{W}_i \cap \mathcal{W}_j$. Consequently, $|M_{(\mathbb{W}_i\wedge\mathbb{W}_j)\downarrow A}| = |\mathcal{W}_i \cap \mathcal{W}_j|$.
3. For every valuation π of A, $\pi \in M_{(\mathbb{R}_i\wedge\mathbb{R}_j)\downarrow A}$ iff $AF(\pi) \in \mathcal{R}_i \cap \mathcal{R}_j$. Consequently, $|M_{(\mathbb{R}_i\wedge\mathbb{R}_j)\downarrow A}| = |\mathcal{R}_i \cap \mathcal{R}_j|$.

Proof.

1. By Definition 3, a set $\mathcal{W}$ is a wrapper iff $\mathtt{MSS}_F \subseteq \mathcal{W} \subseteq \mathtt{SS}_F$, and the remainder of $\mathcal{W}$ is $\mathcal{R} = \mathcal{W} \cap \mathtt{nonMSS}_F$. $\mathcal{W}_i$ and $\mathcal{W}_j$ are wrappers, thus $\mathtt{MSS}_F \subseteq \mathcal{W}_i, \mathcal{W}_j \subseteq \mathtt{SS}_F$, and hence $\mathtt{MSS}_F \subseteq \mathcal{W}_i \cap \mathcal{W}_j \subseteq \mathtt{SS}_F$. Furthermore, $\mathcal{R}_i = \mathcal{W}_i \cap \mathtt{nonMSS}_F$ and $\mathcal{R}_j = \mathcal{W}_j \cap \mathtt{nonMSS}_F$, hence $\mathcal{R}_i \cap \mathcal{R}_j = \mathcal{W}_i \cap \mathcal{W}_j \cap \mathtt{nonMSS}_F$.
2. By Propositions 2, 4, 8 and 11, $M_{(\mathbb{W}_i\wedge\mathbb{W}_j)\downarrow A} = M_{\mathbb{W}_i\downarrow A} \cap M_{\mathbb{W}_j\downarrow A} = \{\pi \,|\, AF(\pi) \in \mathcal{W}_i\} \cap \{\pi \,|\, AF(\pi) \in \mathcal{W}_j\} = \{\pi \,|\, AF(\pi) \in \mathcal{W}_i \cap \mathcal{W}_j\}$. The other direction holds since AF is a one-to-one mapping (bijection).
3. By Propositions 3, 5, 9 and 12, $M_{(\mathbb{R}_i\wedge\mathbb{R}_j)\downarrow A} = M_{\mathbb{R}_i\downarrow A} \cap M_{\mathbb{R}_j\downarrow A} = \{\pi \,|\, AF(\pi) \in \mathcal{R}_i\} \cap \{\pi \,|\, AF(\pi) \in \mathcal{R}_j\} = \{\pi \,|\, AF(\pi) \in \mathcal{R}_i \cap \mathcal{R}_j\}$. The other direction holds since AF is a one-to-one mapping (bijection).

□

Note that all the formulas $\mathbb{W}_2$, $\mathbb{W}_3$ and $\mathbb{W}_4$ use as a subformula $\mathbb{W}_1$. Similarly, all the formulas $\mathbb{R}_2$, $\mathbb{R}_3$ and $\mathbb{R}_4$ use as a subformula $\mathbb{R}_1$. Thus, if we combine two wrappers, we duplicate some clauses in the formulas. In our implementation, we first remove all duplicated clauses from a formula before we pass it to a model counting tool. This simplification is sound as it does not reduce the number of models of the formula.

On Choice of Projected Model Counting

We remark on the choice of reduction of MSS counting to projected model counting. One might wonder whether we could have reduced to the classical problem of model counting. In this context, note that the classical model counting is #P-complete and checking whether an assignment satisfies a CNF formula is in P. In contrast, checking whether a given set of clauses is an MSS is in D^P, and therefore, it is expected to rely on a problem that is perhaps harder than classical model counting from the complexity perspective. Therefore, projected model counting, which is in #NP-hard, is a *good* choice given its hardness and the recent development of efficient techniques.

5 Experimental Evaluation

We have implemented our approach for solving the #MSS problem in a python-based tool. To count the number of projected models of the wrappers, we use the model counter GANAK (Sharma et al. 2019). Furthermore, we employ the MaxSAT solver UWrMaxSat (Piotrów 2019) as a backend while computing the under-approximation I of the intersection of MSSes in the case of the wrapper $\mathcal{W}_2$. Our tool is publicly available at:

https://github.com/jar-ben/MSSCounting

In this section, we experimentally compare the four wrappers, $\mathcal{W}_1, \ldots, \mathcal{W}_4$, and their combinations for the task of determining the MSS count of a given formula. Note that the wrapper $\mathcal{W}_1$ is by the definition subsumed by the remaining three wrappers. Therefore, there are only 8 possible combined wrappers (according to Proposition 13) that make sense: W1 $= \mathcal{W}_1$, W2 $= \mathcal{W}_2$, W3 $= \mathcal{W}_3$, W4 $= \mathcal{W}_4$, W23 $= \mathcal{W}_2 \cap \mathcal{W}_3$, W24 $= \mathcal{W}_2 \cap \mathcal{W}_4$, W34 $= \mathcal{W}_3 \cap \mathcal{W}_4$, W234 $= \mathcal{W}_2 \cap \mathcal{W}_3 \cap \mathcal{W}_4$. At first glance, the wrapper W234 that combines all the base wrappers should be the most suitable one since it provides the most accurate description of MSSes. However, the Boolean formula that describes this wrapper is also the largest one in the number of clauses, and thus it might be hard to deal with for the model counter. Therefore, it makes sense to evaluate all the 8 combinations. Moreover, we compare our wrapper-based approach for counting MSSes with the contemporary MSS counting approach: complete MSS enumeration via an MSS enumeration tool. In particular, we evaluate two contemporary MSS enumeration tools: FLINT (Narodytska et al. 2018)[1], and RIME (Bendík and Černá 2020)[2]. Thus, in total, we compare ten tools (RIME, FLINT, and our approach using one of the eight wrappers).

We use three comparison criteria: 1) the number of benchmarks for which the tools provide the MSS count, 2) the time to provide the MSS count, and 3) the scalability of the tools w.r.t. the MSS count.

We used a collection of 1200 Boolean CNF formulas that were recently used in prior MUS literature (Liu and Luo 2018; Luo and Liu 2019)[3]. The benchmarks contain from 100 to 1000 clauses, use from 50 to 996 variables, and have from 2 to at least 4.74×10^{12} MSSes (the highest MSS count revealed in our evaluation).

All experiments were run using a time limit of 3600 seconds (1 hour) and computed on an AMD EPYC 7371 16-Core Processor, 1 TB memory machine running Debian Linux 4.19.67-2.

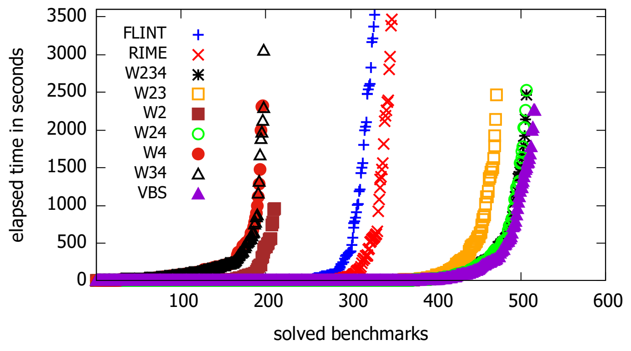

Figure 2: The number of finished benchmarks in time.

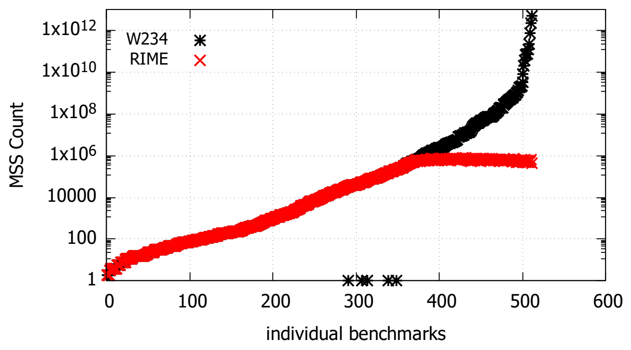

Figure 3: Scalability w.r.t. the MSS count.

5.1 Number of Completed Benchmarks

We now examine the number of benchmarks for which the evaluated tools determined the MSS count; we simply say that the tools *solved* the benchmarks. Only 515 of the 1200 benchmarks were solved by at least one of the tools. Furthermore, 353 benchmarks were solved either by FLINT or by RIME, and 510 benchmarks were solved using one of our wrapper-based tools.

The cactus plot in Figure 2 shows for each tool the number of solved benchmarks and the time to solve the benchmarks. In particular, a point with coordinates $[x, y]$ means that there are x benchmarks for which the corresponding tool provided the MSS count within the first y seconds of the computation. There are only 327 and 347 benchmarks where FLINT and RIME provided the MSS count, respectively. As for our approach, the best result was achieved by

[1]The implementation of FLINT was kindly provided to us by its author, Nina Narodytska.

[2]https://github.com/jar-ben/rime

[3]https://github.com/luojie-sklsde/MUS_Random_Benchmarks

	FLINT	RIME	W234	W24	W23
FLINT	—	6;26	2;181	2;181	2;146
RIME	26;6	—	5;164	5;164	5;129
W234	181;2	164;5	—	4;4	35;0
W24	181;2	164;5	4;4	—	38;3
W23	146;2	129;5	0;35	3;38	—

Table 1: Number of benchmarks where a tool solved more;fewer benchmarks than the other tools.

the wrappers W234 and W24 which both solved 506 benchmarks, i.e., there is an incredible improvement of 46 percent over the best MSS enumerator RIME. W23 solved 471 benchmarks, which is also a solid result. On the other hand, W2, W4, and W34 solved only 209, 195, and 197 benchmarks, respectively. W1 and W3 did not solve even a single benchmark. We also show in the plot the result obtained by the *virtual best system* (VBS), i.e., for each benchmark we consider the fastest of the evaluated tools. Note that W234 and W24 performed almost as good as the VBS.

There are 4 benchmarks that were solved by W234, but not by W24, and vice versa. Table 1 pair-wise compares FLINT, RIME, and the three best wrappers, W234, W24, and W23, w.r.t. this criterion. Each cell contains two numbers, k;l, expressing that there are k benchmarks that were solved by the tool labeling the row but not by the tool labeling the column, and vice versa for l. There are only 2 and 5 benchmarks that were solved by FLINT and RIME, respectively, and that were not solved by any of our wrappers.

5.2 Scalability W.R.T. the MSS Count

An MSS enumerator (e.g., FLINT or RIME) has to explicitly enumerate all MSSes to obtain the MSS count. Consequently, if the MSS count is large, the complete enumeration naturally becomes practically intractable (w.r.t. a reasonable time limit). On the other hand, our approach reduces the MSS counting to model counting, and it is often the case that a model counter needs to explicitly identify only a fraction of the models. Consequently, our approach should hypothetically scale much better w.r.t. the MSS count.

To prove our hypothesis, we compare the best MSS enumeration tool, RIME, with our best wrapper, W234. The plot in Figure 3 shows on the x-axis the benchmarks that were completed by at least one of the two tools and on the y-axis the MSS count of the benchmarks. The benchmarks are sorted by the MSS count. In the case of RIME, the points in the plot show the number of enumerated MSSes within the given time limit, i.e., it is either the exact MSS count or its under-approximation. RIME was able to solve only benchmarks with at most 10^6 MSSes. On the other hand, W234 solved even benchmarks that contain 10^{12} MSSes, i.e., it scales much better. Note that we show in the plot also the 5 benchmarks that were solved by RIME but not by W234; these are illustrated as the 5 points on the x-axis.

6 Conclusion

Motivated by the progress in model counting, we initiate the study of counting the number of MSSes of a given formula. Our novel algorithmic framework relies on the notions of wrappers and their corresponding remainders. We show that wrappers and remainders compose, and the computation of the sizes of wrappers and remainders reduces to the projected model counting. The availability of an efficient projected model counter, GANAK, allowed our MSS counting approach to scale, in terms of the MSS count, significantly better than alternative approaches based on the MSS enumeration.

As for the future work, we would like to address also a better scaling of our approach w.r.t. the number of clauses in the input instance. Whereas we are currently able to handle instances with hundreds of clauses, instances with high thousands or even millions of clauses are still out of our reach. In this context, a promising challenge would be to handle the widely used dataset of 300 CNF formulas from the MUS track of the SAT Competition 2011. A vast majority of benchmarks from this set is not tractable for contemporary MSS enumeration tools due to a large number of MSSes, and it is also not tractable for our approach owing to a large number of clauses in the benchmarks, which in turn leads to an increase in the number of variables for projected model counting queries. An interesting direction to address this scalability challenge is to investigate whether a component caching-based scheme operating natively over the space of MSSes, i.e., avoiding the reduction to model counting, can lead to a better runtime efficiency. Another line of future work is to evaluate other contemporary projected model counting tools such as nestHDB (Hecher, Thier, and Woltran 2020) or projMC (Lagniez and Marquis 2019), and to employ preprocessing techniques such as (Manthey 2012) or (Lagniez, Lonca, and Marquis 2020). Finally, we plan to examine an extension of our MSS counting approach to other constraint domains where MSSes find an application, e.g., F can be a set of LTL (Barnat et al. 2016; Bendík 2017) or SMT (Guthmann, Strichman, and Trostanetski 2016) formulas.

Acknowledgements

This work was supported in part by the National Research Foundation Singapore under its NRF Fellowship Programme [NRF-NRFFAI1-2019-0004] and the AI Singapore Programme [AISG-RP-2018-005], NUS ODPRT Grant [R-252-000-685-13].

References

Aziz, R. A.; Chu, G.; Muise, C.; and Stuckey, P. 2015. exists SAT: Projected Model Counting. In *International Conference on Theory and Applications of Satisfiability Testing*, 121–137. Springer.

Bacchus, F.; and Katsirelos, G. 2015. Using Minimal Correction Sets to More Efficiently Compute Minimal Unsatisfiable Sets. In *CAV (2)*, volume 9207 of *LNCS*, 70–86. Springer.

Bacchus, F.; and Katsirelos, G. 2016. Finding a Collection of MUSes Incrementally. In *CPAIOR*, volume 9676 of *LNCS*, 35–44. Springer.

Bailey, J.; and Stuckey, P. J. 2005. Discovery of minimal unsatisfiable subsets of constraints using hitting set dualization. In *PADL*, 174–186. Springer.

Baluta, T.; Shen, S.; Shinde, S.; Meel, K. S.; and Saxena, P. 2019. Quantitative verification of neural networks and its security applications. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, 1249–1264.

Barnat, J.; Bauch, P.; Beneš, N.; Brim, L.; Beran, J.; and Kratochvíla, T. 2016. Analysing sanity of requirements for avionics systems. *FAoC* 1–19.

Bayardo Jr, R. J.; and Pehoushek, J. D. 2000. Counting models using connected components. In *AAAI/IAAI*, 157–162.

Belov, A.; Lynce, I.; and Marques-Silva, J. 2012. Towards efficient MUS extraction. *AI Commun.* 25(2): 97–116.

Bendík, J. 2017. Consistency checking in requirements analysis. In *ISSTA*, 408–411. ACM.

Bendík, J.; Beneš, N.; Černá, I.; and Barnat, J. 2016. Tunable Online MUS/MSS Enumeration. In *FSTTCS*, volume 65 of *LIPIcs*, 50:1–50:13. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik.

Bendík, J.; and Černá, I. 2020a. MUST: Minimal Unsatisfiable Subsets Enumeration Tool. In *TACAS (1)*, volume 12078 of *LNCS*, 135–152. Springer.

Bendík, J.; and Černá, I. 2020b. Replication-Guided Enumeration of Minimal Unsatisfiable Subsets. In *CP*, volume 12333 of *LNCS*, 37–54. Springer.

Bendík, J.; and Černá, I. 2020. Rotation Based MSS/MCS Enumeration. In *LPAR*, volume 73 of *EPiC Series in Computing*, 120–137. EasyChair.

Bendík, J.; Černá, I.; and Beneš, N. 2018. Recursive Online Enumeration of All Minimal Unsatisfiable Subsets. In *ATVA*, volume 11138 of *LNCS*, 143–159. Springer.

Bendík, J.; and Meel, K. S. 2020. Approximate Counting of Minimal Unsatisfiable Subsets. In *CAV (1)*, volume 12224 of *LNCS*, 439–462. Springer.

Besnard, P.; Grégoire, É.; and Lagniez, J.-M. J. 2015. On computing maximal subsets of clauses that must be satisfiable with possibly mutually-contradictory assumptive contexts. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*.

Biondi, F.; Enescu, M. A.; Heuser, A.; Legay, A.; Meel, K. S.; and Quilbeuf, J. 2018. Scalable approximation of quantitative information flow in programs. In *International Conference on Verification, Model Checking, and Abstract Interpretation*, 71–93. Springer.

Birnbaum, E.; and Lozinskii, E. L. 1999. The good old Davis-Putnam procedure helps counting models. *Journal of Artificial Intelligence Research* 10: 457–477.

Chakraborty, S.; Fremont, D. J.; Meel, K. S.; Seshia, S. A.; and Vardi, M. Y. 2014. Distribution-Aware Sampling and Weighted Model Counting for SAT. In *Proc. of AAAI*, 1722–1730.

Chakraborty, S.; Meel, K. S.; and Vardi, M. Y. 2016. Algorithmic Improvements in Approximate Counting for Probabilistic Inference: From Linear to Logarithmic SAT Calls. In *Proc. of IJCAI*.

Darwiche, A. 2004. New Advances in Compiling CNF into Decomposable Negation Normal Form. In *ECAI*, 328–332. IOS Press.

Darwiche, A. 2011. SDD: A New Canonical Representation of Propositional Knowledge Bases. In *IJCAI*, 819–826. IJCAI/AAAI.

de Kleer, J.; and Williams, B. C. 1987. Diagnosing Multiple Faults. *Artif. Intell.* 32(1): 97–130.

Durand, A.; Hermann, M.; and Kolaitis, P. G. 2005. Subtractive reductions and complete problems for counting complexity classes. *Theoretical Computer Science* 340(3): 496–513.

Guthmann, O.; Strichman, O.; and Trostanetski, A. 2016. Minimal unsatisfiable core extraction for SMT. In *FMCAD*, 57–64. IEEE.

Hecher, M.; Thier, P.; and Woltran, S. 2020. Taming High Treewidth with Abstraction, Nested Dynamic Programming, and Database Technology. In *SAT*, volume 12178 of *LNCS*, 343–360. Springer.

Hunter, A.; and Konieczny, S. 2008. Measuring Inconsistency through Minimal Inconsistent Sets. In *KR*, 358–366. AAAI Press.

Kleine Büning, H.; and Kullmann, O. 2009. Minimal Unsatisfiability and Autarkies. In *Handbook of Satisfiability*, volume 185 of *FAIA*, 339–401. IOS Press.

Kullmann, O. 2000a. An Application of Matroid Theory to the SAT Problem. In *Computational Complexity Conference*, 116. IEEE Computer Society.

Kullmann, O. 2000b. Investigations on autark assignments. *Discrete Applied Mathematics* 107(1-3): 99–137.

Lagniez, J.; Lonca, E.; and Marquis, P. 2020. Definability for model counting. *Artif. Intell.* 281: 103229.

Lagniez, J.; and Marquis, P. 2017. An Improved Decision-DNNF Compiler. In *IJCAI*, 667–673. ijcai.org.

Lagniez, J.-M.; and Marquis, P. 2019. A recursive algorithm for projected model counting. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, 1536–1543.

Liffiton, M. H.; Previti, A.; Malik, A.; and Marques-Silva, J. 2016. Fast, flexible MUS enumeration. *Constraints* 21(2): 223–250.

Liffiton, M. H.; and Sakallah, K. A. 2008. Algorithms for computing minimal unsatisfiable subsets of constraints. *JAR* 40(1): 1–33.

Liu, S.; and Luo, J. 2018. FMUS2: An Efficient Algorithm to Compute Minimal Unsatisfiable Subsets. In *AISC*, volume 11110 of *LNCS*, 104–118. Springer.

Luo, J.; and Liu, S. 2019. Accelerating MUS enumeration by inconsistency graph partitioning. *Science China Information Sciences* 62(11): 212104.

Manthey, N. 2012. Coprocessor 2.0 - A Flexible CNF Simplifier - (Tool Presentation). In *SAT*, volume 7317 of *LNCS*, 436–441. Springer.

Marques-Silva, J.; Heras, F.; Janota, M.; Previti, A.; and Belov, A. 2013. On Computing Minimal Correction Subsets. In *IJCAI*, 615–622. IJCAI/AAAI.

Marques-Silva, J.; Ignatiev, A.; Morgado, A.; Manquinho, V. M.; and Lynce, I. 2014. Efficient Autarkies. In *ECAI*, volume 263 of *FAIA*, 603–608. IOS Press.

Mencía, C.; Kullmann, O.; Ignatiev, A.; and Marques-Silva, J. 2019. On Computing the Union of MUSes. In *SAT*, volume 11628 of *LNCS*, 211–221. Springer.

Meseguer, P.; Bouhmala, N.; Bouzoubaa, T.; Irgens, M.; and Sánchez-Fibla, M. 2003. Current Approaches for Solving Over-Constrained Problems. *Constraints An Int. J.* 8(1): 9–39.

Möhle, S.; and Biere, A. 2018. Dualizing projected model counting. In *2018 IEEE 30th International Conference on Tools with Artificial Intelligence (ICTAI)*, 702–709. IEEE.

Monien, B.; and Speckenmeyer, E. 1985. Solving satisfiability in less than 2^n steps. *Discrete Applied Mathematics* 10(3): 287–295.

Muise, C.; McIlraith, S. A.; Beck, J. C.; and Hsu, E. I. 2012. D sharp: fast d-DNNF compilation with sharpSAT. In *Canadian Conference on Artificial Intelligence*, 356–361. Springer.

Narodytska, N.; Bjørner, N.; Marinescu, M.; and Sagiv, M. 2018. Core-Guided Minimal Correction Set and Core Enumeration. In *IJCAI*, 1353–1361. ijcai.org.

Piotrów, M. 2019. UWrMaxSat-a new MiniSat+-based Solver in MaxSAT Evaluation 2019. *MaxSAT Evaluation 2019* 11.

Previti, A.; Mencía, C.; Järvisalo, M.; and Marques-Silva, J. 2018. Premise Set Caching for Enumerating Minimal Correction Subsets. In *AAAI*, 6633–6640. AAAI Press.

Reiter, R. 1987. A Theory of Diagnosis from First Principles. *Artif. Intell.* 32(1): 57–95.

Sang, T.; Bacchus, F.; Beame, P.; Kautz, H. A.; and Pitassi, T. 2004. Combining Component Caching and Clause Learning for Effective Model Counting. *SAT* 4: 7th.

Sang, T.; Beame, P.; and Kautz, H. A. 2005. Performing Bayesian inference by weighted model counting. In *AAAI*, volume 5, 475–481.

Sashittal, P.; and El-Kebir, M. 2020. Sampling and summarizing transmission trees with multi-strain infections. *Bioinformatics* 36(Supplement_1): i362–i370.

Sharma, S.; Roy, S.; Soos, M.; and Meel, K. S. 2019. GANAK: A Scalable Probabilistic Exact Model Counter. In *IJCAI*, 1169–1176. ijcai.org.

Sperner, E. 1928. Ein satz über untermengen einer endlichen menge. *Mathematische Zeitschrift* 27(1): 544–548.

Stern, R. T.; Kalech, M.; Feldman, A.; and Provan, G. M. 2012. Exploring the Duality in Conflict-Directed Model-Based Diagnosis. In *AAAI*. AAAI Press.

Thimm, M. 2018. On the evaluation of inconsistency measures. *Measuring Inconsistency in Information* 73.

Thurley, M. 2006. sharpSAT–counting models with advanced component caching and implicit BCP. In *International Conference on Theory and Applications of Satisfiability Testing*, 424–429. Springer.

Valiant, L. G. 1979. The Complexity of Enumeration and Reliability Problems. *SIAM J. Comput.* 8(3): 410–421.