SAT-Based Formula Simplification

Alexey Ignatiev[1], Alessandro Previti[2], and Joao Marques-Silva[1,2]

[1] INESC-ID, IST, University of Lisbon, Portugal
[2] CASL, University College Dublin, Ireland

Abstract. The problem of propositional formula minimization can be traced to the mid of the last century, to the seminal work of Quine and McCluskey, with a large body of work ensuing from this seminal work. Given a set of implicants (or implicates) of a formula, the goal for minimization is to find a smallest set of prime implicants (or implicates) equivalent to the original formula. This paper considers the more general problem of computing a smallest prime representation of a non-clausal propositional formula, which we refer to as formula simplification. Moreover, the paper proposes a novel, entirely SAT-based, approach for the formula simplification problem. The original problem addressed by the Quine-McCluskey procedure can thus be viewed as a special case of the problem addressed in this paper. Experimental results, obtained on well-known representative problem instances, demonstrate that a SAT-based approach for formula simplification is a viable alternative to existing implementations of the Quine-McCluskey procedure.

1 Introduction

The Quine-McCluskey [47,48,36] procedure for the minimization of clausal formulae (i.e. formulae either represented in Conjunctive Normal Form (CNF) or Disjunctive Normal Form (DNF)) is widely known, being a standard topic in a number of textbooks (e.g. [22]), with a number of publicly available implementations. This problem is referred to as *formula minimization* in this paper. Formula minimization finds a wide range of practical applications [49,55,46,8,17,54,28,58,19,14,11,5], ranging from security to biology. A typical implementation of Quine-McCluskey starts by computing all the prime implicates (or implicants) of a CNF (or DNF) formula, and then implements a set covering step, where a minimum number of prime implicates (implicants) is selected that is equivalent to the original function. A more general scenario is when the original formula is non-clausal. Clearly, one can still generate all the implicates (or implicants) of the formula, then generate all the prime implicates (or prime implicants), and then execute the set covering step. However, in practice the number of implicates may be much larger than the number of prime implicates. In contrast to the more restricted problem, this problem is referred to as *formula simplification* in this paper. Moreover, regarding the existing implementations of Quine-McCluskey, these are not only limited to clausal formula minimization but also usually restricted to a small number of variables. The latter is also the case for other formula simplification alternatives based on Binary Decision Diagrams (BDDs) [10].

* This work is partially supported by SFI PI grant BEACON (09/IN.1/ I2618), FCT grant POLARIS (PTDC/EIA-CCO/123051/2010) and national funds through Fundação para a Ciência e a Tecnologia (FCT) with reference UID/CEC/50021/2013.

This paper develops novel approaches for formula simplification as well as formula minimization, both of which are entirely SAT-based[1]. The proposed approaches exploit recent work on computing prime implicates (and implicants) with SAT solvers [45,27], but also recent work on solving MaxSAT [40] and on computing smallest minimal unsatisfiable subformulae (SMUS) [31,23,24,26]. For the formula minimization problem, the main technical contribution is a new way to compute the prime implicates (or implicants) of the formula. For the formula simplification problem, the main technical contribution is the integration of prime enumeration with smallest MUS extraction.

Throughout the paper, and similarly to the most common description of the Quine-McCluskey procedure, the focus will be to compute the prime implicants of a propositional formula (possibly represented in DNF) and then to select a minimum size set of prime implicants equivalent to the original formula. However, the algorithms described in the paper also apply when computing and minimizing the set of prime implicates, possibly starting from a CNF representation.

The paper is organized as follows. Section 2 introduces the notation and definitions used throughout the paper. Section 3 describes the novel approach to formula simplification proposed in the paper. Preliminary experimental results are analyzed in Section 4. Finally, Section 5 concludes the paper.

2 Preliminaries

Definitions standard in propositional satisfiability (SAT) and maximum satisfiability (MaxSAT) solving are assumed [4]. In what follows, $\mathcal{F}$ denotes an arbitrary propositional formula. A term t is a conjunction of literals and a clause c is a disjunction of literals, while a literal l is either a Boolean variable or its negation. Whenever convenient, terms and clauses are treated as sets of literals. A formula is said to be in *conjunctive* or *disjunctive normal form* (CNF or DNF, respectively) if it is a conjunction of clauses or disjunction of terms, respectively. Set theory notation will be also used with respect to CNF and DNF formulae when necessary. Moreover, the term *clausal* will be used to denote formulae represented as sets of sets of literals, i.e. either in CNF or DNF.

Definition 1. *A term $\mathcal{I}_n$ is called an implicant of $\mathcal{F}$ if $\mathcal{I}_n \models \mathcal{F}$. An implicant $\mathcal{I}_n$ of $\mathcal{F}$ is called **prime** if any subset $\mathcal{I}'_n \subsetneq \mathcal{I}_n$ is not an implicant of $\mathcal{F}$.*

Definition 2. *A clause $\mathcal{I}_e$ is called an implicate of $\mathcal{F}$ if $\mathcal{F} \models \mathcal{I}_e$. An implicate $\mathcal{I}_e$ of $\mathcal{F}$ is called **prime** if any subset $\mathcal{I}'_e \subsetneq \mathcal{I}_e$ is not an implicate of $\mathcal{F}$.*

The sets of all prime implicants and prime implicates of a Boolean formula $\mathcal{F}$ are denoted by $\mathsf{PI}_n(\mathcal{F})$ and $\mathsf{PI}_e(\mathcal{F})$, respectively. A subset $\mathcal{P}$ of $\mathsf{PI}_n(\mathcal{F})$ (or $\mathsf{PI}_e(\mathcal{F})$) such that $\mathcal{P} \equiv \mathcal{F}$ is said to be a *prime cover* of $\mathcal{F}$. Observe that given $\mathcal{F}$ and a prime implicant $\mathcal{I}_n \models \mathcal{F}$, the clause $\neg\mathcal{I}_n$ is a prime implicate of $\neg\mathcal{F}$, and the other way around. Moreover, a similar connection between $\mathsf{PI}_n(\mathcal{F})$ and $\mathsf{PI}_e(\neg\mathcal{F})$ also holds. Additionally, the concept of an *essential* prime implicant is exploited in the paper. A prime implicant is called essential if it is included in any set of prime implicants covering $\mathcal{F}$. With respect to CNF formulae, the following definitions related to MUSes and MCSes are also used:

[1] Earlier work [52] used SAT as part of the ESPRESSO algorithm [7].

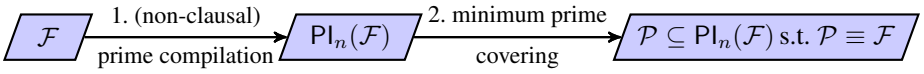


Fig. 1: General steps of the approach

Definition 3. *Given a CNF formula $\mathcal{F}$, a set of clauses $\mathcal{U} \subseteq \mathcal{F}$ is called a minimal unsatisfiable subset (MUS) if $\mathcal{U}$ is unsatisfiable and any subset $\mathcal{U}' \subset \mathcal{U}$ is satisfiable. A minimum size MUS of $\mathcal{F}$ is called a smallest MUS (SMUS).*

Definition 4. *A subset $\mathcal{C}$ of a CNF formula $\mathcal{F}$ is a minimal correction subset (MCS) if $\mathcal{F} \setminus \mathcal{C}$ is satisfiable and $\forall \mathcal{C}' \subseteq \mathcal{C} \wedge \mathcal{C}' \neq \emptyset$, $(\mathcal{F} \setminus \mathcal{C}) \cup \mathcal{C}'$ is unsatisfiable.*

These notions can be extended to the case of *group oriented* CNF formulae [33,41]. A group oriented CNF formula contains groups of clauses instead of single clauses, i.e. $\mathcal{F} = \mathcal{D} \cup \mathcal{G}$, where $\mathcal{G} = \mathcal{G}_1 \cup \ldots \cup \mathcal{G}_k$ is a set of k groups while $\mathcal{D}$ is a *don't care* group. Accordingly, a *group MUS* of $\mathcal{F}$ is a subset of groups $\mathcal{G}' \subseteq \mathcal{G}$ such that formula $\mathcal{D} \cup \bigcup_{G \in \mathcal{G}'} G$ is unsatisfiable and $\forall \mathcal{G}'' \subset \mathcal{G}'$ formula $\mathcal{D} \cup \bigcup_{G \in \mathcal{G}''} G$ is satisfiable.

3 Formula Simplification with SAT

The approach proposed below follows the general steps of the original Quine-McCluskey algorithm [47,48,36] outlined in Figure 1. Given a propositional formula $\mathcal{F}$ in an non-clausal form, it (i) enumerates all prime implicants $\mathsf{PI}_n(\mathcal{F})$ (or prime implicates $\mathsf{PI}_e(\mathcal{F})$); and (ii) computes a minimum size subset $\mathcal{P} \subseteq \mathsf{PI}_n(\mathcal{F})$ (or $\mathcal{P} \subseteq \mathsf{PI}_e(\mathcal{F})$) such that $\mathcal{P} \equiv \mathcal{F}$. Hereinafter, the discussion is conducted with respect to computing a minimum size DNF representation of $\mathcal{F}$ (i.e. using prime implicants of $\mathcal{F}$). However, all of the proposed techniques can be easily adapted for the case of computing a minimum size CNF of $\mathcal{F}$ (i.e. with the use of prime implicates). Indeed, this results from the well-known connection between prime implicants of $\mathcal{F}$ and prime implicates of $\neg\mathcal{F}$ (see Section 2). For this reason and whenever convenient, some particular ideas are explained for implicate-based formula simplification.

3.1 Prime Implicant/Implicate Enumeration

The SAT-based approach being proposed relies on the efficient prime compilation of Boolean formulae. Although (and in contrast to [47,48,36]) the paper is mainly focused on non-clausal Boolean formulae, this section provides a description of the simplified version of the algorithm targeting clausal formulae. The reader is referred to [45] for further details and properties of the general algorithm.

Prime Compilation of Clausal Formulae Although in general the extraction of a prime implicant requires a linear number of calls to a SAT solver, for the case of CNF formulae minimizing a model can be done in polynomial time. The algorithm used in this paper for the extraction of prime implicates is based on the algorithm *primer-b* recently introduced in [45]. When executed on a non-clausal formula, *primer-b* produces the complete set of prime implicates and (as a by-product) a prime implicant cover. At each step, *primer-b* identifies a new partial assignment to be tested. As highlighted in

earlier work [45], when a partial assignment falsifies the formula, then its negation is guaranteed to be a prime implicate. Instead, if it satisfies the formula, the corresponding model has to be reduced to a prime implicant. However, when we deal with CNF formulae, the model can be reduced without employing a SAT solver by means of a procedure running in polynomial time. Suppose that m is a model for a CNF formula $\mathcal{F}$. Then we have to scan all the literals in m one at a time. Let l be the last picked literal. If when setting l to *don't care*, the implicant still satisfies the formula, then literal l is removed. Otherwise, it is a part of the prime implicant under construction. Note that in order to test if a literal is necessary, it is enough to check only the clauses containing it. This can be easily done by using an occurrence list, which for each literal stores the set of clauses where it appears. Additionally, more sophisticated techniques [13] can be also applied for improving the performance of the algorithm.

3.2 Computing a Smallest Prime Cover

This section describes the second phase of the proposed approach, which consists in the following. Given a complete set of prime implicants of a Boolean formula, it computes its subset of the smallest size such that the subset is equivalent to the original formula.

Prime Covering Non-Clausal Formulae Let us assume that for a given non-clausal formula $\mathcal{F}$, the complete set of prime implicants $\mathsf{PI}_n(\mathcal{F})$ is computed as described in Section 3.1. Now one needs to find a minimum size subset $\mathcal{P} \subseteq \mathsf{PI}_n(\mathcal{F})$ such that $\mathcal{P} \equiv \mathcal{F}$. Clearly, by definition of a prime implicant, for any subset $\mathcal{P}$ (and, thus, for the smallest one) the following holds: $\mathcal{P} \models \mathcal{F}$. Therefore, it is enough to check whether $\mathcal{F} \models \mathcal{P}$, which can be done by testing if formula $\neg\mathcal{P} \wedge \mathcal{F}$ is unsatisfiable. Observe that $\mathsf{PI}_n(\mathcal{F}) \equiv \mathcal{F}$ and, thus, $\mathcal{F} \models \mathsf{PI}_n(\mathcal{F})$. Hence, formula

$$\neg\mathsf{PI}_n(\mathcal{F}) \wedge \mathcal{F} \tag{1}$$

is obviously unsatisfiable. This means that finding a minimum size cover $\mathcal{P}$ of $\mathcal{F}$ consists in computing a smallest size group MUS (e.g. see [33,41]) of formula (1) where subformula $\mathcal{F}$ is a *don't care group*, i.e. $\mathcal{F}$ is irrelevant for the size of the solution, and so only clauses $\neg I_n \in \neg\mathsf{PI}_n(\mathcal{F})$ are taken into account. This problem can be solved with an off-the-shelf SMUS extractor (e.g. [34,38,31,23,24,26]).

Note that a smallest size group MUS of formula (1) corresponds to a minimum size prime cover $\mathcal{P}$ of $\mathcal{F}$ with respect to the number of prime implicants in $\mathcal{P}$. However, one might prefer to compute a minimum cover in terms of the *total number of occurrence of literals* in it. For this, a weighted group MUS formula can be considered, i.e. each clause $\neg I_n \in \neg\mathsf{PI}_n(\mathcal{F})$ is associated with a cost equal to $|I_n|$. Now, a smallest cost group MUS of (1) corresponds to a minimum cost prime cover of $\mathcal{F}$.

Observe that essential prime implicants of $\mathcal{F}$ can be identified by group MCS extraction (e.g. see [43,35] and references therein) on the considered formula (1). This is stated in the following proposition.

Proposition 1. *Any unit MCS (i.e. an MCS containing just one clause) of formula* (1) *corresponds to an essential prime implicant of formula* $\mathcal{F}$.

Proof. Due to the minimal hitting set duality between MCSes and MUSes of a (group) CNF formula [50,33], a clause of a unit MCS of the formula is included into any MUS

of the formula. Since, by construction of (1), group unsatisfiable subformulae (hence, MUSes as well) define prime covers of $\mathcal{F}$, unit MCSes of (1) define prime implicants of $\mathcal{F}$ that must be included into *any* prime cover of $\mathcal{F}$. Thus, by definition of essential prime implicants, unit MCSes correspond to essential prime implicants of $\mathcal{F}$. □

Unit MCSes (if any) can be identified with the use of MaxSAT (e.g. [31,23,24,26]). This requires a SAT call for extracting an unsatisfiable core of (1), *relaxing* the corresponding clauses in the core, and enumerating models of the relaxed formula. Each unit MCS is defined by such a model and, thus, requires one SAT call per MCS. Thus, assuming that $\mathcal{F}$ has n essential primes, they can be enumerated with $n + 1$ calls to a SAT oracle. Observe that this approach should be practically more efficient than the well-known alternative of separately checking each prime implicant for essentiality [53,51,22], especially if $|\mathsf{PI}_n(\mathcal{F})|$ is much larger than the number of essential primes.

Moreover, identification of the essential primes can be used for the further simplification of the group SMUS problem. Indeed, since essential prime implicants are included in any cover of $\mathcal{F}$, they can be excluded from $\neg\mathsf{PI}_n(\mathcal{F})$ and added to the don't care group. Let $\mathcal{E}$ denote the set of all essential primes of $\mathcal{F}$, and $\mathcal{Q} = \mathsf{PI}_n(\mathcal{F}) \setminus \mathcal{E}$. Then consider formula $\neg\mathcal{Q} \land (\mathcal{F} \land \mathcal{E})$ where $\mathcal{F} \land \mathcal{E}$ represents the don't care group. For any group SMUS $\mathcal{P}'$ of this formula, the corresponding group SMUS of (1) is $\mathcal{P}' \cup \mathcal{E}$.

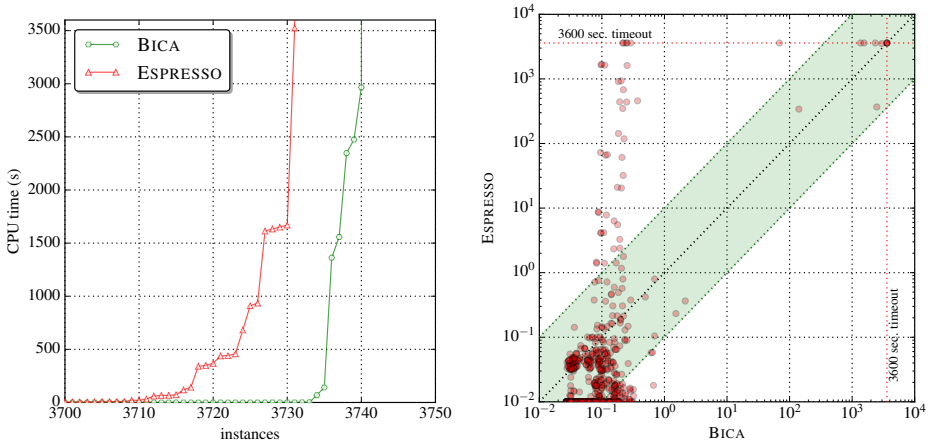
Clausal Formulae Minimization This section briefly explains how one can deal with a particular case of clausal formulae. Recall that given a clausal formula $\mathcal{F}$, the approach being proposed is able to compute the exact minimum size representation of $\mathcal{F}$. Although the general technique described in Section 3.2 can be also applied to clausal formulae, a specialized MaxSAT-based approach to clausal formulae minimization can be proposed, which can be more efficient in practice.

Following the ideas of [47,48,36], one can formulate a set covering problem: given a set of terms $\mathcal{F} = \mathcal{T}_1 \cup \ldots \cup \mathcal{T}_m$ and a complete set of its prime implicants $\mathsf{PI}_n(\mathcal{F})$, one needs to compute a smallest size set of prime implicants $\mathcal{P} \subseteq \mathsf{PI}_n(\mathcal{F})$ such that for each $\mathcal{T}_i \in \mathcal{F}$ there is a prime implicant $\mathcal{I}_j \in \mathcal{P}$ covering term $\mathcal{T}_i$, i.e. $\mathcal{I}_j \subseteq \mathcal{T}_i$. The relation between the set covering problem and MaxSAT was originally put forward in [18,44]. The translation from the set covering problem to MaxSAT is well-known and has been studied elsewhere (e.g. see [56,2]). Note that compared to the general case SMUS-based approach, using this MaxSAT formulation of the problem is preferred for clausal formulae due to a better complexity characterization (decision versions of MaxSAT and SMUS are complete for NP [18,44] and Σ_2^{P} [30,21], respectively).

Approximated Solutions Once the SMUS and MaxSAT formulations of the simplification phase of the approach are introduced, one can immediately notice that various techniques can be applied in order to get approximate solutions of the considered problems. For SMUS, these include MUS extraction (e.g. see [3,42]) and MUS enumeration (see [32]) algorithms. As for MaxSAT, a number of MCS enumeration techniques approximating MaxSAT solutions were proposed in the past (e.g. [43,35,20]).

4 Preliminary Results

This section evaluates the proposed approach to Boolean formula simplification. The experiments were performed in Ubuntu Linux on an Intel Xeon E5-2630 2.60GHz pro-



(a) Cactus plot (b) Scatter plot

Fig. 2: Performance of BICA and ESPRESSO on PLA instances

cessor with 64GByte of memory. The time limit was set to 3600s and the memory limit to 10GByte. The approach proposed above was implemented in a prototype called BICA (***B**oolean **si**mplifier for non-**cla**usal formulae*). The BICA Boolean formula simplifier is written as a Python script, which instruments the flow of the proposed approach and calls the existing binaries both for doing the prime compilation phase and the minimum covering phase. Prime implicate enumeration is done by calling PRIMER [45], while minimum covering is done with the FORQES SMUS extractor [26] for non-clausal formulae, and with the MSCG MaxSAT solver [25,39] if the formulae are clausal. Also note that PRIMER is implemented on top of the MINISAT[2] SAT solver [15] while the underlying SAT solver of MSCG and FORQES is Glucose 3.0[3] [1]. Further details on the experimental evaluation including the chosen benchmark sets are presented below.

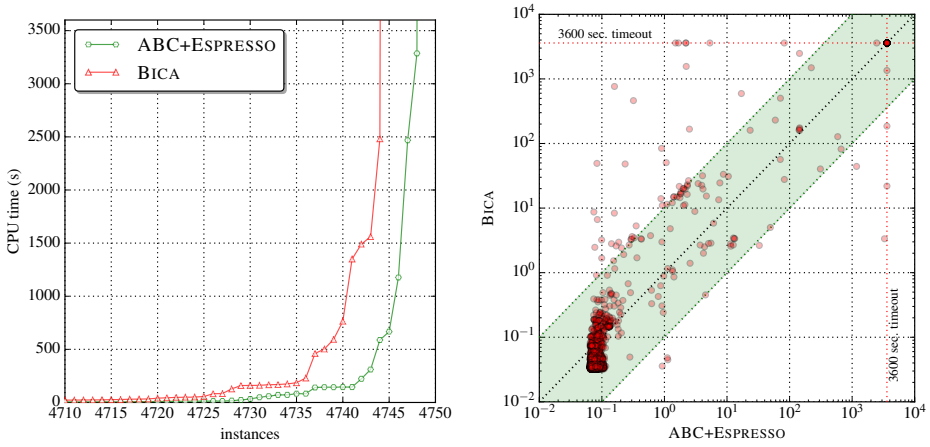
4.1 PLA Benchmarks

In order to assess the efficiency of the new approach applied to clausal Boolean formulae, two sets of PLA circuit benchmark sets were considered. The first set was originally described in [7] and includes 123 easy and 19 hard instances [16]. The second benchmark set called *MCNC91 suite* was proposed in [57] and comprises 41 PLA circuits. Since the approach being proposed currently cannot be applied to multi-output Boolean circuits and in order to compare it with the well-known implementation of the Quine-McCluskey procedure called Espresso [7,16], each of the considered instances was split in the following way. Given a PLA circuit with n inputs and m outputs, m single-output PLA circuits were created, each having n inputs. The total number of resulting PLA circuits constructed this way and considered in the evaluation is 3744.

The new approach was compared to the *exact* version of Espresso [7,16], which is referred to as ESPRESSO and implements the Quine-McCluskey algorithm. Figure 2

[2] https://github.com/niklasso/minisat

[3] http://www.labri.fr/perso/lsimon/glucose



(a) Cactus plot (b) Scatter plot

Fig. 3: Performance of BICA and ESPRESSO on Interpolation instances

shows the performance of ESPRESSO compared to BICA for the considered set of clausal instances. As one can see in Figure 2a, both solvers can minimize most of the circuits. BICA is able to solve 3740 instances (out of 3744), ESPRESSO is not far with 3731 formulae minimized. However, the detailed scatter plot shown in Figure 2b indicates that BICA generally performs better than ESPRESSO (up to 4 orders of magnitude).

4.2 Bi-decomposition Interpolation Benchmarks

The following benchmark set comes from the area of bi-decomposition of a Boolean function (e.g. see [9]). An earlier work on using interpolants for Boolean function decomposition is for example [29], where the function's components are computed through *Craig's interpolation* [12]. Thus, given such interpolants representing the function's components, one can try to simplify them in order to get a simpler decomposed representation of the original Boolean function. The interpolant formulae were generated for the standard ISCAS, ITC, and LGSynth benchmark suites. The total number of the considered interpolant formulae is 4815.

Note that the interpolants are given in a non-clausal form. In this case, one cannot use ESPRESSO directly. First, the formulae need to be translated into a clausal form. For this purpose, the well-known logic synthesis system ABC [6] was used, namely its ability to *collapse* a circuit with the use of BDDs. Figure 3a shows a cactus plot illustrating the performance of both BICA and ABC+ESPRESSO for the considered interpolation benchmarks. Analogously to the PLA benchmarks, both competitors perform quite well being able to solve almost all the instances. BICA simplifies 4744 formulae while ABC+ESPRESSO solves 4748 instances. Figure 3b indicates that there is no clear winner in this case even though ABC+ESPRESSO has some advantage over BICA. A reason for this can be that the CUDD BDD package[4] used in ABC is usually able

[4] http://vlsi.colorado.edu/~fabio/CUDD/

Table 1: Performance of BICA and ABC+ESPRESSO on QG6 instances

	# solved	max. time (s)	min. time (s)	avg. time (s)
BICA	63	3600	0.56	1592.65
ABC+ESPRESSO	0	3600	3600	3600

to *clausify* the considered circuits within a very short time (less than a second). Also, the number of terms reported by CUDD is usually very close to the optimum, which simplifies the Quine-McCluskey procedure performed by ESPRESSO.

4.3 Quasigroup Classification Benchmarks

This set of non-clausal benchmarks called QG6 was proposed in [37] when encoding classification theorems for quasigroups. Out of 256 formulae we chose 83 that are satisfiable. Note that these 83 benchmark instances have either 252 or 360 variables, which is larger than the number of inputs in all circuits considered in Section 4.1 and Section 4.2. Similarly to the interpolation benchmarks, ABC+ESPRESSO was used as an alternative to BICA. However, it was not able to simplify any of these formulae, which is not surprising because these instances are hard for BDDs (this may be caused by the number of variables). (For this reason, no plots are presented for QG6 benchmarks and Table 1 is shown instead). In contrast, BICA is able to simplify 63 (out of 83) formulae.

In summary, the experimental results indicate that the proposed approach is a viable alternative to the existing implementations of the Quine-McCluskey procedure for the case of clausal Boolean formulae. Moreover and as stated in Section 4.3, being focused on non-clausal formulae and based on the state-of-the-art SAT technology, the new approach performs reasonably well for non-clausal formulae with a large number of variables, which can be out of reach for the alternative approaches, e.g. the ones based on BDDs, or ABC and Espresso.

5 Conclusions

This paper develops entirely SAT-based solutions for propositional formula minimization and simplification. In both cases the set of prime implicates (or implicants) is computed using recent work on prime implicate (or implicant) enumeration. For the clausal formula minimization problem, a minimum-size subset of the prime implicates that covers an initial set of implicates is obtained with a set covering approach, which is done with MaxSAT. For non-clausal formula simplification, the problem is more challenging, and the problem is shown to be solved by computing a smallest MUS.

The experimental results are encouraging. For two classes of problem instances, the new approach outperforms a well-known implementation of Quine-McCluskey, whereas for another class of problem instances it loses to the Quine-McCluskey procedure. Future work will investigate settings in which SAT-based formula minimization and simplification can be shown to be the preferred option.

References

1. G. Audemard, J. Lagniez, and L. Simon. Improving Glucose for incremental SAT solving with assumptions: Application to MUS extraction. In *SAT*, pages 309–317, 2013.
2. J. Bautista and J. Pereira. A GRASP algorithm to solve the unicost set covering problem. *Computers & OR*, 34(10):3162–3173, 2007.
3. A. Belov, I. Lynce, and J. Marques-Silva. Towards efficient MUS extraction. *AI Commun.*, 25(2):97–116, 2012.
4. A. Biere, M. Heule, H. van Maaren, and T. Walsh, editors. *Handbook of Satisfiability*, volume 185 of *Frontiers in Artificial Intelligence and Applications*. IOS Press, 2009.
5. J. Boyar, P. Matthews, and R. Peralta. Logic minimization techniques with applications to cryptology. *J. Cryptology*, 26(2):280–312, 2013.
6. R. K. Brayton and A. Mishchenko. ABC: an academic industrial-strength verification tool. In *CAV*, pages 24–40, 2010.
7. R. K. Brayton, A. L. Sangiovanni-Vincentelli, C. T. McMullen, and G. D. Hachtel. *Logic Minimization Algorithms for VLSI Synthesis*. Kluwer Academic Publishers, Norwell, MA, USA, 1984.
8. P. Cabalar, D. Pearce, and A. Valverde. Minimal logic programs. In *ICLP*, pages 104–118, 2007.
9. H. Chen, M. Janota, and J. Marques-Silva. QBF-based Boolean function bi-decomposition. In *DATE*, pages 816–819, 2012.
10. O. Coudert. Two-level logic minimization: an overview. *Integration*, 17(2):97–140, 1994.
11. N. Courtois, D. Hulme, and T. Mourouzis. Solving circuit optimisation problems in cryptography and cryptanalysis. *IACR Cryptology ePrint Archive*, 2011:475, 2011.
12. W. Craig. Linear reasoning. A new form of the Herbrand-Gentzen theorem. *J. Symb. Log.*, 22(3):250–268, 1957.
13. D. Déharbe, P. Fontaine, D. L. Berre, and B. Mazure. Computing prime implicants. In *FMCAD*, pages 46–52, 2013.
14. M. Durzinsky, A. Wagler, and W. Marwan. Reconstruction of extended petri nets from time series data and its application to signal transduction and to gene regulatory networks. *BMC Systems Biology*, 5:113, 2011.
15. N. Eén and N. Sörensson. An extensible SAT-solver. In *SAT*, pages 502–518, 2003.
16. Espresso — multi-valued PLA minimization. `http://embedded.eecs.berkeley.edu/pubs/downloads/espresso`. Accessed: 2015-04-30.
17. E. Filiol. Malware pattern scanning schemes secure against black-box analysis. *Journal in Computer Virology*, 2(1):35–50, 2006.
18. M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1979.
19. J. Ghosh, S. J. Philip, and C. Qiao. Sociological orbit aware location approximation and routing (SOLAR) in MANET. *Ad Hoc Networks*, 5(2):189–209, 2007.
20. É. Grégoire, J. Lagniez, and B. Mazure. An experimentally efficient method for (MSS, CoMSS) partitioning. In *AAAI*, pages 2666–2673, 2014.
21. A. Gupta. *Learning Abstractions for Model Checking*. PhD thesis, Carnegie Mellon University, June 2006.
22. G. D. Hachtel and F. Somenzi. *Logic synthesis and verification algorithms*. Kluwer, 1996.
23. A. Ignatiev, M. Janota, and J. Marques-Silva. Quantified maximum satisfiability: A core-guided approach. In *SAT*, pages 250–266, 2013.
24. A. Ignatiev, M. Janota, and J. Marques-Silva. Quantified maximum satisfiability. *Constraints*, 2015. `http://dx.doi.org/10.1007/s10601-015-9195-9`.
25. A. Ignatiev, A. Morgado, V. M. Manquinho, I. Lynce, and J. Marques-Silva. Progression in maximum satisfiability. In *ECAI*, pages 453–458, 2014.
26. A. Ignatiev, A. Previti, M. Liffiton, and J. Marques-Silva. Smallest MUS extraction with minimal hitting set dualization. In *CP*, 2015.

27. S. Jabbour, J. Marques-Silva, L. Sais, and Y. Salhi. Enumerating prime implicants of propositional formulae in conjunctive normal form. In *JELIA*, pages 152–165, 2014.
28. D. Lafond, Y. Lacouture, and G. Mineau. Complexity minimization in rule-based category learning: Revising the catalog of Boolean concepts and evidence for non-minimal rules. *Journal of Mathematical Psychology*, 51(2):57–74, 2007.
29. R. Lee, J. R. Jiang, and W. Hung. Bi-decomposing large Boolean functions via interpolation and satisfiability solving. In *DAC*, pages 636–641, 2008.
30. P. Liberatore. Redundancy in logic I: CNF propositional formulae. *Artif. Intell.*, 163(2):203–232, 2005.
31. M. H. Liffiton, M. N. Mneimneh, I. Lynce, Z. S. Andraus, J. Marques-Silva, and K. A. Sakallah. A branch and bound algorithm for extracting smallest minimal unsatisfiable subformulas. *Constraints*, 14(4):415–442, 2009.
32. M. H. Liffiton, A. Previti, A. Malik, and J. Marques-Silva. Fast, flexible MUS enumeration. *Constraints*, 2015. http://dx.doi.org/10.1007/s10601-015-9183-0.
33. M. H. Liffiton and K. A. Sakallah. Algorithms for computing minimal unsatisfiable subsets of constraints. *J. Autom. Reasoning*, 40(1):1–33, 2008.
34. I. Lynce and J. Marques-Silva. On computing minimum unsatisfiable cores. In *SAT*, 2004.
35. J. Marques-Silva, F. Heras, M. Janota, A. Previti, and A. Belov. On computing minimal correction subsets. In *IJCAI*, pages 615–622, 2013.
36. E. J. McCluskey. Minimization of Boolean functions. *Bell system technical Journal*, 35(6):1417–1444, 1956.
37. A. Meier and V. Sorge. A new set of algebraic benchmark problems for SAT solvers. In *SAT*, pages 459–466, 2005.
38. M. N. Mneimneh, I. Lynce, Z. S. Andraus, J. Marques-Silva, and K. A. Sakallah. A branch-and-bound algorithm for extracting smallest minimal unsatisfiable formulas. In *SAT*, pages 467–474, 2005.
39. A. Morgado, C. Dodaro, and J. Marques-Silva. Core-guided MaxSAT with soft cardinality constraints. In *CP*, pages 564–573, 2014.
40. A. Morgado, F. Heras, M. H. Liffiton, J. Planes, and J. Marques-Silva. Iterative and core-guided maxsat solving: A survey and assessment. *Constraints*, 18(4):478–534, 2013.
41. A. Nadel. Boosting minimal unsatisfiable core extraction. In *FMCAD*, pages 221–229, 2010.
42. A. Nadel, V. Ryvchin, and O. Strichman. Efficient MUS extraction with resolution. In *FMCAD*, pages 197–200, 2013.
43. A. Nöhrer, A. Biere, and A. Egyed. Managing SAT inconsistencies with HUMUS. In *VAMOS*, pages 83–91, 2012.
44. C. H. Papadimitriou. *Computational complexity*. Addison-Wesley, 1994.
45. A. Previti, A. Ignatiev, A. Morgado, and J. Marques-Silva. Prime compilation of non-clausal formulae. In *IJCAI*, pages 1980–1987, 2015.
46. C. Priesterjahn, D. Steenken, and M. Tichy. Timed hazard analysis of self-healing systems. In *ASAS*, pages 112–151. 2013.
47. W. V. Quine. The problem of simplifying truth functions. *American mathematical monthly*, pages 521–531, 1952.
48. W. V. Quine. A way to simplify truth functions. *American mathematical monthly*, pages 627–631, 1955.
49. M. Rehák, E. Staab, V. Fusenig, J. Stiborek, M. Grill, K. Bartos, M. Pechoucek, and T. Engel. Threat-model-driven runtime adaptation and evaluation of intrusion detection system. In *ICAC*, pages 65–66, 2009.
50. R. Reiter. A theory of diagnosis from first principles. *Artif. Intell.*, 32(1):57–95, 1987.
51. R. L. Rudell and A. L. Sangiovanni-Vincentelli. Multiple-valued minimization for PLA optimization. *IEEE Trans. on CAD of Integrated Circuits and Systems*, 6(5):727–750, 1987.
52. S. Sapra, M. Theobald, and E. M. Clarke. SAT-based algorithms for logic minimization. In *ICCD*, pages 510–517, 2003.

53. T. Sasao. Input variable assignment and output phase optimization of PLA's. *IEEE Trans. Computers*, 33(10):879–894, 1984.
54. R. Vigo. A note on the complexity of Boolean concepts. *Journal of Mathematical Psychology*, 50(5):501–510, 2006.
55. M. Wu. Query optimization for selections using bitmaps. In *SIGMOD*, pages 227–238, 1999.
56. M. Yagiura and T. Ibaraki. Efficient 2 and 3-flip neighborhood search algorithms for the MAX SAT: experimental evaluation. *J. Heuristics*, 7(5):423–442, 2001.
57. S. Yang. Logic synthesis and optimization benchmarks user guide: Version 3.0. Technical Report 1991-IWLS-UG-Saeyang, MCNC, Research Triangle Park, January 1991.
58. Z. Zhou, D. Huang, and Z. Wang. Efficient privacy-preserving ciphertext-policy attribute based-encryption and broadcast encryption. *IEEE Trans. Computers*, 64(1):126–138, 2015.