

LOCAL REDUNDANCY IN SAT: GENERALIZATIONS OF BLOCKED CLAUSES

BENJAMIN KIESL¹, MARTINA SEIDL², HANS TOMPITS¹, AND ARMIN BIERE²

¹ Institute of Logic and Computation, TU Wien

e-mail address: kiesel@kr.tuwien.ac.at

e-mail address: tompits@kr.tuwien.ac.at

² Institute for Formal Models and Verification, Johannes Kepler University

e-mail address: martina.seidl@jku.at

e-mail address: biere@jku.at

ABSTRACT. Clause-elimination procedures that simplify formulas in conjunctive normal form play an important role in modern SAT solving. Before or during the actual solving process, such procedures identify and remove clauses that are irrelevant to the solving result. These simplifications usually rely on so-called redundancy properties that characterize cases in which the removal of a clause does not affect the satisfiability status of a formula. One particularly successful redundancy property is that of blocked clauses, because it generalizes several other redundancy properties. To find out whether a clause is blocked—and therefore redundant—one only needs to consider its resolution environment, i.e., the clauses with which it can be resolved. For this reason, we say that the redundancy property of blocked clauses is local. In this paper, we show that there exist local redundancy properties that are even more general than blocked clauses. We present a semantic notion of blocking and prove that it constitutes the most general local redundancy property. We furthermore introduce the syntax-based notions of set-blocking and super-blocking, and show that the latter coincides with our semantic blocking notion. In addition, we show how semantic blocking can be alternatively characterized via Davis and Putnam’s rule for eliminating atomic formulas. Finally, we perform a detailed complexity analysis and relate our novel redundancy properties to prominent redundancy properties from the literature.

Key words and phrases: SAT, propositional logic, blocked clauses, redundancy properties.

This is an extended version of the paper “Super-Blocked Clauses” [19] which has appeared in the Proceedings of the 8th International Joint Conference on Automated Reasoning (IJCAR 2016).

This work has been supported by the Austrian Science Fund (FWF) under projects W1255-N23 and S11408-N23.

INTRODUCTION

Over the last two decades, there has been enormous progress in the performance of SAT solvers, i.e., decision procedures for the satisfiability problem of propositional logic (SAT) [4]. As a consequence, SAT solvers have become attractive reasoning engines in many user domains like the verification of hardware and software [29] as well as in the backends of other reasoning tools like SMT solvers [2], QBF solvers [15, 27], or even first-order theorem provers [26]. In such applications, however, SAT solvers often reach their limits, motivating the quest for more efficient SAT techniques.

One successful approach to improving the performance of SAT solvers is the use of clause-elimination procedures. Either before (“preprocessing”) or during the actual solving (“inprocessing”) such procedures simplify a given formula in conjunctive normal form (CNF) by removing clauses that are irrelevant to the outcome of the solving process [1, 5, 9, 11, 14, 17, 18, 24]. To distinguish the irrelevant from the relevant, they usually rely on so-called *redundancy properties* that characterize cases in which certain clauses can be removed from a formula without affecting its satisfiability status [11, 18]. For instance, a clause-elimination procedure that is based on the simple redundancy property of *tautologies* identifies and removes tautological clauses—clauses that contain two complementary literals. As these clauses are true regardless of a particular truth assignment, their removal does not influence the outcome of the solving process.

A particularly useful redundancy property is that of *blocked clauses* [16, 21]. Informally, a clause C is *blocked* in a CNF formula F if it contains a literal l such that all possible resolvents of C upon l (with clauses from F) are tautologies. The elimination of blocked clauses considerably improves the performance of modern SAT solvers [16, 24]. Moreover, blocked clauses provide the basis for *blocked-clause decomposition*, a technique that splits a CNF formula into two parts that become solvable by blocked-clause elimination [9]. Blocked-clause decomposition is successfully used for gate extraction, for efficiently finding backbone variables, and for the detection of implied binary equivalences [1, 14]. The winner of the SAT-Race 2015 competition, the solver *abcdSAT* [5], uses blocked-clause decomposition as core technology.

Part of the reason for the success of blocked clauses is the fact that they generalize several other well-known redundancy properties such as those of *pure literals* [6] or the above-mentioned tautologies. This means that a tool for blocked-clause elimination implicitly performs the elimination of pure literals and tautologies (and even more aggressive formula simplifications on top). A closer look at the definition of blocked clauses reveals that to check whether a clause is blocked, it suffices to consider only its resolution environment, i.e., the clauses with which it can be resolved. Because of this, we say that the blocked-clauses redundancy property is *local*. Their locality aspect makes blocked clauses particularly effective when dealing with large formulas in which the resolution environments of clauses are small.

These success stories motivate us to have a closer look at local redundancy properties in general and at blocked clauses in particular. We show in this paper that blocked clauses do not constitute the most general concept of a local redundancy property. In particular, we first present some observations on blocked clauses and provide examples of redundant clauses that are not blocked, yet their redundancy can be identified by considering only their resolution environment. We next use these observations to derive a semantic notion of blocking for which we show that it constitutes the most general local redundancy property. To bring

this semantic notion of blocking closer to practical SAT solving, we then introduce the syntax-based redundancy properties of *set-blocked clauses* and *super-blocked clauses*—both are strictly more general than traditional blocked clauses and for super-blocking we prove that it coincides with our semantic blocking notion.

After introducing the new redundancy properties, we present an alternative syntactic characterization of semantic blocking based on Davis and Putnam’s *rule for eliminating atomic formulas* [6]—a resolution-based rewriting rule that is also known as *variable elimination* [7] in the context of SAT solving. This characterization further clarifies the connection between traditional blocking and semantic blocking. We then proceed by analyzing the complexity of deciding whether a clause is redundant with respect to our novel redundancy properties. Our complexity analysis gives rise to a whole family of redundancy properties that are obtained by restricting the notions of set-blocking and super-blocking in certain ways. Finally, we outline the relationship of the new redundancy properties to prominent redundancy properties from the literature before concluding with an outlook to future work.

A preliminary version of this paper appeared in the proceedings of IJCAR 2016 [19]. Besides several extensions of the text, this version now includes the full proofs of our complexity results while the conference paper [19] contained only proof sketches. In addition, we introduce an alternative characterization of semantic blocking based on variable elimination (cf. Section 5).

1. PRELIMINARIES

We consider propositional formulas in *conjunctive normal form* (CNF) which are defined as follows. A *literal* is either a Boolean variable x (a *positive literal*) or the negation $\neg x$ of a variable x (a *negative literal*). For a literal l , we define $\bar{l} = \neg x$ if $l = x$ and $\bar{l} = x$ if $l = \neg x$. Accordingly, for a set L of literals, we define $\bar{L} = \{\bar{l} \mid l \in L\}$. A *clause* is a disjunction of literals. A *formula* is a conjunction of clauses. We identify a clause with a set of literals and a formula with a set of clauses. A *tautology* is a clause that contains both l and \bar{l} for some literal l . For a literal, clause, or formula E , $var(E)$ denotes the set of variables occurring in E . For convenience, we often treat $var(E)$ as a variable if E is a literal. For a set L of literals and a formula F , we denote by F_L the set of all clauses in F that contain a literal of L , i.e., $F_L = \{C \mid C \in F \text{ and } C \cap L \neq \emptyset\}$. In case L is a singleton set of the form $\{l\}$, we sometimes write F_l instead of $F_{\{l\}}$.

An *assignment* is a function from a set of variables to the truth values 1 (*true*) and 0 (*false*). An assignment is *total* with respect to a formula if it assigns a truth value to all variables occurring in the formula. Unless stated otherwise, we do not assume assignments to be total. Assignments are extended from variables to literals by defining $\tau(l) = \tau(var(l))$ if the literal l is positive and by defining $\tau(l) = 1 - \tau(var(l))$ if l is negative. Given an assignment τ that assigns a truth value to a literal l , we denote by τ_l the assignment obtained from τ by interchanging (“flipping”) the truth value of l , i.e., by defining $\tau_l(x) = 1 - \tau(x)$ if $x = var(l)$ and $\tau_l(x) = \tau(x)$ otherwise.

A literal l is *satisfied* by an assignment τ if $\tau(l) = 1$. A clause is satisfied by an assignment τ if it contains a literal that is satisfied by τ . Finally, a formula is satisfied by an assignment τ if all of its clauses are satisfied by τ . A formula is *satisfiable* if there exists an assignment that satisfies it. Two formulas are *logically equivalent* if they are satisfied by the same total assignments. Two formulas F and F' are *satisfiability equivalent* if either both F and F' are satisfiable or both F and F' are unsatisfiable. We sometimes say that an

assignment τ *falsifies* a literal l if $\tau(l) = 0$. Accordingly τ falsifies a clause C if τ falsifies all literals of C .

Given two clauses C and D together with a literal $l \in C$ such that $\bar{l} \in D$, the clause $(C \setminus \{l\}) \cup (D \setminus \{\bar{l}\})$ is the *resolvent* of C and D upon l and denoted by $C \otimes_l D$. Given a formula F and a clause C , the *resolution environment* of C in F is the set of all clauses in F that can be resolved with C :

Definition 1.1. The *resolution environment* of a clause C with respect to a formula F is the clause set $env_F(C) = \{D \in F \mid \exists l \in D \text{ such that } \bar{l} \in C\}$.

Given a formula F and a clause C , we call the variables in $var(C)$ the *local variables* and the variables in $var(env_F(C)) \setminus var(C)$ the *external variables*. We denote the set of external variables by $ext_F(C)$.

Next, we formally introduce redundancy of clauses. Intuitively, a clause C is redundant with respect to a formula F if neither its addition to F nor its removal from F changes the satisfiability or unsatisfiability of F .

Definition 1.2. A clause C is *redundant* with respect to a formula F if $F \setminus \{C\}$ and $F \cup \{C\}$ are satisfiability equivalent. A *redundancy property* is a set of pairs (F, C) where C is redundant with respect to F . Finally, for two redundancy properties \mathcal{P}_1 and \mathcal{P}_2 , \mathcal{P}_1 is *more general* than \mathcal{P}_2 if $\mathcal{P}_2 \subseteq \mathcal{P}_1$, i.e., if every pair $(F, C) \in \mathcal{P}_2$ is also contained in \mathcal{P}_1 . If $\mathcal{P}_2 \subset \mathcal{P}_1$, then \mathcal{P}_1 is *strictly more general* than \mathcal{P}_2 .

Example 1.3. Consider the formula $F = \{(a \vee b), (\neg a \vee \neg b)\}$. The clause $C = (\neg a \vee \neg b)$ is redundant with respect to F since $F \setminus \{C\}$ and $F \cup \{C\}$ are satisfiability equivalent (although they are not logically equivalent). Moreover, the set

$$\{(F, C) \mid F \text{ is a formula and } C \text{ is a tautology}\}$$

is a redundancy property since for every formula F and every tautology C , $F \setminus \{C\}$ and $F \cup \{C\}$ are satisfiability equivalent.

Note that a clause C is *not* redundant with respect to a formula F if and only if $F \setminus \{C\}$ is satisfiable and $F \cup \{C\}$ is unsatisfiable. To prove that C is redundant with respect to F , it therefore suffices to show that satisfiability of $F \setminus \{C\}$ implies satisfiability of $F \cup \{C\}$. Redundancy properties as defined above yield the basis not only for clause-elimination but also for clause-addition procedures [18].

2. OBSERVATIONS ON BLOCKED CLAUSES

Following Heule et al. [11] we recapitulate the definition of blocked clauses. In the rest of the paper, we refer to this kind of blocked clauses as *literal-blocked clauses*. Motivated by the examples given in this section, we then generalize this traditional notion of blocking to more powerful redundancy properties.

Definition 2.1. A clause C is *blocked* by a literal $l \in C$ in a formula F if $C \cup (D \setminus \{\bar{l}\})$ is a tautology for each clause $D \in F_{\bar{l}}$. A clause C is *literal-blocked* in F if there exists a literal that blocks C in F . By BC we denote the set $\{(F, C) \mid C \text{ is literal-blocked in } F\}$.

Example 2.2. Consider the formula $F = \{(\neg a \vee c), (\neg b \vee \neg a)\}$ and the clause $C = (a \vee b)$. The literal b blocks C in F since the only clause in F that contains $\neg b$ is the clause D with $D = (\neg b \vee \neg a)$ and further $C \cup (D \setminus \{\bar{l}\}) = (a \vee b \vee \neg a)$ is a tautology.

Note that this definition of literal-blocked clauses differs slightly from Kullman’s original definition [21]. The original definition requires that all *resolvents* of C upon a literal l are tautologies. However, it is not necessary to remove l from $C \cup (D \setminus \{\bar{l}\})$ —as would be the case in the resolvent of C and D —to guarantee redundancy of literal-blocked clauses. If l is not removed from C , every tautology C is also a literal-blocked clause, which is not the case for the original definition.

Proposition 2.3. *BC is a redundancy property.*

Proposition 2.3 paraphrases results from Heule et al. [11] and actually follows from our present results given later on (namely from Proposition 4.3 and Corollary 4.9). To see that blocked clauses are redundant, let C be a clause that is blocked by a literal l in a formula F and assume that some assignment τ satisfies $F \setminus \{C\}$ but falsifies C . We can then easily turn τ into a satisfying assignment τ_l of C by flipping the truth value of l .

This flipping could only possibly falsify some of the clauses in $F \setminus \{C\}$ that contain \bar{l} , but the condition that l blocks C guarantees that these clauses stay satisfied: Let D be such a clause that contains \bar{l} . Then, since $C \cup (D \setminus \{\bar{l}\})$ is a tautology, either D is itself a tautology (and therefore trivially satisfied) or it contains a literal $l' \neq l$ such that $\bar{l}' \in C$. In the latter case, since τ falsifies C , it must satisfy l' and since τ_l agrees with τ on all literals but l , τ_l is a satisfying assignment of $F \cup \{C\}$. Hence, C is redundant with respect to F .

Next, we illustrate with an example how a satisfying assignment of $F \cup \{C\}$ can be obtained from one of $F \setminus \{C\}$ [11]. This approach is used to obtain a satisfying assignment of the original formula when literal-blocked clauses have been removed during pre- or inprocessing: Suppose a SAT solver gets an input formula F and removes literal-blocked clauses to obtain a simplified formula F' . The solver then proceeds by searching for a satisfying assignment of F' . Once it has found such an assignment, it can easily turn it into a satisfying assignment of the original formula F .

Example 2.4. Consider again the formula $F = \{(-a \vee c), (\neg b \vee \neg a)\}$ and the clause $C = (a \vee b)$ from Example 2.2. We already know that b blocks C in F . So, let τ denote the assignment that falsifies the variables a , b , and c . Clearly, τ satisfies F but falsifies C . Now, the assignment τ_b , obtained from τ by flipping the truth value of b , satisfies not only C but also all clauses of F : The only clause that could have been falsified by flipping the truth value of b is $(\neg b \vee \neg a)$. But, since τ satisfies $\neg a$ and τ_b agrees with τ on all variables except b , τ_b satisfies $F \cup \{C\}$.

Literal-blocked clauses generalize many other redundancy properties like *pure literal* or *tautology* [11]. One of their particularly important properties is that for testing whether some clause C is literal-blocked in a formula F it suffices to consider only those clauses of F that can be resolved with C , i.e., the clauses in the resolution environment $env_F(C)$ of C .

This raises the question whether there exist redundant clauses that are not blocked but whose redundancy can be identified by considering only their resolution environment. As shown in the next example, this is indeed the case:

$$x \vee \mathbf{b} \vee \neg \mathbf{a} \text{ --- } \mathbf{a} \vee \mathbf{b} \begin{cases} \neg \mathbf{b} \vee \neg x \\ \neg \mathbf{b} \vee \mathbf{a} \end{cases}$$

Figure 1: The clause $(a \vee b)$ from Example 2.5 and its resolution environment.

Example 2.5. Let $C = (a \vee b)$ and F an arbitrary formula with the resolution environment $env_F(C) = \{(x \vee b \vee \neg a), (\neg b \vee \neg x), (\neg b \vee a)\}$ (cf. Fig. 1). The clause C is not literal-blocked in F but redundant: Suppose there exists an assignment τ that satisfies F but falsifies C . Then, τ must satisfy either x or $\neg x$. If $\tau(x) = 1$, then C can be satisfied by flipping the truth value of a , resulting in assignment $\tau' = \tau_a$. Since $\tau'(x) = 1$, the clause $(x \vee b \vee \neg a)$ stays satisfied. In contrast, if $\tau(x) = 0$, we can satisfy C by the assignment τ'' obtained from τ by flipping the truth values of both a and b : The fact that $\tau''(b) = 1$ guarantees that $(x \vee b \vee \neg a)$ stays satisfied whereas $\tau''(x) = 0$ and $\tau''(a) = 1$ guarantee that both $(\neg b \vee \neg x)$ and $(\neg b \vee a)$ stay satisfied. Since flipping the truth values of literals in C does not affect the truth of clauses outside the resolution environment $env_F(C)$ of C , we obtain in both cases a satisfying assignment of F .

3. A SEMANTIC NOTION OF BLOCKING

In the examples of the preceding section, when arguing that a clause C is redundant with respect to some formula F , we showed that every assignment τ that satisfies $F \setminus \{C\}$ but falsifies C can be turned into a satisfying assignment τ' of $F \cup \{C\}$ by flipping the truth values of certain literals in C . Since this flipping only affects the truth of clauses in $env_F(C)$, it suffices to make sure that τ' satisfies $env_F(C)$ in order to guarantee that it satisfies $F \cup \{C\}$. This naturally leads to the following semantic notion of blocking:

Definition 3.1. A clause C is *semantically blocked* in a formula F if, for every satisfying assignment τ of $env_F(C)$, there exists a satisfying assignment τ' of $env_F(C) \cup \{C\}$ such that $\tau(v) = \tau'(v)$ for all $v \notin var(C)$. We denote the set $\{(F, C) \mid C \text{ is semantically blocked in } F\}$ by SEM_{BC} .

Note that clause C in Example 2.5 is semantically blocked in F . Note also that a clause is semantically blocked if its resolution environment is unsatisfiable.

Theorem 3.2. SEM_{BC} is a redundancy property.

Proof. Let F be a formula and C a clause that is semantically blocked in F . We show that $F \cup \{C\}$ is satisfiable if $F \setminus \{C\}$ is satisfiable. Suppose there exists a satisfying assignment τ of $F \setminus \{C\}$. We proceed by a case distinction.

CASE 1: C contains a literal l with $var(l) \notin var(F \setminus \{C\})$. Then, τ can be easily extended to a satisfying assignment τ' of $F \cup \{C\}$ by defining that τ' satisfies l .

CASE 2: $var(C) \subseteq var(F \setminus \{C\})$. In this case, τ is a total assignment with respect to $F \cup \{C\}$. Suppose that τ falsifies C . It follows that C is not a tautology and so it does not contain a literal l such that $\bar{l} \in C$, hence $C \notin env_F(C)$. Thus, $env_F(C) \subseteq F \setminus \{C\}$ and so τ satisfies $env_F(C)$. Since C is semantically blocked in F , there exists a satisfying assignment τ' of $env_F(C) \cup \{C\}$ such that $\tau(v) = \tau'(v)$ for all $v \notin var(C)$. Now, since $\tau'(v)$ differs from τ only on variables in $var(C)$, the only clauses in F that could possibly be falsified by τ' are those with a literal \bar{l} such that $l \in C$. But those are exactly the clauses in $env_F(C)$, so τ' satisfies $F \cup \{C\}$.

Hence, C is redundant with respect to F and thus SEM_{BC} is a redundancy property. \square

If a clause C is redundant with respect to some formula F and if this redundancy can be identified by considering only its resolution environment in F , then we expect C to be

redundant with respect to every formula F' in which C has the same resolution environment as in F . This leads us to the notion of *local* redundancy properties:

Definition 3.3. A redundancy property \mathcal{P} is *local* if, for every clause C and any two formulas F, F' with $env_F(C) = env_{F'}(C)$, it holds that $(F, C) \in \mathcal{P}$ if and only if $(F', C) \in \mathcal{P}$.

The following is easily seen to hold:

Theorem 3.4. SEM_{BC} is a local redundancy property.

Preparatory for showing that SEM_{BC} is actually the most general local redundancy property (cf. Theorem 3.6 below), we first prove the following lemma:

Lemma 3.5. If a clause C is not semantically blocked in a formula F , then there exists a formula F' with $env_{F'}(C) = env_F(C)$ such that C is not redundant with respect to F' .

Proof. Let F be a formula and C a clause that is not semantically blocked in F , i.e., there exists a satisfying assignment τ of $env_F(C)$ but there does not exist a satisfying assignment τ' of $env_F(C) \cup \{C\}$ such that $\tau(v) = \tau'(v)$ for all $v \notin var(C)$. We define the set T of (unit) clauses as follows:

$$T = \{(v) \mid v \notin var(C) \text{ and } \tau(v) = 1\} \cup \{(\neg v) \mid v \notin var(C) \text{ and } \tau(v) = 0\}.$$

We further define the formula $F' = env_F(C) \cup \{C\} \cup T$.

Since C can be falsified (and is therefore not a tautology) and since the clauses in T contain only literals with variables that do not occur in C , it holds that neither C nor any of the clauses in T contain a literal \bar{l} with $l \in C$. It therefore holds that $env_{F'}(C) = env_F(C)$.

Now observe the following: The assignment τ satisfies $env_F(C)$ and, by construction, also T , hence $F' \setminus \{C\} = env_F(C) \cup T$ is satisfiable. Furthermore, every satisfying assignment of T must agree with τ on all variables $v \notin var(C)$. But there exists no satisfying assignment τ' of $env_F(C) \cup \{C\}$ such that $\tau(v) = \tau'(v)$ for all $v \notin var(C)$. Hence, $env_F(C) \cup \{C\} \cup T = F' \cup \{C\}$ is unsatisfiable and thus $F' \setminus \{C\}$ and $F' \cup \{C\}$ are not satisfiability equivalent. It follows that C is not redundant with respect to F' . \square

Theorem 3.6. SEM_{BC} is the most general local redundancy property.

Proof. Suppose there exists a local redundancy property \mathcal{P} that is strictly more general than SEM_{BC} . It follows that \mathcal{P} contains a pair (F, C) such that C is not semantically blocked in F . By Lemma 3.5, there exists a formula F' with $env_{F'}(C) = env_F(C)$ such that C is not redundant with respect to F' . But since \mathcal{P} is local and $env_{F'}(C) = env_F(C)$, it follows that $(F', C) \in \mathcal{P}$, hence \mathcal{P} is not a redundancy property, a contradiction. \square

4. SET-BLOCKED CLAUSES AND SUPER-BLOCKED CLAUSES

In the following, we introduce syntax-based notions of blocking which strictly generalize the original notion of literal-blocking as given in Definition 2.1. We will first introduce the notion of set-blocking which is already a strict generalization of literal-blocking. This notion will then be further generalized to the so-called notion of super-blocking which, as we will prove, coincides with the notion of semantic blocking given in Definition 3.1.

Definition 4.1. Let F be a formula and C a clause. A non-empty set $L \subseteq C$ blocks C in F if, for each clause $D \in F_{\bar{L}}$, $(C \setminus L) \cup \bar{L} \cup D$ is a tautology. We say that a clause is *set-blocked in F* if there exists a set that blocks it in F . We write SET_{BC} to refer to $\{(F, C) \mid C \text{ is set-blocked in } F\}$.

Example 4.2. Let $C = (a \vee b)$ and $F = \{(\neg a \vee b), (\neg b \vee a)\}$. Then, C is set-blocked by $L = \{a, b\}$: Clearly, $F_{\bar{L}} = F$ and $C \setminus L = \emptyset$. Therefore, for $D_1 = (\neg a \vee b)$ we get that $(C \setminus L) \cup \bar{L} \cup D_1 = (\neg a \vee b \vee \neg b)$ is a tautology and for $D_2 = (\neg b \vee a)$ we get that $(C \setminus L) \cup \bar{L} \cup D_2 = (a \vee \neg a \vee \neg b)$ is a tautology too. Note that C is not literal-blocked in F .

Given an assignment τ that satisfies $F \setminus \{C\}$ but falsifies C , the existence of a blocking set L guarantees that a satisfying assignment τ' of $F \cup \{C\}$ can be obtained from τ by flipping the truth values of all the literals in L . Since $(C \setminus L) \cup \bar{L} \cup D$ is a tautology for every clause D in the resolution environment of C , at least one of the following holds:

- (i) D is itself a tautology and thus satisfied by τ' , or
- (ii) D contains a literal of L which is satisfied by τ' since its truth value is flipped, or
- (iii) D contains a literal l which is satisfied since $\bar{l} \in C$ is falsified by τ and the truth value of l is not flipped.

Hence, τ' satisfies $F \cup \{C\}$.

Proposition 4.3. *Set-blocking is strictly more general than literal-blocking, i.e., it holds that $\text{BC} \subset \text{SET}_{\text{BC}}$.*

Proof. Example 4.2 shows that $\text{BC} \neq \text{SET}_{\text{BC}}$. It remains to show that $\text{BC} \subseteq \text{SET}_{\text{BC}}$. Let F be a formula and let C be a literal-blocked clause in F . We distinguish two cases:

CASE 1: C is a tautology. We have that $\{l, \bar{l}\} \subseteq C$ for some literal l . Let $L = \{l, \bar{l}\}$. Then, $(C \setminus L) \cup \bar{L} \cup D$ is a tautology for every $D \in F_{\bar{L}}$.

CASE 2: C is not a tautology. Since C is literal-blocked, there exists some literal $l \in C$ such that for every clause $D \in F_{\bar{l}}$, $C \cup (D \setminus \{\bar{l}\})$ is a tautology. Let $L = \{l\}$ and let $D \in F_{\bar{L}}$. Then, since $F_{\bar{L}} = F_{\bar{l}}$, it follows that $C \cup (D \setminus \{\bar{l}\})$ is a tautology. As C is not a tautology, D must contain some literal $l' \neq \bar{l}$ such that $\bar{l}' \in C \cup (D \setminus \{\bar{l}\})$. Now, since $l' \neq \bar{l}$ we have that $\bar{l}' \neq l$ and thus $\bar{l}' \in (C \setminus \{l\}) \cup D$. But then, $(C \setminus L) \cup \bar{L} \cup D$ is a tautology.

It follows that C is set-blocked in F and therefore $\text{BC} \subseteq \text{SET}_{\text{BC}}$. \square

We already argued informally why set-blocked clauses are redundant. That SET_{BC} is indeed a redundancy property follows directly from the properties of super-blocked clauses, which we introduce next. To define super-blocked clauses, we first define the following formula modification that uses a (possibly partial) assignment to simplify a formula:

Definition 4.4. Given a formula F and an assignment τ , we denote by $F|\tau$ the set of clauses obtained from F by removing all clauses that are satisfied by τ .

Note that in the literature, $F|\tau$ sometimes denotes the set of clauses obtained from F by first removing all clauses that are satisfied by τ and then removing all literals that are falsified by τ . For our purposes, it suffices to remove only satisfied clauses. Before we next define super-blocked clauses, recall that the external variables, $\text{ext}_F(C)$, are the variables that occur in $\text{env}_F(C)$ but not in C .

Definition 4.5. A clause C is *super-blocked* in a formula F if, for every assignment τ over the external variables $\text{ext}_F(C)$, C is set-blocked in $F|\tau$. We write SUP_{BC} for the set $\{(F, C) \mid C \text{ is super-blocked in } F\}$.

Here, by “every assignment τ over the external variables $\text{ext}_F(C)$ ” we mean all assignments whose domain is *exactly* the set $\text{ext}_F(C)$ and not a strict superset thereof. A simple example for a super-blocked clause is the clause C in Example 2.5—although C is not set-blocked

in F , it is super-blocked in F because it is set-blocked in both $F|\tau$ and $F|\tau'$ for $\tau(x) = 1$ and $\tau'(x) = 0$. Again, the idea is that from an assignment τ that satisfies $F \setminus \{C\}$ but falsifies C , we can obtain a satisfying assignment τ' of $F \cup \{C\}$ by flipping the truth values of certain literals of C . However, for making sure that the flipping does not falsify any clause D in the resolution environment of C , we are now also allowed to take into account the truth values of literals $l \in D$ with $\text{var}(l) \in \text{ext}_F(C)$. This is in contrast to set-blocking, where (apart from cases where D is a tautology) we only consider the truth values of literals whose variables occur in C . Finally, note that if an assignment τ' is a superset of an assignment τ , then τ' satisfies all clauses that are satisfied by τ and thus the following statement holds:

Proposition 4.6. *Given a formula F and two assignments τ, τ' , if $\tau \subseteq \tau'$, then $F|\tau' \subseteq F|\tau$.*

From this it follows that if a clause is set-blocked in F , it is also set-blocked in $F|\tau$ for every assignment τ . We conclude:

Proposition 4.7. *Super-blocking is strictly more general than set-blocking, i.e., it holds that $\text{SET}_{\text{BC}} \subset \text{SUP}_{\text{BC}}$.*

Theorem 4.8. *A clause is super-blocked in a formula F if and only if it is semantically blocked in F , i.e., it holds that $\text{SUP}_{\text{BC}} = \text{SEM}_{\text{BC}}$.*

Proof. For the “only if” direction, let C be a clause that is super-blocked in F and let τ be a satisfying assignment of $\text{env}_F(C)$. If τ satisfies C , or C contains a literal l with $\text{var}(l) \notin \text{var}(F)$ (implying that τ can be straightforwardly extended to a satisfying assignment of C), then it trivially follows that C is semantically blocked in F . Assume thus that $\text{var}(C) \subseteq \text{var}(F)$ and that τ does not satisfy C . Furthermore, let τ_E be obtained from τ by restricting it to the external variables $\text{ext}_F(C)$. Since C is super-blocked in F , there exists a non-empty set $L \subseteq C$ that blocks C in $F|\tau_E$. Consider the following assignment:

$$\tau'(v) = \begin{cases} 0 & \text{if } \neg v \in L, \\ 1 & \text{if } v \in L, \\ \tau(v) & \text{otherwise.} \end{cases}$$

Since τ falsifies C there is no literal l with $\{l, \bar{l}\} \subseteq L$, hence τ' is well-defined. Clearly, τ' satisfies C and $\tau'(v) = \tau(v)$ for all $v \notin \text{var}(C)$. It remains to show that τ' satisfies $\text{env}_F(C)$. Since τ' and τ differ only on the truth values of variables in $\text{var}(L)$, τ' can only falsify clauses containing a literal \bar{l} with $l \in L$. Let D be such a clause. We proceed by a case distinction.

CASE 1: D contains an external literal l (i.e., $\text{var}(l) \in \text{ext}_F(C)$) that is satisfied by τ . Then, since $\text{var}(l) \notin \text{var}(C)$ and thus $l \notin L$, it follows that τ' agrees with τ on the truth value of l , hence l is satisfied by τ' .

CASE 2: D does not contain an external literal that is satisfied by τ . In this case, D is contained in $F|\tau_E$ and thus, since L set-blocks C in $F|\tau_E$, we have that $(C \setminus L) \cup \bar{L} \cup D$ is a tautology. If D is a tautology, then it is easily satisfied by τ' , so assume that it is not a tautology. Clearly, since C is not a tautology, we have that $(C \setminus L) \cup \bar{L}$ is not a tautology, hence there are two literals l, \bar{l} such that $l \in D$ and \bar{l} is in $C \setminus L$ or in \bar{L} . If $\bar{l} \in C \setminus L$, then τ' agrees with τ on \bar{l} , hence \bar{l} is falsified by τ' and thus l is satisfied by τ' . In contrast, if $\bar{l} \in \bar{L}$, then $l \in L$ and l is satisfied by τ' . In both cases τ' satisfies l and thus it satisfies D .

For the “if” direction, let F be a formula and let C be a clause that is not super-blocked in F , i.e., there exists an assignment τ_E over the external variables, $ext_F(C)$, such that C is not set-blocked in $F|\tau_E$. Then, let

$$\tau(v) = \begin{cases} 1 & \text{if } \neg v \in C, \\ 0 & \text{if } v \in C, \\ \tau_E(v) & \text{otherwise.} \end{cases}$$

Clearly, τ is well-defined since C cannot be a tautology, for otherwise it would be set-blocked in $F|\tau_E$. Furthermore, τ falsifies C and, since (by definition) every clause $D \in env_F(C)$ contains a literal \bar{l} such that $l \in C$, it satisfies $env_F(C)$.

Now let τ' be a satisfying assignment of C such that $\tau'(v) = \tau(v)$ for all $v \notin var(C)$. As τ' satisfies C , it is obtained from τ by flipping the truth values of some literals $L \subseteq C$. We show that τ' does not satisfy $env_F(C)$. Clearly, τ' agrees with τ_E over the external variables, $ext_F(C)$, and since C is not set-blocked in $F|\tau_E$, there exists a clause $D \in F|\tau_E$ with $D \cap \bar{L} \neq \emptyset$ such that $(C \setminus L) \cup \bar{L} \cup D$ is not a tautology and neither τ_E nor τ' satisfy any external literal in D .

Let now $l \in D$ be a (local) literal with $var(l) \in var(C)$. Since $(C \setminus L) \cup \bar{L} \cup D$ is not a tautology it follows that $\bar{l} \notin C \setminus L$ and $\bar{l} \notin \bar{L}$. Since $var(l) \in var(C)$ we get that $l \in C \setminus L$ or $l \in \bar{L}$. In both cases, l is not satisfied by τ' . Thus, no literal in D is satisfied by τ' and consequently τ' does not satisfy $D \in env_F(C)$, which allows us to conclude that C is not semantically blocked in F . \square

Corollary 4.9. *SET_{BC} is a (local) redundancy property.*

5. THE RELATIONSHIP BETWEEN SEMANTIC BLOCKING AND VARIABLE ELIMINATION

In this section, we present an alternative characterization of semantic blocking based on Davis and Putnam’s *rule for eliminating atomic formulas* [6]—a resolution-based rewriting rule which is also known as *variable elimination* [7] in the context of SAT solving. Intuitively, a variable x is eliminated from a formula by first adding all possible non-tautological resolvents upon x and then removing all the clauses that contain x or $\neg x$. In the following, for a literal l , we denote by $F_l \otimes_l F_{\bar{l}}$ the set of all possible non-tautological resolvents upon l , i.e., $F_l \otimes_l F_{\bar{l}} = \{C \otimes_l D \mid C \in F_l \text{ and } D \in F_{\bar{l}} \text{ and } C \otimes_l D \text{ is not a tautology}\}$.

Definition 5.1. Let F be a formula, x a variable, and $F' = F \setminus (F_x \cup F_{\bar{x}})$. The *elimination of x from F* is given by the formula $F' \cup (F_x \otimes_x F_{\bar{x}})$.

If F does not contain any tautologies, then the variable x does not occur in the resulting formula $F' \cup (F_x \otimes_x F_{\bar{x}})$. Moreover, variable elimination does not affect satisfiability, meaning that F and $F' \cup (F_x \otimes_x F_{\bar{x}})$ are satisfiability equivalent [6].

Example 5.2. Let $F = \{(a \vee b), (x \vee b \vee \neg a), (\neg b \vee \neg x), (\neg b \vee a)\}$ (cf. Example 2.5). Then, $F_a = \{(a \vee b), (\neg b \vee a)\}$ and $F_{\bar{a}} = \{(x \vee b \vee \neg a)\}$. Furthermore, $F' = F \setminus (F_a \cup F_{\bar{a}}) = \{(\neg b \vee \neg x)\}$ and $F_a \otimes_a F_{\bar{a}} = \{(b \vee x)\}$. Finally, the elimination of the variable a from F yields the formula $F_1 = F' \cup (F_a \otimes_a F_{\bar{a}}) = \{(\neg b \vee \neg x), (b \vee x)\}$.

In the rest of this section, we show that the following relationship between semantic blocking and variable elimination holds: To test whether a non-tautological clause is semantically blocked in a tautology-free formula, we can successively eliminate all the clause’s variables

from its resolution environment. If this elimination yields the empty formula, then the clause is semantically blocked, otherwise it is not. The condition that the formula must be tautology-free is not a serious restriction as we can easily remove tautologies before variable elimination. The following example illustrates this relationship:

Example 5.3. Consider again the formula F from Example 5.2. As already shown earlier in Section 4, the clause $(a \vee b)$ from Example 5.2 is super-blocked—and therefore semantically blocked—in F . Because of this, we would expect that the elimination of a and b from F yields the empty formula. Indeed, in $F_1 = \{(-b \vee \neg x), (b \vee x)\}$ (which, as shown in Example 5.2, is obtained from F by eliminating the variable a) there is only a tautological resolvent upon b , namely $(\neg x \vee x)$, hence the elimination of a and b from F yields the empty formula.

In order to prove that this relationship between variable elimination and semantic blocking holds, we first introduce a simple encoding of semantic blocking into quantified Boolean formulas (QBFs). Given a clause C and a propositional formula F , the encoding produces a quantified Boolean formula that is true if and only if C is semantically blocked in F . Based on this encoding, we can then use a result from QBF theory that allows for a short proof of the main statement of this section (Theorem 5.7 on page 12). We therefore shortly recapitulate the syntax and semantics of quantified Boolean formulas [20].

Definition 5.4. A *quantified Boolean formula* (QBF) ϕ in *prenex conjunctive normal form* (PCNF) is of the form $\mathcal{Q}.F$ where \mathcal{Q} is a quantifier prefix and F , called the *matrix* of ϕ , is a propositional formula in CNF. A *quantifier prefix* has the form $Q_1X_1 \dots Q_nX_n$ with disjoint variable sets X_i , $Q_i \in \{\forall, \exists\}$, and $Q_i \neq Q_{i+1}$.

A QBF $\forall x \mathcal{Q}.F$ is true if both $\mathcal{Q}.F[x/\top]$ and $\mathcal{Q}.F[x/\perp]$ are true, and false otherwise, where $\mathcal{Q}.F[x/t]$ is the QBF obtained from $\mathcal{Q}.F$ by replacing all occurrences of the variable x by the truth constant t . Moreover, a QBF $\exists x \mathcal{Q}.F$ is true if at least one of $\mathcal{Q}.F[x/\top]$ and $\mathcal{Q}.F[x/\perp]$ is true, and false otherwise. If the matrix F of a QBF ϕ is empty after eliminating the truth constants according to standard propositional rules, then ϕ is true. Accordingly, ϕ is false if F contains the empty clause after eliminating the truth constants. In the rest of this section, we assume every QBF to be in PCNF.

The following lemma introduces our encoding of semantic blocking into quantified Boolean formulas. Recall that for a clause C and a formula F , $ext_F(C)$ denotes the set of external variables of C , i.e., the set of variables that occur in the resolution environment of C but not in C itself.

Lemma 5.5. *Let C be a non-tautological clause and F a propositional formula. Let furthermore $G = ext_F(C)$ and $L = var(C)$. Then, C is semantically blocked in F if and only if the QBF $\forall G \exists L.(env_F(C) \cup \{C\})$ is true.*

Proof. For the “only if” direction, assume that C is semantically blocked in F . We show that for every assignment τ_G over the variables in G , there exists an assignment τ_L over the variables in L such that $\tau_G \cup \tau_L$ satisfies $env_F(C) \cup \{C\}$. Let τ_G be an arbitrary assignment over the variables in G . To prove the existence of a corresponding τ_L , we first define the following assignment τ'_L :

$$\tau'_L(v) = \begin{cases} 1 & \text{if } \neg v \in C, \\ 0 & \text{if } v \in C. \end{cases}$$

Since C is not a tautology, τ'_L is well-defined. Furthermore, since τ'_L falsifies all literals of C and every clause D in $env_F(C)$ contains a literal \bar{l} with $l \in C$, the assignment $\tau_G \cup \tau'_L$ satisfies $env_F(C)$. Now, since C is semantically blocked, there exists an assignment τ_L over the variables in L such that $\tau_G \cup \tau_L$ satisfies $env_F(C) \cup \{C\}$. It therefore follows that the QBF $\forall G \exists L.(env_F(C) \cup \{C\})$ is true.

For the “if” direction, assume that the QBF $\forall G \exists L.(env_F(C) \cup \{C\})$ is true and let τ be a satisfying assignment of $env_F(C)$. We show that there exists a satisfying assignment τ' of $env_F(C) \cup \{C\}$ that agrees with τ on all the variables not occurring in C . To this end, let τ_G be obtained from τ by restricting it to the variables in G . Since $\forall G \exists L.(env_F(C) \cup \{C\})$ is true, there exists an assignment τ_L over the variables in L such that $\tau' = \tau_G \cup \tau_L$ satisfies $env_F(C) \cup \{C\}$. As τ' agrees with τ on all the variables in G , which are exactly the variables that do not occur in C , it follows that C is semantically blocked in F . \square

With Lemma 5.5 we can give a short proof of Theorem 5.7 because it allows us to use an important result from QBF theory, namely that the elimination of existential variables from the inner-most quantifier block does not affect the truth value of a QBF [12]:

Lemma 5.6. *Let $\mathcal{Q}\exists X.F$ be a QBF where F does not contain any tautologies. Let furthermore F' be obtained from F by eliminating a variable $x \in X$. Then, $\mathcal{Q}\exists X.F$ is true if and only if $\mathcal{Q}\exists(X \setminus \{x\}).F'$ is true.*

We can now finally state and prove the main result of this section:

Theorem 5.7. *Let C be a non-tautological clause, F a propositional formula, and E' obtained from $env_F(C) \cup \{C\}$ by first removing all tautologies and then successively eliminating all variables that occur in C . Then, C is semantically blocked in F if and only if $E' = \emptyset$.*

Proof. Let C be a non-tautological clause, F a formula, and E' obtained from $env_F(C) \cup \{C\}$ by first removing all tautologies and then successively eliminating all variables that occur in C . We show that C is semantically blocked in F if and only if $E' = \emptyset$.

Let $G = ext_F(C)$ and $L = var(C)$. By Lemma 5.5, C is semantically blocked in F if and only if the QBF

$$\phi = \forall G \exists L.(env_F(C) \cup \{C\})$$

is true. Now, since both the removal of tautologies and—by Lemma 5.6—the elimination of existential variables from the inner-most quantifier block of a QBF preserve its truth value, C is semantically blocked in F if and only if the QBF $\phi' = \forall G.E'$ is true, i.e., if the propositional formula E' is valid.

For the “only if” direction, assume that $E' \neq \emptyset$, i.e., E' contains a non-tautological clause D such that $var(D) \subseteq G$. Then, the assignment τ_G that is defined in such a way that it falsifies all literals of D , falsifies E' . Hence, ϕ' is false and thus C is not semantically blocked.

For the “if” direction, assume that $E' = \emptyset$. Then, E' is trivially satisfied by every assignment over the variables in G and thus ϕ' is true. It follows that C is semantically blocked in F . \square

We want to highlight that for Theorem 5.7, the order in which variables are eliminated does not matter. Theorem 5.7 further clarifies the relationship between literal-blocking and semantic blocking: By definition, a non-tautological clause C is literal-blocked in a formula F if it contains a literal l such that all resolvents upon l are tautologies. Since tautological resolvents

are removed during variable elimination, we get the following alternative characterization of literal-blocking:

Proposition 5.8. *A non-tautological clause C is blocked by a literal $l \in C$ in a formula F if and only if the elimination of $\text{var}(l)$ from $\text{env}_F(C) \cup \{C\}$ yields the empty formula.*

In other words, literal-blocking requires that already the elimination of a single variable yields the empty formula while semantic blocking is more general by allowing for the elimination of several variables in order to derive the empty formula.

Note that Theorem 5.7 does not hold without the condition that C must be non-tautological. To see this, consider the following example:

Example 5.9. Let $C = (b \vee \neg b)$ and F a formula in which C has the resolution environment $\text{env}_F(C) = \{(a \vee \neg b), (\neg a \vee \neg b), (b \vee a), (b \vee \neg a)\}$. Then, C is semantically blocked but the elimination of b from $E' = \text{env}_F(C)$ (C is not in E' since it is a tautology) yields the (unsatisfiable) non-empty formula $\{(a), (\neg a)\}$.

To conclude this section, we note that the following theorem is a trivial consequence of Lemma 5.5:

Theorem 5.10. *Let C be a clause, F a formula, $G = \text{ext}_F(C)$, and $L = \text{var}(C)$. Then, C is semantically blocked in F if and only if it is a tautology or the QBF $\forall G \exists L. (\text{env}_F(C) \cup \{C\})$ is true.*

6. COMPLEXITY ANALYSIS

In this section, we analyze the complexity of testing whether a clause is set-blocked or super-blocked. We further consider the complexity of testing restricted variants of set-blocking and super-blocking which gives rise to a whole family of blocking notions. Note that all complexity results are with respect to the size of a clause and its resolution environment.

Definition 6.1. The *set-blocking problem* is the following decision problem: Given a pair (F, C) , where F is a set of clauses and C is a clause such that every clause $D \in F$ contains a literal \bar{l} with $l \in C$, decide whether C is set-blocked in F .

Theorem 6.2. *The set-blocking problem is NP-complete.*

Proof. We first show NP-membership, followed by NP-hardness.

NP-MEMBERSHIP: For a non-empty set $L \subseteq C$, it can be checked in polynomial time whether $(C \setminus L) \cup \bar{L} \cup D$ is a tautology for every clause D with $D \cap \bar{L} \neq \emptyset$. The following is thus an NP-procedure: Guess a non-empty set $L \subseteq C$ and check if it blocks C in F .

NP-HARDNESS: We give a reduction from SAT by defining the following reduction function on input formula F which is w.l.o.g. in CNF:

$$f(F) = (F', C), \text{ with } C = (u \vee x_1 \vee x'_1 \vee \dots \vee x_n \vee x'_n),$$

where $\text{var}(F) = \{x_1, \dots, x_n\}$ and u, x'_1, \dots, x'_n are new variables that do not occur in F . Furthermore, F' is obtained from F by

- replacing every clause $D \in F$ by a clause $t(D)$, obtained from D by adding $\neg u$ and replacing every negative literal $\neg x_i$ by the positive literal x'_i , and
- adding the clauses $(\neg x_i \vee \neg x'_i), (\neg x_i \vee u), (\neg x'_i \vee u)$ for every $x_i \in \text{var}(F)$.

The intuition behind the construction of F' and C is as follows. By including u in C and adding $\neg u$ to every $t(D)$ with $D \in F$, we guarantee that all clauses in F' are in the resolution environment of C , i.e., they contain a literal l with $\bar{l} \in C$. This makes (F', C) a valid instance of the set-blocking problem. The main idea, however, is, that blocking-sets L of C in F' correspond to satisfying assignments of F . We show that F is satisfiable if and only if C is set-blocked in F' .

For the “only-if” direction, assume that there exists a satisfying assignment τ of F and let $L = \{u\} \cup \{x_i \mid \tau(x_i) = 1\} \cup \{x'_i \mid \tau(x_i) = 0\}$. Clearly, $L \subseteq C$. It remains to show that for every $C' \in F'$ with $C' \cap \bar{L} \neq \emptyset$, $(C \setminus L) \cup \bar{L} \cup C'$ is a tautology. We proceed by a case distinction.

CASE 1: C' is of the form $(\neg x_i \vee u)$ or $(\neg x'_i \vee u)$ for $x_i \in X$. Then, since $\neg u \in \bar{L}$ and $u \in C'$, $(C \setminus L) \cup \bar{L} \cup C'$ is a tautology.

CASE 2: C' is of the form $(\neg x_i \vee \neg x'_i)$ for $x_i \in X$. In this case, by the definition of L , only one of x_i and x'_i is in L . Assume w.l.o.g. that $x_i \in L$. Then, $x'_i \in C \setminus L$ and since $\neg x'_i \in C'$, $(C \setminus L) \cup \bar{L} \cup C'$ is a tautology.

CASE 3: C' is of the form $t(D)$ for $D \in F$. Then, τ satisfies a literal $l \in D$. If l is positive, i.e., $l = x_i$ for some $x_i \in X$, then $x_i \in C'$ and $x_i \in L$. In contrast, if l is negative, i.e., $l = \neg x_i$ for $x_i \in X$, then $x'_i \in C'$ and $x'_i \in L$. In both cases, L contains a literal of C' . But then, $(C \setminus L) \cup \bar{L} \cup C'$ is a tautology.

Thus, L blocks C in F' .

For the “if” direction, suppose that C is blocked by some blocking-set L in F' and define τ over $\text{var}(F) = X$ as follows:

$$\tau(x_i) = \begin{cases} 1 & \text{if } x_i \in L, \\ 0 & \text{otherwise.} \end{cases}$$

We show that τ satisfies F . First, observe that u must be contained in L : Assume to the contrary that $u \notin L$. Then, since L is non-empty, some x_i or x'_i must be contained in L . If $x_i \in L$, then, since $C' = (\neg x_i \vee u)$ contains $\neg x_i$, $(C \setminus L) \cup \bar{L} \cup C'$ is a tautology. But, since $(C \setminus L) \cup \bar{L}$ cannot be a tautology, and $x_i \notin (C \setminus L) \cup \bar{L}$, this can only be the case if $u \in L$, a contradiction. The argument for $x'_i \in L$ is analogous.

Now, let $D \in F$ and $C' = t(D)$. Then, since $u \in L$ and $\neg u \in C'$, $(C \setminus L) \cup \bar{L} \cup C'$ is a tautology. Furthermore, C contains only positive literals and for C' this is, apart from $\neg u$, also the case. But, since $u \in L$, u is not contained in $(C \setminus L) \cup \bar{L}$ and thus $(C \setminus L) \cup \bar{L} \cup C'$ can only be a tautology if C' contains a literal $l \in L$ which is different from $\neg u$. Now, if $l = x_i$ for some $x_i \in X$, then $x_i \in D$ and $\tau(x_i) = 1$. If $l = \neg x'_i$ for $x_i \in X$, then $\neg x'_i \in D$ and $\tau(x_i) = 0$. In both cases, τ satisfies D . Hence, F is satisfied by τ . \square

We next analyze the complexity of testing whether a clause is super-blocked. To do so, we define the following problem:

Definition 6.3. The *super-blocking problem* is the following decision problem: Given a pair (F, C) , where F is a set of clauses and C a clause such that every $C' \in F$ contains a literal \bar{l} with $l \in C$, decide whether C is super-blocked in F .

Theorem 6.4. *The super-blocking problem is Π_2^P -complete.*

Proof. Since super-blocking coincides with semantic blocking, membership in Π_2^P is an immediate consequence of Theorem 5.10. For showing hardness, we give a reduction from

$\forall\exists$ -SAT to the super-blocking problem. The reduction is similar to the one used for proving Theorem 6.2. Here, only the existentially quantified variables of the $\forall\exists$ -formula are encoded into C which makes all the universally quantified variables external. Let $\phi = \forall X\exists Y F$ be an instance of $\forall\exists$ -SAT with $Y = \{y_1, \dots, y_n\}$ and assume w.l.o.g. that F is in CNF. We define the reduction function

$$f(\phi) = (F', C), \text{ with } C = (u \vee y_1 \vee y'_1 \vee \dots \vee y_n \vee y'_n),$$

where u, y'_1, \dots, y'_n are new variables not occurring in ϕ . Furthermore, F' is obtained from F by

- replacing every clause $D \in F$ by a clause $t(D)$ which is obtained from D by adding $\neg u$ and replacing every negative literal $\neg y_i$ by the positive literal y'_i for $y_i \in Y$; and
- adding the clauses $(\neg y_i \vee \neg y'_i), (\neg y_i \vee u), (\neg y'_i \vee u)$ for every $y_i \in Y$.

By construction, every clause $C' \in F'$ contains a literal \bar{l} with $l \in C$, hence (F', C) is a valid instance of the super-blocking problem. The intuition behind the reduction is that blocking sets L of C in F' correspond to assignments over the existential variables of ϕ while the assignments over the external variables, $ext_{F'}(C)$, correspond to the assignments over the universally quantified variables of ϕ . As super-blocking coincides with semantic blocking, we show that ϕ is true if and only if C is semantically blocked in F' .

For the “only-if” direction, assume that ϕ is true and that there exists a satisfying assignment τ of F' . We show that there exists a satisfying assignment τ' of $F' \cup \{C\}$ with $\tau'(v) = \tau(v)$ for all $v \notin C$. Let therefore τ_X be obtained from τ by restricting it to the variables in X . Since ϕ is true, there exists an assignment τ_Y such that $\tau_X \cup \tau_Y$ satisfies F . Consider now the following assignment:

$$\tau'(v) = \begin{cases} \tau_X \cup \tau_Y(v) & \text{if } v \in X \cup Y, \\ 0 & \text{if } v = y'_i \text{ and } \tau_Y(y_i) = 1 \text{ for } y_i \in Y, \\ 1 & \text{otherwise.} \end{cases}$$

Clearly, C is satisfied by τ' since $u \in C$ and $\tau'(u) = 1$ by definition. Furthermore, $\tau'(v) = \tau(v)$ for all $v \notin C$ since all variables that are not in C are contained in $ext_{F'}(C) = X$. We show that τ' also satisfies F' . Let therefore C' be an arbitrary clause in F' . We proceed by a case distinction.

CASE 1: C' is of the form $(\neg y_i \vee u)$ or $(\neg y'_i \vee u)$ for $y_i \in Y$. Then, C' is trivially satisfied.

CASE 2: C' is of the form $(\neg y_i \vee \neg y'_i)$ for $y_i \in Y$. Then, by definition of τ' , $\tau'(y_i) \neq \tau'(y'_i)$. Hence, one of y_i and y'_i must be satisfied by τ' .

CASE 3: $C' = t(C'')$ for $C'' \in F$. Since $\tau_X \cup \tau_Y$ satisfies F , there exists some literal $l \in C''$ such that l is satisfied by $\tau_X \cup \tau_Y$. Now, if $var(l) \in X$ or l is a positive literal, then l is also contained in C' and thus, C' is satisfied by τ' . In contrast, if l is a negative literal with $var(l) \in Y$, then C' contains the literal y'_i . Since $\neg y_i$ is satisfied by $\tau_X \cup \tau_Y$ it follows that $\tau_Y(y_i) = 0$ and thus $\tau'(y'_i) = 1$, hence C' is satisfied by τ' .

It follows that τ' satisfies $F' \cup \{C\}$.

For the “if” direction, assume that C is semantically blocked in F' and let σ_X be an arbitrary assignment over the variables in X . We show that there exists an assignment σ_Y over the variables in Y such that $\sigma_X \cup \sigma_Y$ satisfies F . To this end, we first define the

following assignment over the variables in F' and C :

$$\tau(v) = \begin{cases} \sigma_X(v) & \text{if } v \in X, \\ 0 & \text{otherwise.} \end{cases}$$

Clearly, τ falsifies C and since every $C' \in F'$ contains a literal \bar{l} with $l \in C$, τ satisfies F' . Thus, since C is semantically blocked in F' , there exists a satisfying assignment τ' of $F' \cup \{C\}$ such that $\tau'(v) = \tau(v)$ for all $v \notin \text{var}(C)$. Since no variable from X is contained in C , τ' agrees with σ_X over the variables in X . Now, let $\tau_{X \cup Y}$ be the assignment τ' restricted to the variables in $X \cup Y$. We show that $\tau_{X \cup Y}$ satisfies F .

Let C' be an arbitrary clause in F . By construction, F' contains the clause $t(C')$ which is satisfied by τ' . Clearly, $\neg u \in t(C')$ is falsified by τ' because of the following: The assignment τ' must satisfy some literal $l \in C$. Since $l \in C$, either $l = u$ or F' contains the clause $(\bar{l} \vee u)$. In both cases, since τ' satisfies $F' \cup \{C\}$, τ' must satisfy u . Hence, $\neg u$ is falsified by τ' and thus τ' must satisfy some literal l' in $t(C')$ which is different from $\neg u$. We proceed by a case distinction.

CASE 1: $\text{var}(l') \in X \cup Y$. Then, by the definition of $t(C')$, $l' \in C'$ and since $\tau_{X \cup Y}$ agrees with τ' over $X \cup Y$, $l' \in C'$ is satisfied by $\tau_{X \cup Y}$.

CASE 2: $\text{var}(l') \notin X \cup Y$. In this case, l' is of the form y'_i . Since τ' satisfies y'_i as well as the clause $(\neg y_i \vee \neg y'_i) \in F'$, it follows that τ' satisfies $\neg y_i$. Now, since $t(C')$ was obtained from C' by adding $\neg u$ and replacing every negative literal $\neg y_i$ by a literal y'_i , we get that $\neg y_i \in C'$ is satisfied by τ' and since $\tau_{X \cup Y}$ agrees with τ' over $X \cup Y$, C' is satisfied by $\tau_{X \cup Y}$.

Hence, $\tau_{X \cup Y}$ satisfies F and thus ϕ is true. \square

We have already seen that the set-blocking problem is NP-complete in the general case. However, we obtain a restricted variant of set-blocking by only allowing blocking sets whose size is bounded by a constant. Then, the resulting problem of testing whether a clause C is blocked by some non-empty set $L \subseteq C$, whose size is at most k for $k \in \mathbb{N}^+$, turns out to be polynomial: For a finite set C and $k \in \mathbb{N}^+$, there are only polynomially many non-empty subsets $L \subseteq C$ with $|L| \leq k$. To see this, observe (by basic combinatorics) that the exact number of such subsets is given by the following sum which reduces to a polynomial with degree at most k :

$$\sum_{i=1}^k \binom{|C|}{i}.$$

Hence, the number of non-empty subsets $L \subseteq C$ with $|L| \leq k$ is polynomial in the size of C . This line of argumentation is actually very common. For the sake of completeness, however, we provide the following example:

Example 6.5. Let $|C| = n$ and $k = 3$ (with $k \leq n$). Then, the number of non-empty subsets $L \subseteq C$ with $|L| \leq k$ is given by the polynomial

$$\sum_{i=1}^3 \binom{n}{i} = \frac{n(n-1)(n-2)}{6} + \frac{n(n-1)}{2} + n = \frac{1}{6}n^3 + \frac{5}{6}n$$

of degree $k = 3$.

As there are only polynomially many potential blocking sets and since it can be checked in polynomial time whether a given set $L \subseteq C$ blocks C in F (as argued in the proof of

Theorem 6.2), it can be checked in polynomial time whether for some clause C there exists a blocking set L of size at most k .

Since the definition of super-blocking is based on the definition of set-blocking, we can also consider the complexity of restricted versions of super-blocking where the size of the according blocking sets is bounded by a constant. We thus define an infinite number of decision problems (one for every $k \in \mathbb{N}^+$) as follows:

Definition 6.6. For any $k \in \mathbb{N}^+$, the k -super-blocking problem is the following decision problem: Given a pair (F, C) , where F is a set of clauses and C a clause such that every clause $D \in F$ contains a literal \bar{l} with $l \in C$, decide if for every assignment τ over the external variables $ext_F(C)$, there exists a non-empty set $L \subseteq C$ with $|L| \leq k$ that blocks C in $F|\tau$.

Theorem 6.7. *The k -super-blocking problem is in co-NP for all $k \in \mathbb{N}^+$.*

Proof. Consider the statement that has to be tested for the complement of the k -super-blocking problem:

There exists an assignment τ over the external variables, $ext_F(C)$, such that no non-empty subset of C with $|C| \leq k$ blocks C in $F|\tau$.

Since it can be checked in polynomial time whether a given set $L \subseteq C$ blocks C in $F|\tau$, the following is an NP-procedure:

Guess an assignment τ over the external variables, $ext_F(C)$, and check for every non-empty subset of C (with $|C| \leq k$) whether it blocks C in $F|\tau$. If there is one, return *no*, otherwise return *yes*.

Hence, for every integer $k \in \mathbb{N}^+$, the k -super-blocking problem is in co-NP. □

Hardness for the complexity class co-NP can be shown already for $k = 1$:

Theorem 6.8. *The 1-super-blocking problem is co-NP-hard.*

Proof. We show the result by providing a reduction from the unsatisfiability problem of propositional logic. Let $F = \{C_1, \dots, C_n\}$ be a formula in CNF and define the reduction function

$$f(F) = (F', C), \text{ with } C = (u_1 \vee \dots \vee u_n),$$

where u_1, \dots, u_n are new variables that do not occur in F , and $F' = \bigcup_{i=1}^n F_i$ with $F_i = \{(\neg u_i \vee \bar{l}) \mid l \in C_i\}$. Clearly, (F', C) is a valid instance of the 1-super-blocking problem and $var(F) = ext_{F'}(C)$. We show that F is unsatisfiable if and only if, for every assignment τ over $ext_{F'}(C)$, there exists a literal $u_i \in C$ such that $\{u_i\}$ set-blocks C in $F'|\tau$.

For the “only if” direction, assume that F is unsatisfiable and let τ be an assignment over $ext_{F'}(C)$. Since $var(F) = ext_{F'}(C)$ it follows that there exists a clause C_i in F that is falsified by τ . But then, since every clause in F_i contains a literal \bar{l} with $l \in C_i$, it follows that F_i is satisfied by τ . Hence, $F_i \cap F'|\tau = \emptyset$ and thus, since $\neg u_i$ only occurs in F_i , $\{u_i\}$ trivially set-blocks C in F' .

For the “if” direction, assume that for every τ over $ext_{F'}(C)$, there exists a literal $u_i \in C$ such that $\{u_i\}$ set-blocks C in $F'|\tau$. Since $var(F) = ext_{F'}(C)$ it follows that for every assignment τ of F and every clause $(\neg u_i \vee \bar{l}) \in F'|\tau$ (with $l \in C_i$), $T = (C \setminus \{u_i\}) \cup \{\neg u_i\} \cup \{\neg u_i, \bar{l}\}$ is a tautology. But since T cannot contain complementary literals it must be the

case that $(\neg u_i \vee \bar{l}) \notin F'|\tau$, which implies that every $l \in C_i$ is falsified by τ . It follows that F is unsatisfiable. \square

The above reduction actually works for all k -super-blocking-problems with $k \in \mathbb{N}^+$. To see this, observe that for every $k \in \mathbb{N}^+$, C is k -super-blocked in F' if and only if it is 1-super-blocked in F' : If a clause is 1-super-blocked in a formula, then it is by definition also k -super-blocked for all $k \in \mathbb{N}^+$. Conversely, due to the way F' is constructed, if a set $L \subseteq C$ blocks C in $F'|\tau$, with τ being an arbitrary assignment over $\text{ext}_{F'}(C)$, then there exists a singleton set $L' \subseteq L$ that blocks C in $F'|\tau$ and thus C is 1-super-blocked in F' . We thus get:

Corollary 6.9. *The k -super-blocking problem is co-NP-complete for all $k \in \mathbb{N}^+$.*

The notions of set-blocking and super-blocking, together with the corresponding restrictions discussed in this section, give rise to a whole family of blocking notions which differ in both generality and complexity. We conclude the following:

- (i) Considering the assignments over external variables (as is the case for super-blocking) leads to co-NP-hardness.
- (ii) If blocking sets of arbitrary size are considered, the (sub-)problem of checking whether there exists a blocking set is NP-hard.
- (iii) If the size of blocking sets is bounded by a constant k , the (sub-)problem of testing whether there exists a blocking set turns out to be polynomial.
- (iv) The problem of testing whether a clause is super-blocked in the most general sense, where the size of blocking sets is not bounded by a constant, is Π_2^P -complete.

Hence, we can summarize the following complexity results:

	$ L $ is unrestricted	$ L \leq k$ for $k \in \mathbb{N}^+$
Super-Blocking	Π_2^P -complete	co-NP-complete
Set-Blocking	NP-complete	P

Note that the cardinality $|L|$ of blocking sets is of course bounded by the length of the clauses, thus we can restrict $|L| \leq |C|$. This is particularly interesting for formula instances with (uniform) constant or maximal clause length.

Finally, we conclude the discussion by returning to the starting point of this paper: literal-blocked clauses. Obviously, we can write the definition for set-blocking with $|L| \leq 1$ as follows: A set $\{l\} \subseteq C$ blocks a clause C in a formula F if for each clause $D \in F$ with $\bar{l} \in D$, $(C \setminus \{l\}) \cup D$ is a tautology. (Note that we write $(C \setminus \{l\}) \cup D$ instead of $(C \setminus \{l\}) \cup \{\bar{l}\} \cup D$ since \bar{l} is anyhow required to be contained in D .) This is very similar to the original definition of literal-blocked clauses which requires $C \cup (D \setminus \{l\})$ to be a tautology.

7. COMPARISON WITH OTHER REDUNDANCY PROPERTIES

In the following, we consider several local and non-local redundancy properties as presented by Järvisalo et al. [18] and relate them to the previously discussed local redundancy properties. From the three basic redundancy properties of *tautologies* (T), *subsumed clauses* (S), and *literal-blocked clauses* (BC), extended redundancy properties are derived by *asymmetric-literal addition* and/or a “resolution-look-ahead step”. We start with asymmetric-literal addition:

Definition 7.1. A literal l is an *asymmetric literal* with respect to a clause C in a formula F if $F \setminus \{C\}$ contains a clause $D \vee \bar{l}$ such that $D \subseteq C$.

The addition of an asymmetric literal l to a clause C preserves equivalence in the sense that $F \setminus \{C\} \models (C \equiv C \vee l)$.

Example 7.2. Consider the formula $F = \{(a \vee b), (b \vee c), (\neg c \vee d), (a \vee \neg c \vee \neg d)\}$ and let $C = (a \vee b)$. The literal $\neg c$ is an asymmetric literal with respect to C in F because for the subclause (b) of $(b \vee c)$ it holds that $(b) \subseteq (a \vee b)$. Therefore, the addition of $\neg c$ to C to preserves equivalence and we obtain the clause $C_1 = (a \vee b \vee \neg c)$. After this, $\neg d$ becomes an asymmetric literal with respect to C_1 because of $(\neg c \vee d)$. Adding $\neg d$ to C_1 yields $C_2 = (a \vee b \vee \neg c \vee \neg d)$. Now, $\neg a$ becomes an asymmetric literal with respect to C_2 because of $(a \vee \neg c \vee \neg d)$ and so we add it to obtain the tautology $C_3 = (a \vee \neg a \vee b \vee \neg c \vee \neg d)$.

In the example above, the addition of asymmetric literals turned the clause C into a tautology. Since tautologies are redundant and since asymmetric-literal addition preserves equivalence, we can infer that C is redundant with respect to F . This leads to the notion of an asymmetric tautology:

Definition 7.3. A clause C is an *asymmetric tautology* in a formula F if there exists a sequence l_1, \dots, l_n of literals such that $C \vee l_1 \vee \dots \vee l_n$ is a tautology and l_i is an asymmetric literal with respect to $C \vee l_1 \vee \dots \vee l_{i-1}$ in F for each $i \in 1, \dots, n$. By AT we denote the redundancy property $\{(F, C) \mid C \text{ is an asymmetric tautology in } F\}$.

The notions of *asymmetric subsumed clauses* (AS) and *asymmetric blocked clauses* (ABC) are defined analogously: A clause is an asymmetric subsumed clause if the addition of asymmetric literals turns it into a subsumed clause; it is an asymmetric blocked clause if the addition of asymmetric literals turns it into a blocked clause.

Asymmetric tautologies are particularly popular because they coincide with so-called RUP clauses (RUP stands for *reverse unit propagation*) [28]. It follows for instance from results by Beame, Kautz, and Sabharwal [3] that the conflict clauses computed by conflict-driven-clause-learning SAT solvers are RUP clauses, or, equivalently, asymmetric tautologies. Moreover, variants of asymmetric-literal addition and its converse, asymmetric-literal elimination, are used for minimizing clauses during SAT solving [31, 25, 13, 23, 8]. For instance, Luo et al. [23] minimize a given learned clause C by checking if there exists a subclause of C that is an asymmetric tautology; if so, they replace the original clause by its (stronger) subclause.

Finally, we introduce redundancy properties that are intuitively obtained from existing ones by performing an additional “resolution-look-ahead step”. Their names are obtained by adding the prefix R to the abbreviated name of the redundancy property they extend [18]. More formally: Given a redundancy property \mathcal{P} , the pair (F, C) is contained in $R\mathcal{P}$ if either

- (i) $(F, C) \in \mathcal{P}$, or
- (ii) C contains a literal l such that for each clause $D \in F_{\bar{l}}$, $(F, C \cup (D \setminus \{\bar{l}\})) \in \mathcal{P}$.

Examples are RT (*resolution tautologies*), RS (*resolution-subsumed clauses*), and RAT (*resolution asymmetric tautologies*). Observe that resolution tautologies are nothing else than literal-blocked clauses. Especially RAT [10, 18] is well-known: As almost all modern SAT solving techniques—including preprocessing, inprocessing, and clause learning—can be simulated by the addition and elimination of resolution asymmetric tautologies, they provide the basis for the DRAT proof system [30], which is the standard in practical SAT solving.

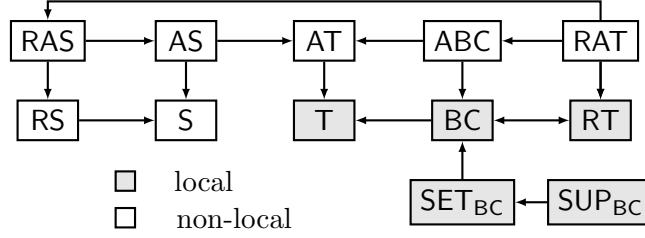


Figure 2: Hierarchy of redundancy properties [18] extended with novel local redundancies. An arrow from \mathcal{P}_1 to \mathcal{P}_2 denotes that \mathcal{P}_1 is more general than \mathcal{P}_2 .

The mentioned redundancy properties lead to the hierarchy depicted in Figure 2. We next show that our new redundancy properties are incomparable with redundancy properties based on T; showing incomparability with subsumption-based properties works analogously.

Proposition 7.4. $AT \not\subseteq SET_{BC}$ and $SET_{BC} \not\subseteq AT$.

Proof. Let $C = (a \vee b \vee c)$ and $F = \{(\neg a \vee x), (\neg b \vee x), (\neg c \vee x), (a \vee b)\}$. Because of the clause $(a \vee b)$, the literal $\neg b$ is an asymmetric literal with respect to C and thus it follows that $(F, C) \in AT$. Now, assume that C is set-blocked by some non-empty set $L \subseteq C$ in F , i.e., for every clause $D \in F_{\bar{L}}$, $(C \setminus L) \cup \bar{L} \cup D$ is a tautology. Since L is non-empty, at least one of $(\neg a \vee x)$, $(\neg b \vee x)$, and $(\neg c \vee x)$ must be contained in $F_{\bar{L}}$. Assume without loss of generality that $D = (\neg a \vee x)$ is contained in $F_{\bar{L}}$. Then, $a \in L$ and thus $a \notin C \setminus L$. Moreover, a is not contained in \bar{L} . Hence, $(C \setminus L) \cup \bar{L} \cup D$ is not a tautology and thus C is not set-blocked by L in F , a contradiction. We conclude that $(F, C) \notin SET_{BC}$.

To see that there exist set-blocked clauses that are not asymmetric tautologies, let $F = \emptyset$ and let $C = (a)$. Clearly, $(F, C) \in SET_{BC}$, but $(F, C) \notin AT$. \square

Proposition 7.5. $AT \not\subseteq SUP_{BC}$ and $SUP_{BC} \not\subseteq AT$.

Proof. Consider again $C = (a \vee b \vee c)$ and $F = \{(\neg a \vee x), (\neg b \vee x), (\neg c \vee x), (a \vee b)\}$ from the proof of Proposition 7.4, and observe that $ext_F(C) = \{x\}$. Here, for the assignment τ that falsifies the external variable x , $F|\tau = F$ and since C is not set-blocked in F (as shown in the proof of Proposition 7.4), it is not set-blocked in $F|\tau$, hence $(F, C) \notin SUP_{BC}$.

To see that $SUP_{BC} \not\subseteq AT$, let $F = \emptyset$ and $C = (a)$. Then, since $(F, C) \in SET_{BC}$ and $SET_{BC} \subset SUP_{BC}$, we get that $(F, C) \in SUP_{BC}$ but $(F, C) \notin AT$. \square

From Proposition 7.5 together with the fact that $AT \subset RAT$ we get:

Corollary 7.6. $RAT \not\subseteq SUP_{BC}$.

Proposition 7.7. $SET_{BC} \not\subseteq RAT$.

Proof. Consider the clause $C = (a \vee b)$ and the formula $F = \{(a \vee b), (\neg a \vee b), (a \vee \neg b)\}$. Clearly, C is set-blocked by $L = \{a, b\}$ in F and thus $(F, C) \in SET_{BC}$.

Now, for the literal a there is only the clause $D_1 = (\neg a \vee b)$ that contains $\neg a$ and $C \cup D_1 \setminus \{\neg a\} = (a \vee b)$. Furthermore, for the literal b there is only the clause $D_2 = (a \vee \neg b)$ that contains $\neg b$ and here we again get that $C \cup D_2 \setminus \{\neg b\} = (a \vee b)$. Since $(a \vee b)$ is not an asymmetric tautology in F , $(F, C) \notin RAT$. \square

Corollary 7.8. RAT is incomparable with both SET_{BC} and SUP_{BC} .

An intuitive explanation for the incomparability of SET_{BC} and SUP_{BC} with both AT and RAT is the following: On the one hand, SET_{BC} and SUP_{BC} have the advantage that they can flip the truth values of multiple literals when trying to determine a clause’s redundancy, whereas AT and RAT can only flip the truth value of a single literal. On the other hand, SET_{BC} and SUP_{BC} can only use information from the resolution environment of a clause while asymmetric-literal addition allows AT and RAT to also use information from outside the resolution environment.

8. CONCLUSION

We showed that there exist redundancy properties that are more general than blocked clauses while still being local, meaning that they can be checked by considering only the resolution environment of a clause. This locality aspect is part of the reason why blocked clauses have been successful in the past and it is particularly appealing in the context of real-world verification where problem encodings into SAT often lead to very large formulas in which the resolution environments of clauses are still small.

By introducing a semantic notion of blocking, we characterized the essence of local redundancy: It suffices to flip only the truth values of some of the clause’s literals to turn satisfying assignments of its resolution environment into assignments that satisfy also the clause itself. With the aim of bringing this semantic blocking notion closer towards practical SAT solving, we then introduced the syntax-based notions of set-blocking and super-blocking. The notion of set-blocking strictly generalizes the traditional notion of blocking by allowing to flip the truth values of more than only one literal. Super-blocked clauses are even more general because the assignments on external variables are also taken into account when deciding redundancy. For super-blocked clauses, we proved that they coincide with semantically blocked clauses.

In addition, we gave an alternative characterization of semantically blocked clauses based on variable elimination. This characterization helps to clarify the relationship between traditionally blocked clauses and semantically blocked clauses: Traditional blocking requires that already the elimination of a single variable from a clause and its resolution environment yields the empty formula. In comparison, semantic blocking is more general as it allows for the elimination of several variables in order to derive the empty formula. Our complexity analysis showed that checking the newly introduced redundancy properties is computationally expensive in the worst case. At first glance, this seems to limit their practical applicability. However, we presented bounded variants that can be checked more efficiently and we expect them to improve the solving process considerably when added to our SAT solvers.

While the focus of this paper lies on the theoretical investigation of local redundancy properties, thereby contributing to gaining a deeper understanding of blocked clauses, a practical evaluation is subject to future work. Another direction for future work is lifting the new redundancy properties to QSAT, the satisfiability problem of quantified Boolean formulas (QBF). There, blocked clauses have shown to be practically even more effective than in SAT solving [11, 22] and we expect this to also hold for quantified variants of set-blocked clauses and super-blocked clauses.

REFERENCES

- [1] Tomas Balyo, Andreas Fröhlich, Marijn Heule, and Armin Biere. Everything you always wanted to know about blocked sets (but were afraid to ask). In Carsten Sinz and Uwe Egly, editors, *Proc. of the 17th Int. Conference on Theory and Applications of Satisfiability Testing (SAT 2014)*, volume 8561 of *LNCS*, pages 317–332, Cham, 2014. Springer.
- [2] Clark W. Barrett, Roberto Sebastiani, Sanjit A. Seshia, and Cesare Tinelli. Satisfiability modulo theories. In *Handbook of Satisfiability*, pages 825–885. IOS Press, 2009.
- [3] Paul Beame, Henry A. Kautz, and Ashish Sabharwal. Towards understanding and harnessing the potential of clause learning. *Journal of Artificial Intelligence Research*, 22:319–351, 2004.
- [4] Armin Biere, Marijn Heule, Hans van Maaren, and Toby Walsh, editors. *Handbook of Satisfiability*. IOS Press, 2009.
- [5] Jingchao Chen. Fast blocked clause decomposition with high quality. *CoRR*, abs/1507.00459, 2015.
- [6] Martin Davis and Hilary Putnam. A computing procedure for quantification theory. *Journal of the ACM*, 7(3):201–215, 1960.
- [7] Niklas Eén and Armin Biere. Effective preprocessing in SAT through variable and clause elimination. In *Proc. of the 8th Int. Conference on Theory and Applications of Satisfiability Testing (SAT 2005)*, volume 3569 of *LNCS*, pages 61–75. Springer, 2005.
- [8] Hyojung Han and Fabio Somenzi. Alembic: An efficient algorithm for CNF preprocessing. In *Proc. of the 44th Design Automation Conference (DAC 2007)*, pages 582–587. IEEE, 2007.
- [9] Marijn Heule and Armin Biere. Blocked clause decomposition. In Ken McMillan, Aart Middeldorp, and Andrei Voronkov, editors, *Proc. of the 19th Int. Conference on Logic for Programming, Artificial Intelligence, and Reasoning (LPAR-19)*, volume 8312 of *LNCS*, pages 423–438, Heidelberg, 2013. Springer.
- [10] Marijn Heule, Warren A. Hunt, Jr., and Nathan Wetzler. Verifying refutations with extended resolution. In Maria Paola Bonacina, editor, *Proc. of the 24th Int. Conference on Automated Deduction (CADE 2013)*, volume 7898 of *LNCS*, pages 345–359, Heidelberg, 2013. Springer.
- [11] Marijn Heule, Matti Järvisalo, Florian Lonsing, Martina Seidl, and Armin Biere. Clause elimination for SAT and QSAT. *Journal of Artificial Intelligence Research*, 53:127–168, 2015.
- [12] Marijn J. H. Heule, Martina Seidl, and Armin Biere. A unified proof system for QBF preprocessing. In *Proc. of the 7th Int. Joint Conference on Automated Reasoning (IJCAR 2014)*, volume 8562 of *LNCS*, pages 91–106. Springer, 2014.
- [13] Marijn J.H. Heule, Matti Järvisalo, and Armin Biere. Efficient CNF simplification based on binary implication graphs. In Karem A. Sakallah and Laurent Simon, editors, *Proc. of the 14th Int. Conference on Theory and Applications of Satisfiability Testing (SAT 2011)*, volume 6695 of *LNCS*, pages 201–215, Heidelberg, 2011. Springer.
- [14] Markus Iser, Norbert Manthey, and Carsten Sinz. Recognition of nested gates in CNF formulas. In Marijn Heule and Sean Weaver, editors, *Proc. of the 18th Int. Conference on Theory and Applications of Satisfiability Testing (SAT 2015)*, volume 9340 of *LNCS*, pages 255–271, Cham, 2015. Springer.
- [15] Mikolás Janota, William Klieber, Joao Marques-Silva, and Edmund M. Clarke. Solving QBF with counterexample guided refinement. *Artificial Intelligence*, 234:1–25, 2016.
- [16] Matti Järvisalo, Armin Biere, and Marijn Heule. Blocked clause elimination. In Javier Esparza and Rupak Majumdar, editors, *Proc. of the 16th Int. Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS 2010)*, volume 6015 of *LNCS*, pages 129–144, Heidelberg, 2010. Springer.
- [17] Matti Järvisalo, Armin Biere, and Marijn Heule. Simulating circuit-level simplifications on CNF. *Journal on Automated Reasoning*, 49(4):583–619, 2012.
- [18] Matti Järvisalo, Marijn Heule, and Armin Biere. Inprocessing rules. In Bernhard Gramlich, Dale Miller, and Uli Sattler, editors, *Proc. of the 6th Int. Joint Conference on Automated Reasoning (IJCAR 2012)*, volume 7364 of *LNCS*, pages 355–370, Heidelberg, 2012. Springer.
- [19] Benjamin Kiesl, Martina Seidl, Hans Tompits, and Armin Biere. Super-blocked clauses. In Nicola Olivetti and Ashish Tiwari, editors, *Proc. of the 8th Int. Joint Conference on Automated Reasoning (IJCAR 2016)*, volume 9706 of *LNCS*, pages 45–61, Cham, 2016. Springer.
- [20] Hans Kleine Büning and Uwe Bubeck. Theory of quantified boolean formulas. In *Handbook of Satisfiability*, pages 735–760. IOS Press, 2009.
- [21] Oliver Kullmann. On a generalization of extended resolution. *Discrete Applied Mathematics*, 96-97:149–176, 1999.

- [22] Florian Lonsing, Fahiem Bacchus, Armin Biere, Uwe Egly, and Martina Seidl. Enhancing search-based QBF solving by dynamic blocked clause elimination. In Martin Davis, Ansgar Fehnker, Annabelle McIver, and Andrei Voronkov, editors, *Proc. of the 20th Int. Conference on Logic for Programming, Artificial Intelligence (LPAR-20)*, volume 9450 of *LNCS*, pages 418–433, Heidelberg, 2015. Springer.
- [23] Mao Luo, Chu-Min Li, Fan Xiao, Felip Manyà, and Zhipeng Lü. An effective learnt clause minimization approach for CDCL SAT solvers. In Carles Sierra, editor, *Proc. of the 26th Int. Joint Conference on Artificial Intelligence (IJCAI 2017)*, pages 703–711. ijcai.org, 2017.
- [24] Norbert Manthey, Tobias Philipp, and Christoph Wernhard. Soundness of inprocessing in clause sharing SAT solvers. In Matti Järvisalo and Allen Van Gelder, editors, *Proc. of the 16th Int. Conference on Theory and Applications of Satisfiability Testing (SAT 2013)*, volume 7962 of *LNCS*, pages 22–39, Heidelberg, 2013. Springer.
- [25] Cédric Piette, Youssef Hamadi, and Lakhdar Sais. Vivifying propositional clausal formulae. In Malik Ghallab, Constantine D. Spyropoulos, Nikos Fakotakis, and Nikolaos M. Avouris, editors, *Proc. of the 18th European Conference on Artificial Intelligence (ECAI 2008)*, volume 178 of *Frontiers in Artificial Intelligence and Applications*, pages 525–529. IOS Press, 2008.
- [26] Giles Reger, Martin Suda, and Andrei Voronkov. Playing with AVATAR. In P. Amy Felty and Aart Middeldorp, editors, *Proc. of the 25th Int. Conference on Automated Deduction (CADE 2015)*, volume 9195 of *LNCS*, pages 399–415, Cham, 2015. Springer.
- [27] Horst Samulowitz and Fahiem Bacchus. Using SAT in QBF. In Peter van Beek, editor, *Proc. of the 11th Int. Conference on Principles and Practice of Constraint Programming (CP 2005)*, volume 3709 of *LNCS*, pages 578–592, Heidelberg, 2005. Springer.
- [28] Allen Van Gelder. Verifying RUP proofs of propositional unsatisfiability. In *Proc. of the 10th Int. Symposium on Artificial Intelligence and Mathematics (ISAIM 2008)*, 2008.
- [29] Y. Vitzel, G. Weissenbacher, and S. Malik. Boolean satisfiability solvers and their applications in model checking. *Proc. of the IEEE*, 103(11):2021–2035, 2015.
- [30] Nathan D. Wetzler, Marijn J. H. Heule, and Warren A. Hunt Jr. DRAT-trim: Efficient checking and trimming using expressive clausal proofs. In Carsten Sinz and Uwe Egly, editors, *Proc. of the 17th Int. Conference on Theory and Applications of Satisfiability Testing (SAT 2014)*, volume 8561 of *LNCS*, pages 422–429, Cham, 2014. Springer.
- [31] Siert Wieringa and Keijo Heljanko. Concurrent clause strengthening. In Matti Järvisalo and Allen Van Gelder, editors, *Proc. of the 16th Int. Conference on Theory and Applications of Satisfiability Testing (SAT 2013)*, volume 7962 of *LNCS*, pages 116–132, Heidelberg, 2013. Springer.