Super-Blocked Clauses*

Benjamin Kiesl[1], Martina Seidl[2], Hans Tompits[1], Armin Biere[2]

[1] Institute for Information Systems, Vienna University of Technology, Austria
[2] Institute for Formal Models and Verification, JKU Linz, Austria

Abstract. In theory and practice of modern SAT solving, clause-elimination procedures are essential for simplifying formulas in conjunctive normal form (CNF). Such procedures identify redundant clauses and faithfully remove them, either before solving in a preprocessing phase or during solving, resulting in a considerable speed up of the SAT solver. A wide number of effective clause-elimination procedures is based on the clause-redundancy property called *blocked clauses*. For checking if a clause C is blocked in a formula F, only those clauses of F that are resolvable with C have to be considered. Hence, the blocked-clauses redundancy property can be said to be *local*. In this paper, we argue that the established definitions of blocked clauses are not in their most general form. We introduce more powerful generalizations, called *set-blocked clauses* and *super-blocked clauses*, respectively. Both can still be checked locally, and for the latter it can even be shown that it is the most general local redundancy property. Furthermore, we relate these new notions to existing clause-redundancy properties and give a detailed complexity analysis.

1 Introduction

Over the last two decades, we have seen enormous progress in the performance of SAT solvers, i.e., tools for solving the satisfiability problem of propositional logic (SAT) [1]. As a consequence, SAT solvers have become attractive reasoning engines in many user domains like the verification of hardware and software [2] as well as in the backends of other reasoning tools like SMT solvers [3] or even first-order theorem provers [4]. In such applications, however, SAT solvers often reach their limits, motivating the quest for more efficient SAT techniques.

Clause-elimination procedures which simplify formulas in conjunctive normal form (CNF) play a crucial role regarding the performance of modern SAT solvers [5–12]. Either before solving ("preprocessing") or during solving ("inprocessing"), such procedures identify redundant clauses and remove them without changing the satisfiability or unsatisfiability of the formula [6, 7].

An important redundancy property is that of *blocked clauses* [13, 14]. Informally, a clause C is blocked in a CNF-formula F if it contains a literal l such that all possible resolvents of C on l with clauses from F are tautologies. As

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

only the resolution environment of a clause C and not the whole formula F has to be considered to check whether C is blocked, the blocked-clauses condition is said to be a *local* redundancy property.

Blocked clauses have not only shown to be important for speeding up the solving process [8, 14], but they also yield the basis for blocked-clause decomposition which splits a CNF into two parts such that blocked-clause elimination can solve it. Blocked-clause decomposition [9] is successfully used for gate extraction, for efficiently finding backbone variables, and for the detection of implied binary equivalences [10, 11]. The winner of the SATRace 2015 competition, the solver abcdSAT [12], uses blocked-clause decomposition as core technology.

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 the established definitions of local clause redundancy properties like blocked clauses are not in their most general form and introduce more powerful generalizations, called *set-blocked clauses* and *super-blocked clauses*. Both can still be checked locally and for the latter we show that it is actually the most general local redundancy property. Furthermore, we relate these new notions to existing clause redundancy properties and give a detailed complexity analysis.

Our paper is structured as follows. After introducing the necessary preliminaries in Section 2, we present some observations on blocked clauses in Section 3. In Section 4, we introduce the notion of *semantic blocking* and show that it is the most general local redundancy property. After this, the syntax-based notions of *set-blocking* and *super-blocking* are introduced in Section 5, where we also relate the different redundancy properties to each other and show that super-blocking coincides with semantic blocking. In Section 6, we give a detailed complexity analysis and in Section 7, we outline the relationship to existing redundancy properties before concluding with an outlook to future work in Section 8.

2 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. A clause can be seen as a set of literals and a formula as 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 F, $var(F)$ denotes the variables in F. For convenience, we treat $var(F)$ as a variable if F is a literal, and as a set of variables otherwise.

An *assignment* over a set V of variables is a function that assigns to every variable in V either 1 or 0. If for an assignment τ and a formula F, the domain of τ coincides with $var(F)$, then τ is said to be an assignment *of* F. Given an assignment τ and a literal l, τ_l is the assignment obtained from τ by interchanging ("flipping") the truth value of l, i.e., by defining $\tau_l(v) = 1 - \tau(v)$ if $v = var(l)$ and $\tau_l(v) = \tau(v)$ otherwise.

A literal l is *satisfied* by an assignment τ if l is positive and $\tau(var(l)) = 1$ or if it is negative and $\tau(var(l)) = 0$. 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 assignments. Two formulas F and F' are *satisfiability equivalent* if F is satisfiable if and only if F' is satisfiable.

Given two clauses C_1 and C_2 with literal $l \in C_1$ and $\bar{l} \in C_2$, the clause $C = (C_1 \setminus \{l\}) \cup (C_2 \setminus \{\bar{l}\})$ is called the *resolvent* of C_1 and C_2 on l. Given a formula F and a clause C, the *resolution environment*, $env_F(C)$, of C in F is the set of all clauses in F that can be resolved with C:

$$env_F(C) = \{C' \in F \mid \exists l \in C' \text{ such that } \bar{l} \in C\}.$$

The variables in $var(C)$ are referred to as *local variables* and the variables in $var(env_F(C)) \setminus var(C)$ are the *external variables*, denoted by $ext_F(C)$.

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

Definition 1. *A clause C is* redundant *w.r.t. 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 w.r.t. F. Finally, for two redundancy properties $\mathcal{P}_1$ and $\mathcal{P}_2$, $\mathcal{P}_1$ is* more general *than $\mathcal{P}_2$ if $\mathcal{P}_2 \subseteq \mathcal{P}_1$. Accordingly, $\mathcal{P}_1$ is* strictly more general *than $\mathcal{P}_2$ if $\mathcal{P}_2 \subset \mathcal{P}_1$.*

As an example, consider the formula $F = \{(a \vee b), (\neg a \vee \neg b)\}$. The clause $C = (\neg a \vee \neg b)$ is redundant w.r.t. F since $F \setminus \{C\}$ and $F \cup \{C\}$ are satisfiability equivalent (although they are not logically equivalent). Furthermore, 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, $F \setminus \{C\}$ is satisfiability equivalent to $F \cup \{C\}$.

Also note that C is *not* redundant w.r.t. F if and only if $F \setminus \{C\}$ is satisfiable and $F \cup \{C\}$ is unsatisfiable. Redundancy properties as defined above yield not only the basis for clause-elimination but also for clause-addition procedures [7].

3 Observations on Blocked Clauses

In the following, we recapitulate the notion of blocked clauses due to Heule et al. [6] which we will refer to as *literal-blocked clauses* in the rest of the paper. Motivated by the examples given in this section, we will generalize this notion of blocking to more powerful redundancy properties.

Definition 2. *Given a formula F, a clause C, and a literal $l \in C$, l* blocks *C in F if for each clause $C' \in F$ with $\bar{l} \in C'$, $C \cup (C' \setminus \{\bar{l}\})$ is a tautology. A clause C is* literal-blocked *in F if there exists a literal that blocks C in F. By* BC *we denote the set $\{(F, C) \mid C \text{ is literal-blocked in } F\}$.*

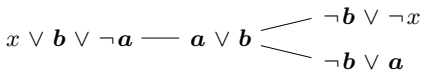

Fig. 1. The clause $(a \vee b)$ from Example 3 and its resolution environment.

Example 1. 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 $C' = (\neg b \vee \neg a)$, and $C \cup (C' \setminus \{\bar{l}\}) = (a \vee b \vee \neg a)$ is a tautology.

Proposition 1. BC *is a redundancy property.*

Proposition 1 paraphrases results from [6] and actually follows from results in this paper (cf. Proposition 6 and Corollary 9). Intuitively, if an assignment τ satisfies $F \setminus \{C\}$ but falsifies C which is blocked by literal l, then τ_l satisfies C. The condition that l blocks C thereby guarantees that τ_l does not falsify any other clauses in F. Hence, τ_l satisfies $F \cup \{C\}$ and thus $F \setminus \{C\}$ and $F \cup \{C\}$ are satisfiability equivalent. Next, we illustrate how a satisfying assignment of $F \cup \{C\}$ can be obtained from one of $F \setminus \{C\}$ [6]. This approach is used when blocked clauses have been removed from a formula during pre- or inprocessing.

Example 2. Consider again the formula $F = \{(\neg a \vee c), (\neg b \vee \neg a)\}$ and the clause $C = (a \vee b)$ from Example 1. We already know that b blocks C in F. So let τ be 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 $\neg a$ is still satisfied by τ_b we get that τ_b satisfies $F \cup \{C\}$. □

Literal-blocked clauses generalize many other redundancy properties like *pure literal* or *tautology* [6]. One of their particularly important properties is that for testing if 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 which can be identified by considering only their resolution environment, but which are not literal-blocked. This is indeed the case:

Example 3. Let $C = (a \vee b)$ and F an arbitrary formula with the resolution environment $env_F(C) = \{(x \vee b \vee \neg a), (\neg b \vee \neg x), (\neg b \vee a)\}$ (see Fig. 1). The clause C is not literal-blocked in F but redundant: Suppose that 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 $\tau' = \tau_a$. Thereby, $\tau'(x) = 1$ guarantees that 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: Then, $\tau''(b) = 1$ 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, $env_F(C)$, we obtain in both cases a satisfying assignment of F. □

4 A Semantic Notion of Blocking

In the examples of the preceding section, when arguing that a clause C is redundant w.r.t. 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 the resolution environment, $env_F(C)$, of 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. *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 $v \notin var(C)$. By $\mathsf{SEM}_{\mathsf{BC}}$ we denote the set $\{(F, C) \mid C$ is semantically blocked in $F\}$.*

Note that clause C in Example 3 is semantically blocked in F. Note also that if the resolution environment, $env_F(C)$, of a clause C is not satisfiable, then C is semantically blocked.

Theorem 2. $\mathsf{SEM}_{\mathsf{BC}}$ *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 that 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 $F \cup \{C\}$ that satisfies l.

Case 2: $var(C) \subseteq var(F \setminus \{C\})$. In this case, τ is an assignment of $F \cup \{C\}$. 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 w.r.t. F and thus $\mathsf{SEM}_{\mathsf{BC}}$ is a redundancy property. □

If a clause C is redundant w.r.t. some formula F and this redundancy can be identified by considering only its resolution environment in F, then we expect C to be redundant w.r.t. 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 4. *A redundancy property $\mathcal{P}$ is* local *if, for any two formulas F, F' and every clause C with $env_F(C) = env_{F'}(C)$, either $\{(F, C), (F', C)\} \subseteq \mathcal{P}$ or $\{(F, C), (F', C)\} \cap \mathcal{P} = \emptyset$.*

Theorem 3. $\mathsf{SEM}_{\mathsf{BC}}$ *is a local redundancy property.*

Preparatory for showing that $\mathsf{SEM}_{\mathsf{BC}}$ is actually the most general local redundancy property (cf. Theorem 5 below), we first prove the following lemma.

Lemma 4. *Let F be a formula and C a clause that is not semantically blocked in F. Then, there exists a formula F' with $env_{F'}(C) = env_F(C)$ such that C is not redundant w.r.t. 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:

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

We furthermore define $F' = env_F(C) \cup \{C\} \cup T$. Clearly, since C can be falsified and since the clauses in T contain only literals with variables that do not occur in C, we get that neither C nor any clause of T contains a literal $\bar{l}$ with $l \in C$. We thus have that $env_{F'}(C) = env_F(C)$.

Now observe the following: The assignment τ satisfies $env_F(C)$ and, clearly, also T, hence $F' \setminus \{C\}$ is satisfiable. Furthermore, by the construction of T, every assignment that satisfies F' must agree with τ on all variables $v \notin var(C)$. Now, since there does not exist a satisfying assignment τ' of $env_F(C) \cup \{C\}$ such that $\tau(v) = \tau'(v)$ for all $v \notin var(C)$, it follows that $F' \cup \{C\} = F'$ is unsatisfiable. Therefore, $F' \setminus \{C\}$ and $F' \cup \{C\}$ are not satisfiability equivalent and thus C is not redundant w.r.t. F'. □

Theorem 5. $\mathsf{SEM}_{\mathsf{BC}}$ *is the most general local redundancy property.*

Proof. Suppose there exists a local redundancy property $\mathcal{P}$ that is strictly more general than $\mathsf{SEM}_{\mathsf{BC}}$. Then, there exists some pair (F, C) such that $(F, C) \in \mathcal{P}$ but $(F, C) \notin \mathsf{SEM}_{\mathsf{BC}}$. Now, since $(F, C) \notin \mathsf{SEM}_{\mathsf{BC}}$ it follows by Lemma 4 that there exists a formula F' with $env_{F'}(C) = env_F(C)$ such that C is not redundant w.r.t. 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. □

5 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. 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.

Definition 5. *Let F be a formula and C a clause. A non-empty set $L \subseteq C$* blocks *C in F if, for each clause $C' \in F$ with $C' \cap \bar{L} \neq \emptyset$, $(C \setminus L) \cup \bar{L} \cup C'$ is a tautology. We say that a clause is* set-blocked in *F if there exists a set that blocks it. We write $\mathsf{SET}_{\mathsf{BC}}$ to refer to $\{(F, C) \mid C$ is set-blocked in $F\}$.*

Example 4. Let $C = (a \vee b)$ and $F = \{(\neg a \vee b), (\neg b \vee a)\}$. Then, C is set-blocked by $L = \{a, b\}$ but 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 the literals in L. Since $(C \setminus L) \cup \bar{L} \cup C'$ is a tautology for every C' in the resolution environment of C, it holds that (i) C' itself is a tautology and thus satisfied by τ', or (ii) C' contains a literal of L which is satisfied by τ' since its truth value is flipped, or (iii) C' 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 6. *Set-blocking is strictly more general than literal-blocking, i.e., it holds that* $\mathsf{BC} \subset \mathsf{SET}_{\mathsf{BC}}$.

Proof. Example 4 shows that $\mathsf{BC} \neq \mathsf{SET}_{\mathsf{BC}}$. It remains to show that $\mathsf{BC} \subseteq \mathsf{SET}_{\mathsf{BC}}$. Let F be a formula and C a literal-blocked clause in F. We distinguish two cases:

Case 1: C is a tautology. Then, $l, \bar{l} \in C$ for some literal l. Let $L = \{l, \bar{l}\}$. It follows that $(C \setminus L) \cup \bar{L} \cup C'$ is a tautology for every C' with $C' \cap \bar{L} \neq \emptyset$.

Case 2: C is not a tautology. Since C is literal-blocked, there exists some literal $l \in C$ such that for every clause $C' \in F$ with $\bar{l} \in C'$, $C \cup (C' \setminus \{\bar{l}\})$ is a tautology. Let $L = \{l\}$ and let $C' \in F$ with $C' \cap \bar{L} \neq \emptyset$. Then, as C' contains $\bar{l}$, $C \cup (C' \setminus \{\bar{l}\})$ is a tautology. Since C is not a tautology, C' contains some literal $l' \neq \bar{l}$ such that $\bar{l'} \in C \cup (C' \setminus \{\bar{l}\})$. Now, since $l' \neq \bar{l}$ we have that $\bar{l'} \neq l$ and thus $\bar{l'} \in (C \setminus \{l\}) \cup C'$. But then, $(C \setminus L) \cup \bar{L} \cup C'$ is a tautology.

Thus, C is set-blocked in F and therefore $\mathsf{BC} \subseteq \mathsf{SET}_{\mathsf{BC}}$. □

We already argued slightly informally why set-blocked clauses are redundant. However, the fact that $\mathsf{SET}_{\mathsf{BC}}$ is a redundancy property follows directly from the properties of super-blocked clauses, which we introduce next. In the following, for 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. Recall that the external variables, $ext_F(C)$, are those that are contained in $env_F(C)$ but not in C.

Definition 6. *A clause C is* super-blocked *in a formula F if, for every assignment τ over the external variables, $ext_F(C)$, C is set-blocked in $F|\tau$. We write* $\mathsf{SUP}_{\mathsf{BC}}$ *for the set* $\{(F, C) \mid C \text{ is super-blocked in } F\}$.

For instance, the clause C in Example 3 is not set-blocked but super-blocked in F since it is set-blocked in $F|\tau$ 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, a satisfying assignment τ' of $F \cup \{C\}$ can be obtained by flipping the truth values of certain literals of C. However, for making sure that the flipping does not falsify any clauses C' in the resolution environment of C, also the truth values of literals $l \in C'$ with $var(l) \in ext_F(C)$ are considered. This is in contrast to set-blocking, where only the truth values of literals whose variables are contained in $var(C)$ are considered. Finally, note that if a clause is set-blocked in F, then it is also set-blocked in every $F' \subseteq F$ and thus in every $F|\tau$. Hence we get:

Proposition 7. *Super-blocking is strictly more general than set-blocking, i.e., it holds that* $\mathsf{SET}_{\mathsf{BC}} \subset \mathsf{SUP}_{\mathsf{BC}}$.

Theorem 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* $\mathsf{SUP}_{\mathsf{BC}} = \mathsf{SEM}_{\mathsf{BC}}$.

Proof. For the "only if" direction, let F be a formula, C a clause that is super-blocked in F, and τ a satisfying assignment of $env_F(C)$. If τ satisfies C, or C contains a literal l with $var(l) \notin var(F)$ (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 $var(C) \subseteq var(F)$ and that τ does not satisfy C. Furthermore, let τ_E be obtained from τ by restricting it to the external variables $ext_F(C)$. 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 $l, \bar{l} \in L$, hence τ' is well-defined. Clearly, τ' satisfies C and $\tau'(v) = \tau(v)$ for all $v \notin var(C)$. It remains to show that τ' satisfies $env_F(C)$. Since τ' differs from τ only on the truth values of variables in $var(L)$, τ' can only falsify clauses containing a literal $\bar{l}$ with $l \in L$. Let C' be such a clause. We proceed by a case distinction.

Case 1: C' contains an external literal l (i.e., $var(l) \in ext_F(C)$) that is satisfied by τ. Then, since $var(l) \notin var(C)$ and thus $l \notin L$, it follows that τ' agrees with τ on the truth value of l and thus l is satisfied by τ'.

Case 2: C' does not contain an external literal that is satisfied by τ. In this case, C' is contained in $F|\tau_E$ and thus, since L set-blocks C in $F|\tau_E$, we have that $(C \setminus L) \cup \bar{L} \cup C'$ is a tautology. If C' 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 $l \in C'$ 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 thus l is satisfied by τ'. In both cases τ' satisfies l and thus C'.

For the "if" direction, let F be a formula and C 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 $C' \in env_F(C)$ 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 $C' \in F|\tau_E$ with $C' \cap \bar{L} \neq \emptyset$ such that $(C \setminus L) \cup \bar{L} \cup C'$ is not a tautology and neither τ_E nor τ' satisfy any external literal in C'.

Let $l \in C'$ be a (local) literal with $var(l) \in var(C)$. Since $(C \setminus L) \cup \bar{L} \cup C'$ 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 C' is satisfied by τ' and consequently τ' does not satisfy $C' \in env_F(C)$, which then allows to conclude that C is not semantically blocked in F. □

Corollary 9. $\mathsf{SET}_{\mathsf{BC}}$ *is a (local) redundancy property.*

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 w.r.t. the size of a clause and its resolution environment.

Definition 7. *The* set-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 $l \in C$, is C set-blocked in F?*

Theorem 10. *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 $(C \setminus L) \cup \bar{L} \cup C'$ is a tautology for every C' with $C' \cap \bar{L} \neq \emptyset$. 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 (*proof sketch*): We give a reduction from SAT by defining the following reduction function on input formula F which is w.l.o.g. in CNF:

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

where $var(F) = \{x_1, \ldots, x_n\}$ and $u, x'_1, \ldots, x'_n$ 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 $x_i \in var(F)$.

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' 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 τ of F.

An assignment τ, obtained from a blocking set L by defining $\tau(x_i) = 1$ if $x_i \in L$ and $\tau(x_i) = 0$ otherwise, satisfies F because of the following:

1. Since all $C' = t(D)$ with $D \in F$, as well as C, contain—apart from $\neg u$—only positive literals, $(C \setminus L) \cup \bar{L} \cup C'$ is only a tautology if L contains a literal of C'. Now, the clauses $(\neg x_i \vee u), (\neg x'_i \vee u)$ force u to be contained in L and thus L must contain a literal $l \neq \neg u$ of every $t(D)$ with $D \in F$.
2. The reason why negative literals $\neg x_i$ are replaced by positive literals x'_i is as follows: If C were of the form $(u \vee x_1 \vee \neg x_1 \vee \cdots \vee x_n \vee \neg x_n)$, it would be trivially blocked by every set L containing two complementary literals $x_i, \neg x_i$. Hence, satisfying assignments would not correspond to blocking sets.
3. The clauses $(\neg x_i \vee \neg x'_i)$ guarantee that x_i and x'_i cannot both be contained in L. Since L contains a literal of every $t(D)$, it is thus guaranteed that τ satisfies every $D \in F$: If L contains a positive literal $x_i \in t(D)$, then $x_i \in D$ is satisfied by τ. If L contains a negative literal $x'_i \in t(D)$, then $x_i \notin L$, hence $\tau(x_i) = 0$ and thus $\neg x_i \in D$ is satisfied by τ.

Similarly, one can show that every set L, obtained from a satisfying assignment τ of F by defining $L = \{u\} \cup \{x_i \mid \tau(x_i) = 1\} \cup \{x'_i \mid \tau(x_i) = 0\}$, blocks C in F'. □

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

Definition 8. *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 $l \in C$, is C super-blocked in F?*

Theorem 11. *The super-blocking problem is Π_2^P-complete.*

Proof. Again, we first show Π_2^P-membership followed by Π_2^P-hardness.

Π_2^P-MEMBERSHIP: The following is a Σ_2^P-procedure for testing whether C is *not* super-blocked in F: Guess an assignment τ over the external variables, $ext_F(C)$, and ask an NP-oracle whether C is set-blocked in $F|\tau$. If the oracle answers *no*, then return *yes*, otherwise return *no*.

Π_2^P-HARDNESS (*proof sketch*): We give a reduction from $\forall\exists$-SAT to the super-blocking problem. Let $\phi = \forall X \exists Y F$ be an instance of $\forall\exists$-SAT and assume w.l.o.g. that F is in CNF. We define the reduction function

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

where $Y = \{y_1, \ldots, y_n\}$ and $u, y'_1, \ldots, y'_n$ 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 by
– adding the clauses $(\neg y_i \vee \neg y'_i), (\neg y_i \vee u), (\neg y'_i \vee u)$ for every $y_i \in Y$.

As super-blocking coincides with semantic blocking, we show that ϕ is satisfiable if and only if C is semantically blocked in F'.

The reduction is similar to the one used for proving Theorem 10. Here, however, only the existentially quantified variables of ϕ are encoded into C, hence all $x_i \in X$ are external variables.

For the "only if" direction, we assume that ϕ is satisfiable and that we are given some arbitrary satisfying assignment τ of F'. By restricting τ to the variables in X we can then obtain an assignment σ_X over the variables in X. Since ϕ is satisfiable, there exists an assignment σ_Y over the variables in Y such that $\sigma_X \cup \sigma_Y$ satisfies F. From this we can in turn obtain a satisfying assignment τ' of $F' \cup \{C\}$ by defining $\tau'(x_i) = \sigma_X$ for $x_i \in X$, $\tau'(y_i) = \sigma_Y(y_i)$ and $\tau'(y'_i) = 1 - \sigma_Y(y_i)$ for $y_i \in Y$, and finally $\tau'(u) = 1$. Since τ' differs from τ only on variables in $var(C)$, C is semantically blocked in F'.

Likewise, for showing the "if" direction, we assume that C is semantically blocked in F' and that we are given some arbitrary assignment σ_X over the variables in X. The crucial observation is then that for σ_X we can construct an assignment τ that satisfies F', by defining $\tau(x_i) = \sigma_X(x_i)$ for all $x_i \in X$ and $\tau(v) = 0$ for all $v \in C$. The assignment τ satisfies F' since every $C' \in F'$ contains a literal $\bar{l}$ with $l \in C$. Then, since C is semantically blocked in F', there exists a satisfying assignment τ' of $F' \cup \{C\}$ that corresponds with σ_X over X. Since $(\neg y_i \vee u)$ and $(\neg y'_i \vee u)$ are in F' for every $y_i \in Y$, it is also guaranteed that u must be satisfied by τ' and thus τ' satisfies a literal $l \neq \neg u$ in every $t(D)$ with $D \in F$. Finally, an assignment σ_Y over the variables in Y can be obtained by defining $\sigma_Y(y_i) = 1$ if and only if $\tau'(y_i) = 1$. Then, $\sigma_X \cup \sigma_Y$ is a satisfying assignment of F. □

We have already seen that the set-blocking problem is NP-complete in the general case. However, a restricted variant of set-blocking is obtained 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 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 $\sum_{i=1}^{3} \binom{n}{i} = \frac{1}{6}n^3 + \frac{5}{6}n$ of degree $k = 3$. □

Now, 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 10), 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, one 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 9. *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 $C' \in F$ contains a literal $\bar{l}$ with $l \in C$, does it hold that, 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 12. *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 13. *The 1-super-blocking problem is* co-NP*-hard.*

Proof. By a reduction from the unsatisfiability problem of propositional logic. Let $F = \{C_1, \ldots, C_n\}$ be a formula in CNF and define the reduction function

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

where $u_1, \ldots, u_n$ 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 $u_i \in C$ 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 $u_i \in C$ 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. □

Corollary 14. *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:

	$\lvert L\rvert$ is unrestricted	$\lvert L\rvert \leq k$ 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 $C' \in F$ with $\bar{l} \in C'$, $(C \setminus \{l\}) \cup C'$ is a tautology. (Note that we write $(C \setminus \{l\}) \cup C'$ instead of $(C \setminus \{l\}) \cup \{\bar{l}\} \cup C'$ since $\bar{l}$ is anyhow required to be contained in C'.) This is very similar to the original definition of literal-blocked clauses which requires $C \cup (C' \setminus \{l\})$ to be a tautology.

7 Comparison with Other Redundancy Properties

In the following, we consider several local and non-local redundancy properties as presented in [7] and relate them to the previously discussed local redundancy

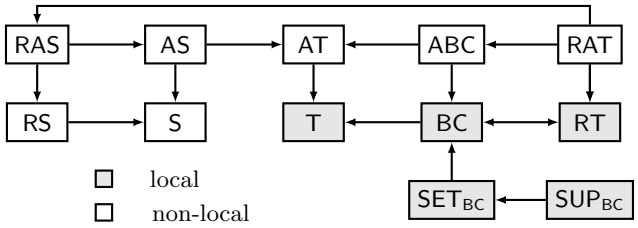

Fig. 2. Hierarchy of redundancy properties [7] extended with novel local redundancies. For redundancy properties $\mathcal{P}_1$ and $\mathcal{P}_2$, an arrow from $\mathcal{P}_1$ to $\mathcal{P}_2$ denotes that $\mathcal{P}_2 \subseteq \mathcal{P}_1$.

properties. From the three basic redundancy properties *tautology* (T), *subsumption* (S), and *literal-blocked clauses* (BC), extended redundancy properties are derived as follows.

Given a formula F and a clause C, $\mathsf{ALA}(F, C)$ is the *unique* clause obtained from C by repeating *asymmetric literal addition*, as defined in the following, until a fixed point is reached: If $l_1, \ldots, l_k \in C$ and there is a clause $(l_1 \vee \cdots \vee l_k \vee l) \in F \setminus \{C\}$ for some literal l, let $C := C \cup \{\bar{l}\}$. The special case where $k = 1$ is called *hidden literal addition* (HLA). Due to space limitations, we will not consider HLA separately. Given a formula F and a clause C, $(F, C) \in \mathsf{AT}$ (resp., AS or ABC) if $(F, \mathsf{ALA}(F, C)) \in \mathsf{T}$ (resp., S or BC).

Finally, we introduce the redundancy properties prefixed with R [7]. Given a formula F and a clause C, $(F, C) \in \mathsf{R}\mathcal{P}$ if either (i) $(F, C) \in \mathcal{P}$ or (ii) there is a literal l in C such that for each clause $C' \in F$ with $\bar{l} \in C'$, $(F, C \cup C' \setminus \{\bar{l}\}) \in \mathcal{P}$. Examples are RT, RS, and RAT. Especially RAT is extremely powerful, because it captures all known SAT solving techniques including preprocessing, inprocessing, and clause learning [7, 15].

These notions of redundancy lead to the hierarchy depicted in Figure 2 which we extend with the previously introduced set-blocked and super-blocked clauses. We discuss the incomparability with redundancy properties based on T in detail; incomparability with subsumption-based properties works analogously.

Proposition 15. $\mathsf{AT} \not\subseteq \mathsf{SET}_{\mathsf{BC}}$ *and* $\mathsf{SET}_{\mathsf{BC}} \not\subseteq \mathsf{AT}$.

Proof. Let $C = (a \vee b \vee c)$ and $F = \{(\neg a \vee x), (\neg b \vee x), (\neg c \vee x), (a \vee b)\}$. Since $\neg b \in \mathsf{ALA}(F, C)$, it follows that $(F, C) \in \mathsf{AT}$. Now, assume that C is set-blocked by some set $L \subseteq C$, i.e., for every C' with $C' \cap \bar{L} \neq \emptyset$, $(C \setminus L) \cup \bar{L} \cup C'$ is a tautology. Since $L \subseteq C$ is non-empty, $(\neg v \vee x) \cap \bar{L} \neq \emptyset$ for at least one $(\neg v \vee x)$ with $v \in \{a, b, c\}$. Let therefore C' be such a $(\neg v \vee x)$. Then, $v \notin (C \setminus L)$ and $v \notin \bar{L}$. Hence, $(C \setminus L) \cup \bar{L} \cup C'$ is not a tautology and thus C is not set-blocked by L, a contradiction. We conclude that $(F, C) \notin \mathsf{SET}_{\mathsf{BC}}$.

Finally, let $F = \emptyset$ and $C = (a)$. Then, $(F, C) \in \mathsf{SET}_{\mathsf{BC}}$, but $(F, C) \notin \mathsf{AT}$. □

Proposition 16. $\mathsf{AT} \not\subseteq \mathsf{SUP}_{\mathsf{BC}}$ *and* $\mathsf{SUP}_{\mathsf{BC}} \not\subseteq \mathsf{AT}$.

Proof. Consider again the clause $C = (a \vee b \vee c)$ and the formula $F = \{(\neg a \vee x), (\neg b \vee x), (\neg c \vee x), (a \vee b)\}$ from the proof of Proposition 15, 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 15), it is not set-blocked in $F|\tau$, hence $(F, C) \notin \mathsf{SUP}_{\mathsf{BC}}$.

To see that $\mathsf{SUP}_{\mathsf{BC}} \not\subseteq \mathsf{AT}$, let $F = \emptyset$ and $C = (a)$. Then, since $(F, C) \in \mathsf{SET}_{\mathsf{BC}}$ and $\mathsf{SET}_{\mathsf{BC}} \subset \mathsf{SUP}_{\mathsf{BC}}$, we get that $(F, C) \in \mathsf{SUP}_{\mathsf{BC}}$ but $(F, C) \notin \mathsf{AT}$. □

From Proposition 16 together with the fact that $\mathsf{AT} \subset \mathsf{RAT}$ we get:

Corollary 17. $\mathsf{RAT} \not\subseteq \mathsf{SUP}_{\mathsf{BC}}$.

Proposition 18. $\mathsf{SET}_{\mathsf{BC}} \not\subseteq \mathsf{RAT}$.

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 $(F, C) \in \mathsf{SET}_{\mathsf{BC}}$.

Now, for the literal a there is only the clause $C' = (\neg a \vee b)$ that contains $\neg a$ and $C \cup C' \setminus \{\neg a\} = (a \vee b)$. Furthermore, for the literal b there is only the clause $C'' = (a \vee \neg b)$ that contains $\neg b$ and here again we get that $C \cup C'' \setminus \{\neg b\} = (a \vee b)$. Since $\mathsf{ALA}(F \setminus \{C\}, (a \vee b)) = (a \vee b)$ is not a tautology, $(F, C) \notin \mathsf{RAT}$. □

Corollary 19. RAT *is incomparable with* $\mathsf{SET}_{\mathsf{BC}}$ *and* $\mathsf{SUP}_{\mathsf{BC}}$.

8 Conclusion

Previous research and recent SAT competitions have clearly revealed the power of solving techniques based on the redundancy property of literal-blocked clauses. One reason for the success of this redundancy property is that it is local in the sense that it can be efficiently checked by considering only the resolution environment of a clause [8, 12, 14]. In this paper, we showed that there are even more general local redundancy properties like set-blocked clauses ($\mathsf{SET}_{\mathsf{BC}}$) and super-blocked clauses ($\mathsf{SUP}_{\mathsf{BC}}$). Local redundancy properties are 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.

Our complexity analysis showed that checking the newly introduced redundancy properties is computationally expensive in the worst case. This seemingly limits their practical applicability at first glance. However, we presented bounded variants that can be checked more efficiently and we expect them to considerably improve the solving process 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, literal-blocked clauses have been shown to be even more effective than in SAT solving [6, 16] and we expect that this also holds for quantified variants of $\mathsf{SET}_{\mathsf{BC}}$ and $\mathsf{SUP}_{\mathsf{BC}}$.

References

1. Biere, A., Heule, M., van Maaren, H., Walsh, T., eds.: Handbook of Satisfiability. IOS Press (2009)
2. Vizel, Y., Weissenbacher, G., Malik, S.: Boolean satisfiability solvers and their applications in model checking. Proc. of the IEEE **103**(11) (2015) 2021–2035
3. Barrett, C.W., Sebastiani, R., Seshia, S.A., Tinelli, C.: Satisfiability modulo theories. In: Handbook of Satisfiability. IOS Press (2009) 825–885
4. Reger, G., Suda, M., Voronkov, A.: Playing with AVATAR. In Felty, P.A., Middeldorp, A., eds.: Proc. of the 25th Int. Conf. on Automated Deduction (CADE 2015). Volume 9195 of LNCS., Cham, Springer (2015) 399–415
5. Järvisalo, M., Biere, A., Heule, M.: Simulating circuit-level simplifications on CNF. Journal on Automated Reasoning **49**(4) (2012) 583–619
6. Heule, M., Järvisalo, M., Lonsing, F., Seidl, M., Biere, A.: Clause elimination for SAT and QSAT. Journal of Artifical Intelligence Research **53** (2015) 127–168
7. Järvisalo, M., Heule, M., Biere, A.: Inprocessing rules. In Gramlich, B., Miller, D., Sattler, U., eds.: Proc. of the 6th Int. Joint Conf. on Automated Reasoning (IJCAR 2012). Volume 7364 of LNCS., Heidelberg, Springer (2012) 355–370
8. Manthey, N., Philipp, T., Wernhard, C.: Soundness of inprocessing in clause sharing SAT solvers. In Järvisalo, M., Van Gelder, A., eds.: Proc. of the 16th Int. Conf. on Theory and Applications of Satisfiability Testing (SAT 2013). Volume 7962 of LNCS., Heidelberg, Springer (2013) 22–39
9. Heule, M., Biere, A.: Blocked clause decomposition. In McMillan, K., Middeldorp, A., Voronkov, A., eds.: Proc. of the 19th Int. Conf. on Logic for Programming, Artificial Intelligence, and Reasoning (LPAR-19). Volume 8312 of LNCS., Heidelberg, Springer (2013) 423–438
10. Iser, M., Manthey, N., Sinz, C.: Recognition of nested gates in CNF formulas. In Heule, M., Weaver, S., eds.: Proc. of the 18th Int. Conf. on Theory and Applications of Satisfiability Testing (SAT 2015). Volume 9340 of LNCS., Cham, Springer (2015) 255–271
11. Balyo, T., Fröhlich, A., Heule, M., Biere, A.: Everything you always wanted to know about blocked sets (but were afraid to ask). In Sinz, C., Egly, U., eds.: Proc. of the 17th Int. Conf. on Theory and Applications of Satisfiability Testing (SAT 2014). Volume 8561 of LNCS., Cham, Springer (2014) 317–332
12. Chen, J.: Fast blocked clause decomposition with high quality. CoRR **abs/1507.00459** (2015)
13. Kullmann, O.: On a generalization of extended resolution. Discrete Applied Mathematics **96-97** (1999) 149–176
14. Järvisalo, M., Biere, A., Heule, M.: Blocked clause elimination. In Esparza, J., Majumdar, R., eds.: Proc. of the 16th Int. Conf. on Tools and Algorithms for the Construction and Analysis of Systems (TACAS 2010). Volume 6015 of LNCS., Heidelberg, Springer (2010) 129–144
15. Heule, M., Hunt, Jr., W.A., Wetzler, N.: Verifying refutations with extended resolution. In Bonacina, M.P., ed.: Proc. of the 24th Int. Conf. on Automated Deduction (CADE 2013). Volume 7898 of LNCS., Heidelberg, Springer (2013) 345–359
16. Lonsing, F., Bacchus, F., Biere, A., Egly, U., Seidl, M.: Enhancing search-based QBF solving by dynamic blocked clause elimination. In Davis, M., Fehnker, A., McIver, A., Voronkov, A., eds.: Proc. of the 20th Int. Conf. on Logic for Programming, Artificial Intelligence (LPAR-20). Volume 9450 of LNCS., Heidelberg, Springer (2015) 418–433