EPiC Series in Computing

Volume 57, 2018, Pages 583–603

LPAR-22. 22nd International Conference on Logic for Programming, Artificial Intelligence and Reasoning

A Theory of Satisfiability-Preserving Proofs in SAT Solving

Adrián Rebola-Pardo[1]* and Martin Suda[2]†

[1] TU Wien, Austria,
arebolap@forsyte.at

[2] Czech Technical University in Prague, Czech Republic,
martin.suda@cvut.cz

Abstract

We study the semantics of propositional interference-based proof systems such as DRAT and DPR. These are characterized by modifying a CNF formula in ways that preserve satisfiability but not necessarily logical truth. We propose an extension of propositional logic called *overwrite logic* with a new construct which captures the meta-level reasoning behind interferences. We analyze this new logic from the point of view of expressivity and complexity, showing that while greater expressivity is achieved, the satisfiability problem for overwrite logic is essentially as hard as SAT, and can be reduced in a way that is well-behaved for modern SAT solvers. We also show that DRAT and DPR proofs can be seen as overwrite logic proofs which preserve logical truth. This much stronger invariant than the mere satisfiability preservation maintained by the traditional view gives us better understanding on these practically important proof systems. Finally, we showcase this better understanding by finding intrinsic limitations in interference-based proof systems.

1 Introduction

The story of SAT solving is one of great success, which has made SAT solvers widely used in practical applications due to their ability to routinely solve instances with millions of variables. Two recent interrelated breakthroughs in SAT solving are the extension of conflict-driven clause learning (CDCL) solvers [34] with inprocessing techniques that allow efficient solving in fragments where CDCL has exponential behavior [1, 24, 19] and the introduction of increasingly expressive and easy to check proof systems to certify the correctness of unsatisfiability results [7, 11, 41, 17].

The connection between these two features is subtle, and it is yet to be fully understood how to exploit it to improve the solvers' efficiency. Inprocessing techniques [19], including clause elimination [10], symmetry breaking [1], bounded variable addition [24], and parity reasoning [35], often tackle fragments hard for CDCL [37, 38, 9]. Behind the scene, these techniques

*Supported by the Austrian National Research Network S11403-N23 (RiSE), the LogiCS doctoral program W1255-N23 of the Austrian Science Fund (FWF), the Vienna Science and Technology Fund (WWTF) through grant VRG11-005, and Microsoft Research through its PhD Scholarship Programme.

†Supported by the ERC Consolidator grant AI4REASON 649043, ERC Starting grant SYMCAR 639270, and the Austrian research project FWF S11409-N23.

briefly violate the rules of the proof system underlying SAT solvers, namely DAG-like resolution [2], in a sound way. Escaping the limitations of this proof system also allows one to circumvent their known pitfalls in terms of complexity. For example, symmetry breaking or bounded variable addition allow to derive clauses that are not implied by the formula, but rather just consistent with it. In doing so, potential exponential speed-ups are attained [12, 17], and the solver briefly operates under much more general implicit proof systems.

The delete resolution asymmetric tautology (DRAT) and delete propagation redundancy (DPR) proof systems were developed to express and check these operations [41, 17]. While DRAT proof generation is widely supported [25, 28, 13], DPR is fairly recent and is yet to be adopted in practice. Recent results show that these proof systems are polynomially equivalent to the extended resolution proof system [16, 20], for which no exponential lower bounds are known [22]. In other words: if SAT solvers fully exploited the power of DRAT and DPR, no known fragments of CNF formulas would be hard to solve. Alas, this is far from the case, despite some recent advances in exploiting DPR-based reasoning [18]. Constructing methods that capitalize on DPR is notably complex due to the involvement of the whole formula in the inferences, a phenomenon known as *interference* [14].

Interference has practical implications beyond SAT solving, especially when one considers the use of proofs with aims other than certifying the solvers' results. For example, no method exists in the literature to generate Craig interpolants [5] from DRAT proofs that can be used in model checking [27], and the allowed inferences lack of some of the intuitive features familiar from other proof systems such as resolution [29]. The issues of interference all boil down to the allowed inferences being just satisfiability-preserving, rather than truth-preserving [29]. For example, methods to obtain interpolants from other proof systems work by recursively constructing partial interpolants throughout the proof tree [26, 40, 8, 33], but the invariants ensuring the correctness of this procedure rely strongly on the proof being truth-preserving, i.e. the conclusion of every inference being a consequence of its premises [29]. Even worse, DRAT and DPR proofs are not even tree-shaped. Rather, these proofs modify the formula in an incremental way by introducing and deleting clauses. Again, this is mandated by interference: since the *absence* of some clauses is a requirement for some inferences, proofs in the shape of clause trees do not make much sense. It has been argued that further research is needed to fully understand the potential and possible shortcomings of interference reasoning [14].

Contributions One way to understand DPR proofs is to look for semantic invariants preserved throughout proofs. In truth-preserving proof systems this is straightforward, but previous results imply that no such invariants exist for DPR proofs [29]. In this paper, we argue that DPR can be construed as operating over a more general logic, called *overwrite logic*, which derives expressions that generalize clauses. Reasoning under this perspective is similar to assumptions "without loss of generality" that are familiar in mathematics. Not only can we find stronger invariants when using overwrite logic: DPR proofs, which are satisfiability-preserving proofs when considered over propositional logic, become truth-preserving proofs. Moreover, we argue that for any practical application overwrite logic is no more different from CNF formulas than propositional logic, and Tseitin-like procedures [30] exist with similar complexity. Finally, this new perspective enables inferences that cannot be performed with interference reasoning.

Related work DRAT and DPR proofs [41, 17] were developed to increase the reliability of SAT solvers' unsatisfiability results by allowing powerful inferences that easily expressed the reasoning techniques used by SAT solvers [25, 13, 28]. Recent work shows that both proof systems are essentially as expressive as extended resolution [16, 20], for which no exponential lower bounds exist [22]. Some early work on interference-based solving techniques exists but is still relatively limited [18]; a goal of this work is to provide a deeper understanding of interference that furthers the development of such techniques.

The issue of the semantics preserved by DRAT proofs has been tackled in [29] for a restricted case without deletion, presenting an intuitive explanation in terms of permissive definitions. A result presented there seemed to prevent semantic invariants stronger than satisfiability preservation in the unrestricted case. We show that overwrite logic is able to overcome this limitation. The model extraction method from [19] bears some clear similarities to our work, although the focus there is on identifying correctness criteria for interference during solving. Some inconsistencies on DRAT proof checking have their roots in the unexpected interaction between deletion and interference reasoning [32].

Many methods for extracting interpolants from resolution-like proofs have been proposed [26, 40, 8, 33], but truth-preservation in inferences is required. (Propositional) interpolant extraction from extended resolution proofs, hence from DRAT proofs, has been shown to be exponential under cryptographic assumptions [3, 23]. Since many model checking approaches are interpolation-based [27], or can work as interpolant generators [4], there are compelling reasons to study the interaction between interpolation and interference reasoning.

2 Preliminaries

We consider a countable set of propositional variables. An interpretation maps each propositional variable to either 0 or 1. Throughout this paper we define several kinds of logical expressions and semantics for them. For each logical expression E, we give a notion of when an interpretation I satisfies E; in this case we write $I \vDash E$. An expression E is satisfiable whenever there is some interpretation I with $I \vDash E$, and unsatisfiable otherwise. Furthermore, given two logical expressions E_1, E_2, we say that E_1 entails E_2 whenever $I \vDash E_2$ for every interpretation I such that $I \vDash E_1$; and that E_1 satisfiability-entails E_2 whenever the existence of an interpretation I_1 with $I_1 \vDash E_1$ implies the existence of an interpretation I_2 such that $I_2 \vDash E_2$. We denote this by $E_1 \vDash E_2$ and $E_1 \vDash_{\text{sat}} E_2$, respectively. We say that E_1 is equivalent to E_2 whenever $E_1 \vDash E_2$ and $E_2 \vDash E_1$; and that E_1 is satisfiability-equivalent to E_2 whenever $E_1 \vDash_{\text{sat}} E_2$ and $E_2 \vDash_{\text{sat}} E_1$. We denote this by $E_1 \equiv E_2$ and $E_1 \equiv_{\text{sat}} E_2$, respectively. The set of variables occurring in an expression E is denoted by $\mathrm{Var}(E)$.

In this paper we consider CNF formulas and propositional formulas as expressions of different logics, in order to clearly compare them and their variants from a complexity perspective. Let us first start with propositional logic. We consider propositional logic (PL) formulas as recursively defined by the following Backus-Naur form:

$$\varphi \coloneqq \top \mid \bot \mid p \mid \neg\varphi \mid \varphi \vee \varphi \mid \varphi \wedge \varphi$$

where p represents propositional variables. The semantics are defined as usual, e.g. $I \vDash p$ whenever $I(p) = 1$, and $I \vDash \varphi \vee \psi$ whenever $I \vDash \varphi$ or $I \vDash \psi$.

We now define clausal normal form (CNF) logic. A literal is an expression of the form $+p$ or $-p$, where p is a propositional variable.[1] The complement of a literal is defined by $\overline{+p} = -p$ and $\overline{-p} = +p$; we then say that $+p$ and $-p$ are complementary literals. A cube is an expression of the form $\langle l_1, \dots, l_n \rangle$, and a clause one of the form $[l_1, \dots, l_n]$, where the l_i form a finite, complement-free set of unique literals. We regard both cubes and clauses as sets

[1] Albeit unusual, the distinction between literals and variables or their negations makes somewhat easier the definition of overwrites later in Section 3; it does not otherwise affect the content.

of literals. The complements of clauses and cubes are given by $\overline{\langle l_1, \dots, l_n \rangle} = [\overline{l_1}, \dots, \overline{l_n}]$ and $\overline{[l_1, \dots, l_n]} = \langle \overline{l_1}, \dots, \overline{l_n} \rangle$. A CNF formula is a finite set of clauses. The semantics of CNF logic are given as usual: the literal $+p$ is equivalent to p, and $-p$ is equivalent to $\neg p$; clauses (resp. cubes) are interpreted as disjunctions (resp. conjunctions) of their literals; and CNF formulas are interpreted as conjunctions of their clauses. Given a clause C and a cube Q such that $Q \nvDash C$, i.e. Q and C have no literals in common, we then define the clause $C|_Q$ as $C \setminus \overline{Q}$. We lift this definition to CNF formulas via $F|_Q = \left\{ C|_Q \mid C \in F \text{ and } Q \nvDash C \right\}$.

2.1 Redundancy notions on CNF logic

For the rest of this section, we focus on CNF logic, since it is the logic modern SAT solvers operate within. They work by iteratively modifying a CNF formula in a satisfiability-equivalent way [10, 41]. In particular, redundant clauses are introduced into or deleted from the formula. A clause C is said to be redundant in a formula F whenever $F \cup \{C\} \equiv_{\text{sat}} F$. Several refinements of the notion of redundancy have been proposed, and new inference techniques arise from them.

Entailment criteria It is straightforward to observe that C is redundant in F whenever $F \vDash C$, so entailment criteria are a good source of redundancy criteria. Checking whether C is entailed by F is coNP-complete [18], but more restricted criteria are relatively easy to check. Two simple entailment criteria can be combined to express every entailed clause [21]. The first one is resolution: given two clauses C and D, and a literal l, we say that C is resolvable with D upon l (or simply that the resolvent $C \otimes_l D$ exists) whenever l is the only literal k such that $k \in C$ and $\overline{k} \in D$. In this case, we call the clause $C \otimes_l D$ defined by $(C \setminus \{l\}) \vee (D \setminus \{\overline{l}\})$ the resolvent of C with D upon l; it can be then shown that $\{C, D\} \vDash C \otimes_l D$. Since the literal l is unique, we will sometimes omit it and simply write $C \otimes D$. The second criterion is subsumption: we write $C \sqsubseteq D$ whenever every literal in C is contained in D; in that case, we have $C \vDash D$.

A particular combination of resolution and subsumption yields an entailment criterion able to express clauses introduced in CDCL SAT solving [6, 2], called reverse unit propagation (RUP) [7]. Let us call a resolvent $C \otimes_l D$ a merge resolution whenever $D \setminus \{\overline{l}\} \sqsubseteq C$ [42]. Then, a subsumption-merge chain[2] is a string of clauses $E_0, \dots, E_n$ such that some clauses $A_0, \dots, A_n$ exist with $E_0 \sqsubseteq A_0$, and the resolvent $A_i = A_{i-1} \otimes E_i$ exists and is a merge resolution for each $i = 1, \dots, n$; in this case, we say that the subsumption-merge chain derives the clause A_n. A clause C is a RUP clause in a CNF formula F if there exists some SM chain $E_0, \dots, E_n$ deriving C where $\{E_0, \dots, E_n\} \subseteq F$; if this is the case, then $F \vDash C$. Although this is not obvious from the definition, it is relatively easy to establish whether a clause is a RUP in a formula by using unit propagation, and subsumption-merge chains can be easily extracted in this case. We refer the reader to [29] for further reference.

Interference criteria More generally, redundancy criteria exist for which the redundant clause is not entailed by the formula, but rather is simply consistent with it. In such criteria, both the presence of some clauses and *the absence of others* are required; this phenomenon is known as *interference* [14]. Several interrelated interference criteria have been proposed. Resolution asymmetric tautologies (RAT) [10, 41] are widely used, but so far the most powerful criterion, subsuming all others to the best of our knowledge, is that of propagation redundancy (PR) [17].

[2]Subsumption-merge chains have been called subsumption self-subsuming resolution (SSSR) chains in [29].

A clause C is a PR clause in F upon a cube Q whenever, $Q \cap C$ is nonempty and, furthermore, every clause in $F|_Q$ is a RUP in the CNF formula $F|_{\overline{C}}$. In this case, it can be shown that $F \equiv_{\text{sat}} F \cup \{C\}$ [17]. From a model-theoretical standpoint, the condition that such clauses must be RUPs is quite restrictive; in fact, the satisfiability-equivalence above holds if "RUP" is replaced by any other entailment criterion, including entailment itself. PR clauses are very powerful: every non-empty redundant clause can be expressed as a PR clause [17]; unfortunately, finding the appropriate cube Q is NP-complete [18].

2.2 Interference-based proofs

Traditional proof systems represent proofs as trees (or, if clauses are repeated, as DAGs) with clauses as nodes and edges representing truth-preserving inferences, i.e. inferences where the conclusion is entailed by the premises. However, this approach is inadequate for interference-based inferences, since a representation of the "clauses derived so far" must be kept. In general, satisfiability-preserving inferences preclude tree-shaped proofs, since they violate monotonicity: if C is redundant in F, it does not follow that C is redundant in $F \cup G$, whereas this property holds if entailment is considered instead of redundancy [29].

There are compelling reasons to use satisfiability-preserving inferences. From a theoretical perspective, satisfiability-preserving inferences allow exponentially shorter proofs [22]. A more pragmatic standpoint also shows that proof trees are not appropriate: proofs generated by SAT solvers follow the solver run, deriving essentially the same clauses [11]. However, SAT solvers frequently delete redundant clauses [10, 11], and since we do not have monotonicity, deletion can *enable* satisfiability-preserving inferences, which would never happen with truth-preserving inferences. This makes it hard to reason about interference criteria, to the extent that the interplay between interference and clause deletion is at the root of some discrepancies between the theory and the practice of proof logging in SAT solvers [32]. Hence, for interference-based proofs, an instruction string approach is more appropriate.

Instructions are expressions which come in two flavors: introductions $\mathbf{i}\!:\!C$ and deletions $\mathbf{d}\!:\!C$, where C is a clause. Optionally, introduction instructions can be annotated by a cube Q as in $\mathbf{i}\!: Q \blacktriangleright C$. Each instruction string maps a CNF formula F into its *accumulated CNF formula*, also called *effect*, as follows:

$$\mathrm{eff}_F(\epsilon) = F \qquad \mathrm{eff}_F(\mathbf{i}\!: C,\, \pi) = \mathrm{eff}_{F \cup \{C\}}(\pi) \qquad \mathrm{eff}_F(\mathbf{d}\!: C,\, \pi) = \mathrm{eff}_{F \setminus \{C\}}(\pi)$$

Intuitively, the accumulated formula applies the instructions along the string: $\mathbf{i}\!: C$ introduces C in the CNF formula and $\mathbf{d}\!: C$ deletes C from it.

With the purpose of designing unsatisfiability certificates that can be both efficiently generated and efficiently checked, the *Delete Resolution Asymmetric Tautology* (DRAT) and *Delete Propagation Redundant* (DPR) proof systems were introduced [41, 17]. State-of-the-art SAT solvers are able to generate a DRAT proof when reporting an unsatisfiable CNF formula. Its extension to DPR proofs was developed recently, and allows for practically shorter and easier to generate proofs than DRAT [17]; theoretically, both proof systems are (polynomially) as powerful as extended resolution, though [16, 20]. The DPR proof system allows introduction and deletion instructions under the following conditions:

- ϵ is a DPR derivation from F.
- $\mathbf{i}\!: C,\, \pi$ is a DPR derivation from F if π is a DPR derivation from F, and C is a RUP clause in $\mathrm{eff}_F(\pi)$.

- **i:** $Q \blacktriangleright C$, π is a DPR derivation from F if π is a DPR derivation from F, and C is a PR clause in $\mathrm{eff}_F(\pi)$ upon Q.
- **d:** C, π is a DPR derivation from F whenever π is a DPR derivation from F.

The DRAT proof system can be obtained by further requiring the annotation Q in the third condition to have exactly one literal l: the clause C would in that case be RAT in $\mathrm{eff}_F(\pi)$ upon l [17]. A DPR refutation of a CNF formula F is a DPR derivation π from F such that $[\,] \in \mathrm{eff}_F(\pi)$. If π is a DPR derivation from a CNF formula F, then $F \vDash_{\mathrm{sat}} \mathrm{eff}_F(\pi)$; in particular, if it is a refutation of F, then F is unsatisfiable.

3 Overwrite propositional logic

The proof of correctness of PR clause introduction in [17] relies on modifying an interpretation by *overwriting* variable truth assignments. It is thus apparent that assignment overwriting plays a pivotal role in the semantics of DPR proofs. This was already crucial in the approach from [29], where a variation on the form of formulas is explored. Unfortunately, the definitional formulas discussed in that work are cumbersome to work with and not versatile enough to deal with deletion. We propose a more natural approach which only adds one constructor to propositional logic; we call the obtained logic *overwrite propositional logic* (OPL).

Given an interpretation I and a cube B, the overwrite of B on I is the interpretation $I + B$ defined by:

$$(I+B)(p) = I(p), \text{ if } p \notin \mathrm{Var}(B) \quad (I+B)(p) = 0, \text{ if } -p \in B \quad (I+B)(p) = 1, \text{ if } +p \in B$$

The interpretation $I + B$ is well-defined, because B does not contain complementary literals. Intuitively, the interpretation $I + B$ is obtained by minimally forcing I to satisfy B.

We define a rule as an expression of the form $(B :\!- E)$, where B is a cube and E is any logical expression. Rules are not logical expressions, i.e. they are merely syntactic building blocks and do not have truth values. Given an interpretation I, the conditional overwrite of B on I subject to E is the interpretation $I + (B :\!- E)$ defined by:

$$I + (B :\!- E) = I + B, \text{ if } I \vDash E \qquad\qquad I + (B :\!- E) = I, \text{ if } I \nvDash E$$

Starting from propositional logic, *overwrite propositional logic* (OPL) is obtained by considering one extra connective: given two OPL formulas φ and ψ, and a cube B, the expression $\nabla(B :\!- \psi).\, \varphi$ is an OPL formula. Its semantics are given by conditional overwrites: $I \vDash \nabla(B :\!- \psi).\, \varphi$ if and only if $I + (B :\!- \psi) \vDash \varphi$.

In the following, we consider strings of rules $\vec{\varepsilon} = \varepsilon_1 \dots \varepsilon_n$, where the ε_i are rules. We denote by $\nabla\vec{\varepsilon}.\, E$ the iterative application of overwrite operators to the expression E, i.e. $\nabla\varepsilon_1.\, \nabla\varepsilon_2.\, \dots \nabla\varepsilon_n.\, E$. Similarly, we denote by $I + \vec{\varepsilon}$ an iterative conditional overwrite on I, i.e. $I + \varepsilon_1 + \varepsilon_2 + \dots + \varepsilon_n$.

Clause-based normal forms As for propositional logic, clause-based normal forms can be defined for overwrite propositional logic. An initial restriction can be made on the form of rules. Whereas OPL allows rules of the form $(B :\!- \varphi)$, where φ is an arbitrary OPL formula, we will show that it suffices to consider *cubic rules* given by $(B :\!- Q)$, where Q is a cube. Overwrite clausal normal form (OCNF) is obtained by stacking overwrite connectives with cubic rules over each clause. Another restriction can be obtained by writing a stack of overwrite connectives over a CNF formula; we call such expressions uniformly overwrite clausal normal forms (UOCNF). As we show in Section 3.2, the expressive power of OPL, OCNF and UOCNF is polynomially the same; this is unlike the case for propositional logic, where PL is exponentially more expressive than CNF.

An overwrite clause is an expression of the form $\nabla\vec{\varepsilon}.\,C$ where $\vec{\varepsilon}$ is a (possibly empty) string of cubic rules and C is a clause; an OCNF formula is a set of overwrite clauses. The semantics are given as expected: I satisfies the overwrite clause $\nabla\vec{\varepsilon}.\,C$ whenever $I + \vec{\varepsilon}$ satisfies C; and I satisfies an OCNF formula F whenever I satisfies every overwrite clause in F. Furthermore, if F is a (standard) CNF formula, then we say that $\nabla\vec{\varepsilon}.\,F$ is a UOCNF formula, again with the expected semantics: $I \vDash \nabla\vec{\varepsilon}.\,F$ whenever $I + \vec{\varepsilon} \vDash F$. The following result shows that UOCNF can be naturally embedded into (i.e. regarded as a sub-logic of) OCNF:

Proposition 1. *$\nabla(B :\!-\, \psi)$ distributes with $\vee$, $\wedge$, $\neg$. In particular, $\nabla\vec{\varepsilon}.\,F \equiv \{\nabla\vec{\varepsilon}.\,C \mid C \in F\}$.*

Expressivity and complexity of overwrite formulas Having extended propositional logic PL with the overwrite connective to obtain the new logic OPL and its restrictions OCNF and UOCNF, it is natural to ask how these logics compare with respect to expressivity and what is the complexity of deciding their satisfiability. Expressivity can be studied from a *qualitative* as well as *quantitative* point of view. For instance, it is well known that for every CNF formula an equivalent PL formula exists and *vice versa* [39], so we say that PL and CNF have the same qualitative expressivity. However, CNF formulas need in the worst case to be exponentially larger than some equivalent PL formulas [39]; we thus say that PL is quantitatively more expressive than CNF.

When we talk about expressivity we consider the existence of *truth-equivalent* formulas. A different question is the complexity of the satisfiability problem for a given formula in a logic, in particular whether a polynomially-sized, *satisfiability-equivalent* formula can be found in polynomial time. Following our example, even if PL is quantitatively more expressive than CNF, the Tseitin procedure [30] provides a way to reduce the satisfiability problem for PL to the satisfiability problem for CNF in linear time, and so both satisifiability problems are NP-complete.

Figure 1 displays what we will know by the end of this section. For example, three arrows are shown between PL and CNF. The inclusion arrow shows that CNF can be embedded in PL; the coiled arrow shows that transforming PL formulas into equivalent CNF formulas is worst-case exponential; and the dashed arrow shows that the satisfiability problem for PL can be polynomially reduced to the satisfiability problem for CNF. These simulations are shown in black, because they are known from previous work. Simulations in grey are straightforward, namely the embeddings CNF $\subseteq$ PL $\subseteq$ OPL, and CNF $\subseteq$ UOCNF $\subseteq$ OCNF $\subseteq$ OPL. Finally, three non-trivial simulations are shown in red, so we explain them in the next sections. All in all, by the end of this section we will have argued the following results:

1. CNF, PL, UOCNF, OCNF and OPL all have the same qualitative expressivity.

2. UOCNF, OCNF and OPL all have the same quantitative expressivity.

3. The procedure we provide to simulate OPL through PL is worst-case exponential.

4. Despite the latter two results, the satisfiability problem for OPL can be reduced to the satisfiability problem for CNF in a manner suitable for SAT solving.

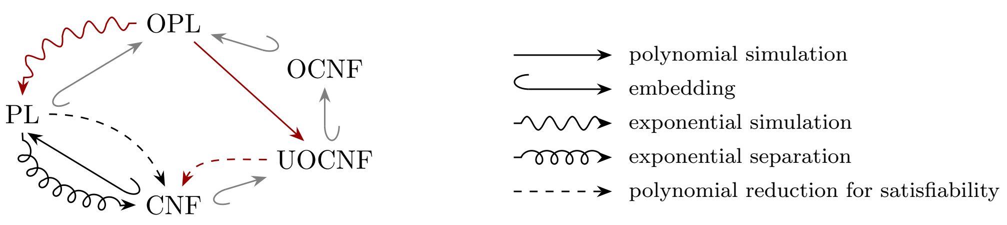


Figure 1: Simulation landscape between the CNF, PL, OPL, OCNF and UOCNF logics.

3.1 Qualitative expressivity

In the following paragraphs we argue that OPL, OCNF, UOCNF, PL and CNF have all the same qualitative expressivity, i.e. for every formula in one of these logics, we can find a truth-equivalent formula in any other of them. This does not a priori need to be the case, for there exist sets of interpretations that are not expressible within PL, e.g. the set of interpretations that map exactly one variable to 1. We show this by finding a cycle of simulations through all of these logics. The embeddings CNF $\subseteq$ UOCNF $\subseteq$ OCNF $\subseteq$ OPL, together with the well-known (exponential) transformation of PL formulas into equivalent CNF formulas [39], leaves only the simulation of OPL by PL to be shown. We devote the rest of this section to obtain a procedure that, given an OPL formula, generates a truth-equivalent PL formula.

Let us first consider PL formulas ψ and φ, so they do not contain ∇ connectives. One can show the following equivalence:

$$\nabla(B :\!-\, \psi).\, \varphi \equiv (\psi \vee \varphi) \wedge (\neg\psi \vee \varphi|_B) \tag{1}$$

where $\varphi|_B$ is obtained by replacing variables x in φ by $\top$ if $+x \in B$ and by $\bot$ if $-x \in B$. Observe that the PL formula $\varphi|_B$ is truth-equivalent to the CNF formula $F|_B$ as defined in Section 2 whenever F is a CNF formula truth-equivalent to φ, so the notation is consistent; also for the sake of consistency, we call $\varphi|_B$ the reduct of φ under B. Formula (1) expresses the intuition that either the condition ψ holds, in which case we must evaluate φ with respect to the interpretation overwritten by B, or ψ does not hold and we evaluate φ as usual.

In order to extend this transformation to arbitrary OPL formulas, we need to generalize also the notion of reduct to OPL formulas. However, this is not so straightforward, since it raises the question how would one replace literals inside overwrite rules by $\top$ or $\bot$. We instead generalize the notion of reduct in such a way that it directly yields a PL formula, which effectively defines a truth-equivalent transformation from OPL to PL. In particular, we define:

$$(\nabla(B :\!-\, \psi).\, \varphi)|_A = (\psi\,|_A \vee \varphi|_A) \wedge (\neg\psi\,|_A \vee \varphi|_{B\cup A\setminus\overline{B}})$$

Before we continue, let us justify this definition, which may seem odd at first. In order to compute this reduct, we first apply the transformation from (1) to the formula $\nabla(B :\!-\, \psi).\, \varphi$, and then apply the reduct using the definitions for the other connectives. We then obtain:

$$(\psi\,|_A \vee \varphi|_A) \wedge (\neg\psi\,|_A \vee \varphi\,|_B\,|_A)$$

Finally, we observe that $\varphi\,|_B\,|_A = \varphi|_{B\cup A\setminus\overline{B}}$ holds for every propositional formula. Having clarified this point, we now show a result characterizing the semantics of reducts, and we obtain as an immediate consequence that PL qualitatively simulates OPL.

Theorem 1. *Let φ be an OPL formula and A be a cube. For every interpretation I, we have $I \vDash \varphi|_A$ if and only if $I + A \vDash \varphi$. In particular, the PL formula $\varphi|_\emptyset$ is truth-equivalent to φ.*

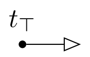
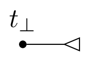
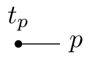
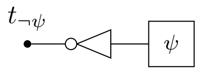
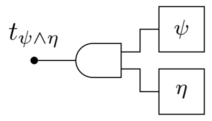
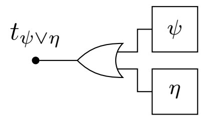
$\top$	$\bot$	p	$\neg\psi$	$\psi \land \eta$	$\psi \lor \eta$
					
$\{t_\top \leftrightarrow \top\}$	$\{t_\bot \leftrightarrow \bot\}$	$\{t_p \leftrightarrow p\}$	$\{t_{\neg\psi} \leftrightarrow \neg t_\psi\}$	$\{t_{\psi\land\eta} \leftrightarrow t_\psi \land t_\eta\}$	$\{t_{\psi\lor\eta} \leftrightarrow t_\psi \lor t_\eta\}$

Figure 2: Tseitin transformations of subformulas into circuits and definitions. Variables t_φ represent the circuit output for subformulas φ.

3.2 Quantitative expressivity

We now analyze the question of the relative quantitative expressiveness of each logic. It is known that PL is exponentially more expressive than CNF. This in particular means that there exists no mapping Φ that maps PL formulas φ to truth-equivalent CNF formulas $\Phi(\varphi)$ such that the size of $\Phi(\varphi)$ is polynomial in the size of φ.

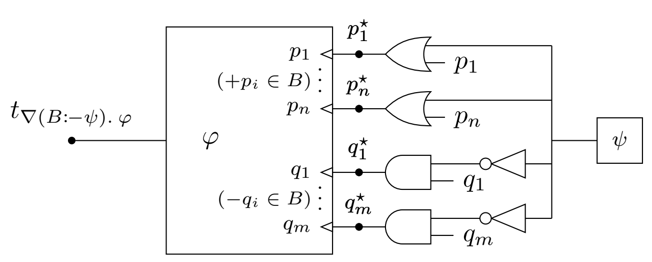
Remarkably, a similar separation does not exist between OPL formulas and neither of their clause-based normal forms OCNF and UOCNF. In fact, any OPL formula can be linearly transformed into a *truth-equivalent* UOCNF formula. The Tseitin transformation achieves a similar result from PL formulas into CNF formulas, but in this case the transformation is only satisfiability-preserving, albeit some additional features are preserved. Together with the embeddings UOCNF $\subseteq$ OCNF $\subseteq$ OPL, the quantitative equivalence of these three logics follows.

Our procedure is very similar to the Tseitin transformation, with two main differences. On the one hand, the presence of an overwrite prefix in an UOCNF allows us to circumvent the loss of truth preservation: values are assigned to Tseitin variables using overwrites. On the other hand, an additional transformation for the overwrite connective is necessary.

The Tseitin transformation can be understood as follows. First, a formula is recursively transformed into a Boolean circuit, possibly with sharing, and such a circuit does not contain cycles. Then, new variables are introduced for every node in the circuit. For every such new variable x, connected through a gate to nodes whose input nodes have variables $y_1, \dots, y_n$, a definition of the form $x \leftrightarrow \varphi(y_1, \dots, y_n)$ of bounded size is generated, as shown in Figure 2.

Our procedure completes the Tseitin transformation by providing transformation rules to turn the overwrite connective $\nabla(B :\!- \psi).\, \varphi$ into a circuit and a set of definitions. Let us split B into its positive literals $+p_1, \dots, +p_n$ and its negative literals $-q_1, \dots, -q_m$. Given circuits for the subformulas φ and ψ, we construct the circuit shown in Figure 3. The circuit for φ has as inputs the variables occurring in φ, which may or may not include variables from $p_1, \dots, p_n, q_1, \dots, q_m$. Now, for any interpretation I, the formula φ is evaluated under the interpretation $I + (B :\!- \psi)$, and so the variables p_i take the new truth value 1 if either ψ or p_i are satisfied by I; and the variables q_i take the new truth value 1 if q_i is satisfied and ψ is falsified by I. Hence, the circuit computes new auxiliary variables $p_1^\star, \dots, p_n^\star, q_1^\star, \dots, q_m^\star$ that express the values of the modified variables after the application of the rule $(B :\!- \psi)$, and then uses the starred variables as the input of φ. If a variable does not occur in B, then that variable is itself used as input in φ.

In order to encode each generated sub-circuit, definitions as shown in Figures 2 and 3 can be generated and collected in a set D. Observe that, for every node named with x, a unique definition of the form $x \leftrightarrow \psi$ exists in D. Furthermore, since the circuit is acyclic, the nodes can be topologically sorted as $x_1, \dots, x_n$, i.e. if there is a path from x_i to x_j in the circuit, then $i < j$. Now, for every node x_i let us clausify the definition $x_i \leftrightarrow \varphi_i$, obtaining a CNF formula



$$t_{\nabla(B:-\psi).\,\varphi} \leftrightarrow t_{\varphi^\star}$$

$$p_1^\star \leftrightarrow p_1 \vee t_\psi$$
$$\dots$$
$$p_n^\star \leftrightarrow p_n \vee t_\psi$$

$$q_1^\star \leftrightarrow q_1 \wedge \neg t_\psi$$
$$\dots$$
$$q_m^\star \leftrightarrow q_m \wedge \neg t_\psi$$

Figure 3: Circuit associated to the formula $\nabla(B :- \psi).\,\varphi$. Triangles represent the use of the outputs $p_i^\star, q_i^\star$ as the inputs p_i, q_i in the ψ circuit. Introduced definitions are shown in the right: variables $p_i^\star, q_i^\star$ are required to be new.

H_i, whose size is bounded by a constant. Each obtained clause $C \in H_i$ contains either $+x_i$ or $-x_i$, so it can then be turned into the rule $\left(\pm x_i :- \overline{C}\right)$, where the polarity of x_i is chosen to be the same as that of the corresponding occurrence in C. Then, we define a string $\vec{\varepsilon_i}$ given by the concatenation of all such rules for the definition $x_i \leftrightarrow \varphi_i$. The following result then holds:

Theorem 2. *Under the conditions above, $\varphi \equiv \nabla\varepsilon_1 \dots \varepsilon_n.\, \{[+t_\varphi]\}$, where t_φ is the variable associated to the output of the constructed circuit.*

Let us briefly explain the intuition behind Theorem 2. By induction, one can show that, after applying the rule string $\vec{\varepsilon_1} \dots \vec{\varepsilon_i}$, the variables $x_1, \dots, x_i$ get assigned the respective truth values that the nodes labeled by $x_1, \dots, x_i$ would get in the circuit; in particular, the node $x_n = t_\varphi$ gets assigned the truth value of the circuit output, which is equivalent to φ.

This finishes the proof that OPL, UOCNF and OCNF are all equally expressive. Furthermore, given the embedding PL $\subseteq$ OPL, and that PL is exponentially more expressive than CNF, we know that there is an exponential separation between CNF and OPL. However, the question whether OPL is exponentially more expressive than PL remains. In general, the translation from Section 3.1 is exponential:

Example 1. Consider the sequence of OPL formulas $\varphi_n = \nabla(+x_n :- x_{n-1}) \dots (+x_1 :- x_0).\, \bot$ for $n \in \mathbb{N}$. Given any cube B which does not refer to variables $x_0, \dots, x_n$, one can show by induction that $\varphi_{n+1}|_B = \left(x_n \vee \varphi_n|_\emptyset\right) \wedge \left(\neg x_n \vee \varphi_n|_\emptyset\right)$. This means that $\left|\varphi_n|_\emptyset\right|$ is exponential on n, which implies that our transformation from OPL formulas to PL formulas is worst-case exponential. ■

From our argument throughout this section, it becomes apparent that the expressivity of OPL is similar to that of (sharing) circuits. To the best of our knowledge, it is an open question whether an exponential separation exists between circuits and propositional formulas, so we speculate the following:

Conjecture 1. *OPL is exponentially more expressive than PL.*

3.3 Complexity of the satisfiability problem

Any potential use of overwrite logic in practice would be hindered if new solvers for this logic would need to be developed. In particular, there are two closely related questions that must be answered: 1) what is the complexity of the satisfiability problem for OPL, and 2) whether efficient and effective translations into existing solving paradigms. In Section 3.2 we have shown that an OPL formula can be linearly translated into an equivalent circuit. Linear encodings of circuits into *almost* truth-equivalent formulas through the Tseitin procedure and variations thereof are well known [30], in the sense that all interpretations of each formula coincide in the variables occurring in the OPL formula. In a more formal way, the definitions from Figures 2 and 3, together with an assertion of the output node, is a suitable translation to CNF in this sense. We then conclude:

Corollary 1. *The satisfiability problem for OPL formulas is NP-complete. Furthermore, every OPL formula φ can be linearly translated into a CNF formula F such that $F \vDash \varphi$, and for every interpretation I satisfying φ another interpretation J satisfying F exists such that I and J coincide on* $\mathrm{Var}(\varphi)$.

4 Understanding DPR

Throughout this section, we will obtain results that clarify the semantics of DPR proofs, i.e. the semantic invariants they preserve. However, before we get to that, we believe it is useful to understand DPR proofs in terms of a technique widely used in handwritten proofs, namely making assumptions without loss of generality. We now provide two examples of the use of these techniques; later in this section we will explain their relation to DPR proofs.

Example 2. A special case of Ramsey's theorem can be stated as follows: given any set of three people a, b and c, either one of them is a friend of everybody else, or one of them is a friend of nobody else (assuming symmetry of the friendship relation). In order to show this, we can assume without loss of generality the following constraint: if c is a friend of a, then it is also a friend of b. Were this not the case, one can swap the roles of a and b, and obtain a new problem satisfying both the preconditions of the problem and our assumption. ■

Example 3. The pigeonhole problem asks whether m pigeons can be fit into n pigeonholes in such a way that no two pigeons are allocated to the same hole. To show this, one can assume without loss of generality that pigeons 1 and m are not both assigned to holes n and 1, respectively. If this happened, pigeon 1 can be moved to hole 1 while pigeon m is moved to hole n. This operation applied to any solution of the problem yields another solution of the problem, and additionally our assumption is fulfilled. ■

In either case, assumptions without loss of generality are not implied by the problem, but rather just consistent with it. The argument to introduce them is similar in both cases: if, under the conditions of the problem, the assumption does not hold then one can perform a transformation that again falls into the conditions of the problem and, additionally, the assumption holds after the transformation.

4.1 Satisfiability preservation as a proof invariant

One pleasant and intuitive property of more traditional proof systems is truth preservation: if we can derive G from F then we know that every model of F is a model of G. This property holds for DRUP proofs, i.e. DPR proofs without PR introduction instructions, since both RUP introduction and clause deletion are truth-preserving operations. Truth equivalence cannot be concluded, because arbitrary clause deletion may allow additional models. Truth preservation acts as a semantic invariant: if we consider the set M of all interpretations satisfying F then the accumulated formulas F' throughout the proof have the property that M is a set of satisfying interpretations for F', and in particular for G. Hence, one such DRUP proof acts as a witness that all models of F are models of G. Truth preservation is a very strong invariant. For example, effective methods to construct interpolants from resolution or DRUP proofs implicitly rely heavily on truth preservation [33] and reasoning about non-truth-preserving proofs is notably harder and often involves global reasoning about proofs [15, 31, 36].

Truth preservation is nevertheless lost once PR clause introduction is considered. Introducing a PR clause is only satisfiability-equivalent, but in general is not truth-preserving. Satisfiabilility-preservation in PR introduction, together with truth-preservation of deletions and RUP introductions, only establishes satisfiability preservation as a semantic invariant of PR proofs. In particular, the property preserved by the proof is that the set of interpretations satisfying each accumulated formula is non-empty. A reasonable question is then whether this is the strongest semantic invariant preserved by DPR proofs: stronger invariants would allow us to extract more information and have a more local understanding of proofs. In this regard, a converse to the satisfiability-preservation result appeared in [29]:

Theorem 3. *Let F and G be CNF formulas such that $F \vDash_{sat} G$. Then, there exists a DPR proof that derives G from F.*

This result seems at first discouraging. One interpretation thereof is that the existence of a proof does not yield stronger semantic invariants than satisfiability preservation. Another interpretation is that no stronger semantic invariant can be obtained within propositional logic. As we will see in the rest of this paper, moving our framework to the fringes of these understandings (in the former case, by considering the proof itself instead of its mere existence; in the latter, by understanding DPR inferences as acting over OCNFs) leads to new semantic invariants.

4.2 Proof-dependent semantic invariants

Our understanding of satisfiability preservation in DPR proofs boils down to [17, Theorem 1] which states that if F is a CNF formula and C is a PR clause in F upon a cube Q, then the existence of *some* model of F implies the existence of *some* model of $F \cup \{C\}$. This is actually an understatement: a careful look into the proof of the result from [17] shows that we know precisely how to transform *every* model of F into a model of $F \cup \{C\}$.

Theorem 4. *Let F be a CNF formula and C a PR clause in F upon some cube Q. Then, for every interpretation I satisfying F, the interpretation $I + \left(Q :\!- \overline{C}\right)$ satisfies $F \cup \{C\}$.*

Theorem 4 gives a new view on PR introduction, from which the relation to assumptions without loss of generality becomes apparent: if an interpretation satisfying the problem F violates the assumption C, we refine the interpretation in such a way that the resulting interpretation satisfies both the problem and the assumption. In the case of PR introduction, interpretation refinement is only allowed in the form of block overwrites: specific literals, given by the witness cube Q, are set to true or false. In particular, PR clauses cannot (directly) encode refinements where a variable takes the truth value of another variable.

Example 4. We can encode the problem from Example 2 into the following CNF formula F, where variable x_{uv} is satisfied if u and v are friends:

$$[+x_{ab},\, +x_{bc}] \quad [-x_{ab},\, -x_{bc}] \quad [+x_{ab},\, +x_{ac}] \quad [-x_{ab},\, -x_{ac}] \quad [+x_{ac},\, +x_{bc}] \quad [-x_{ac},\, -x_{bc}]$$

This is the problem from Example 4 in [13]. There, the symmetry breaker $[-x_{ac},\, +x_{bc}]$ is introduced which expresses the assumption without loss of generality given in Example 2. This clause can be introduced as a PR clause in F upon the witness cube $\langle -x_{ac},\, +x_{bc}\rangle$, and the latter expresses precisely the swap of the roles of a and b. ■

Example 5. The pigeonhole problem from Example 3 is expressed as the CNF formula F containing the following clauses:

$$\{[+x_{i1}, \dots, +x_{in}] \mid 1 \le i \le m\} \cup \{[-x_{ik},\, -x_{jk}] \mid 1 \le i < j \le m \text{ and } 1 \le k \le n\},$$

where the variable x_{uv} is satisfied whenever pigeon u is in hole v. Our assumption without loss of generality from Example 3 is written as the clause $[-x_{1n},\, -x_{m1}]$. This clause can be introduced in F as a PR clause upon the witness cube $\langle -x_{1n},\, -x_{m1},\, +x_{11},\, +x_{mn}\rangle$. As in the previous example, this cube represents the transformation proposed there, which moves pigeon 1 to hole 1, and pigeon m to hole n. This PR introduction corresponds to the first clause introduced in the DPR proof for the pigeonhole problem with $n+1$ pigeons and n holes presented in [17]. ■

Semantics in terms of assumption without loss of generality can be expressed, as given by Theorem 4, using conditional overwrites. By collecting introduced overwrite rules, new semantic invariants can be developed to circunvent the limitations posed by Theorem 3. These semantics do not depend only on the existence of a proof, but also on the specific features of the proof, specifically on the witness cubes. In addition to the accumulated CNF formula throughout a proof, we recursively define the *accumulated rule string* throughout a DPR proof π as follows:

$$\operatorname{ars}(\epsilon) = \epsilon \quad \operatorname{ars}(\mathbf{i}\!:C,\, \pi) = \operatorname{ars}(\mathbf{d}\!:C,\, \pi) = \operatorname{ars}(\pi) \quad \operatorname{ars}(\mathbf{i}\!:Q \blacktriangleright C,\, \pi) = \big(Q :\!-\, \overline{C}\big),\, \operatorname{ars}(\pi)$$

Intuitively, $\operatorname{ars}(\pi)$ collects the set of overwrite rules that would be applied by every PR introduction throughout the proof π in the same order as they occur. Theorem 4 can then straightforwardly be extended to work over whole DPR proofs, which gives a much stronger semantics than satisfiability preservation:

Corollary 2. *Let F be a CNF formula and π be a DPR derivation of a CNF formula G from F. Then, for every interpretation I such that $I \vDash F$, we have $I + ars(\pi) \vDash G$.*

4.3 DPR proofs as truth-preserving proofs on UOCNFs

An alternative way to understand DPR proofs is to embed the interpretation transformations induced by PR introductions in Theorem 4 directly into the formula syntax. Overwrite logics are especially appropriate for this: from Theorem 4 it becomes apparent that PR introduction naturally operates over UOCNF formulas.

Corollary 3. *Let F be a CNF formula and C a PR clause in F upon some cube Q. Then, $F \vDash \nabla\big(Q :\!-\, \overline{C}\big).\, F \cup \{C\}$.*

Observe a key feature of this perspective on PR introduction: when considered as an operation over UOCNF formulas, it is not only a satisfiability-preserving operation, but actually a *truth*-preserving one. Hence, the question of what are the semantics preserved by DPR proofs becomes trivial under the UOCNF perspective. This means that DPR proofs can be construed as (truth-preserving!) proofs over UOCNF formulas as follows:

DPR instruction	inference over CNFs	inference over UOCNFs
$\mathbf{i}\!: C$	$F \vDash F \cup \{C\}$	$\nabla\vec{\varepsilon}.\, F \vDash \nabla\vec{\varepsilon}.\, F \cup \{C\}$
$\mathbf{i}\!: C \blacktriangleright Q$	$F \vDash_{\text{sat}} F \cup \{C\}$	$\nabla\vec{\varepsilon}.\, F \vDash \nabla\vec{\varepsilon},\, \big(Q :\!-\, \overline{C}\big).\, F \cup \{C\}$
$\mathbf{d}\!: C$	$F \vDash F \setminus \{C\}$	$\nabla\vec{\varepsilon}.\, F \vDash \nabla\vec{\varepsilon}.\, F \setminus \{C\}$

Intuitively, in the same way that the accumulated CNF formula throughout a DPR proof applies the clause introductions and deletions specified by the proof, the accumulated UOCNF formula additionally applies the overwrites associated to PR introductions. In particular, a DPR proof π deriving a CNF formula G from a CNF formula F can be understood as deriving from an UOCNF formula $\nabla\vec{\varepsilon}.\, F$ the UOCNF $\nabla\vec{\varepsilon}, \operatorname{ars}(\pi).\, G$. The semantics of DPR proofs then become clear in the way that one would expect from a traditional proof system:

Corollary 4. *Let π be a DPR proof deriving a UOCNF formula Σ from a UOCNF formula Π, in the sense explained above. Then, $\Pi \vDash \Sigma$.*

5 New insights with overwrite logic

In Section 4.3, a perspective on DPR proofs as proofs over UOCNF formulas was explored. However, the answer is rather unsatisfactory: although now we have truth-preservation as an invariant, the usual properties of truth-preserving proofs are not attained. In particular, proofs are still interference-based: the whole CNF part of the formula is involved in the inference criterion, which precludes tree-shaped proofs where inferences depend exclusively on the presence of some clauses, rather than on their absence. The choice of UOCNF logic is to blame for this limitation, since clauses cannot be split from the formulas. However, understanding DPR proofs from an OCNF perspective eliminates this disadvantange.

Observe that, if a DPR proof can derive the UOCNF formula $\nabla\vec{\varepsilon}.\, G$ from a formula F, then we have $F \vDash \nabla\vec{\varepsilon}.\, G \equiv \{\nabla\vec{\varepsilon}.\, C \mid C \in G\}$, where the latter is a OCNF formula. In particular, every overwrite clause $\nabla\vec{\varepsilon}.\, C$ independently follows from F, *regardless of other clauses.* This suggests that interference can then be avoided. Our goal is to enrich a typical proof system over clauses with resolution and subsumption as proof rules into a proof system over overwrite clauses and to show that such a proof system can be used to simulate DPR in a tree-shaped manner. The inferences of this *overwrite resolution* proof system are shown in Figure 4.

Theorem 5. *The inference rules from Figure 4 are truth-preserving.*

The rules ow-res and ow-sub are simply generalizations of the resolution and subsumption inference rules for clauses. Following the characterization of RUPs as clauses derived by SSSR chains, these allow the simulation of RUP inferences in an OCNF setting. The rule ow-bot is necessary to eliminate the string of overwrite rules to ultimately derive a contradiction. Finally, the role of PR clause introduction is performed by the ow-wlog, ow-taut and ow-ax rules. Perhaps counterintuitively, introducing *any* clause C under an overwrite rule $\left(B :\!-\, \overline{C}\right)$ only requires that $B \vDash C$, as shown by the ow-ax rule; this is a necessary but insufficient condition for C to be a PR clause upon B, and it corresponds only to the requirement for PR introduction that B and C share some literal. In essence, this inference simply proves that an assumption without loss of generality holds after the associated model transformation. However, the troublesome part of reasoning without loss of generality is usually arguing that the model still satisfies the original conditions after the transformation, i.e. showing that $F \vDash \nabla\left(B :\!-\, \overline{C}\right).\, F$. The rules ow-wlog and ow-taut allow to independently derive the overwrite clauses corresponding to this kind of reasoning: each of these rules introduces a new overwrite rule at the bottom of the prefix of an overwrite clause, deriving from $\nabla\vec{\varepsilon}.\, D$ the overwrite clause $\nabla\varepsilon\left(B :\!-\, \overline{C}\right).\, D$. The reason why we need two such inference rules is simply that the clause $C \cup D|_B$ in ow-wlog does not exist when $B \vDash C$ or $\overline{C} \vDash D|_B$, in which case it is simply not needed, obtaining the inference rule ow-taut.

We now need to argue that this proof system can indeed simulate the wealth of instructions allowed by the DPR proof system. Firstly, observe that deletion instructions can be outright

$$\frac{\nabla\vec{\varepsilon}.\ C \qquad \nabla\vec{\varepsilon}.\ D}{\nabla\vec{\varepsilon}.\ C \otimes D}\ \textsf{ow-res} \qquad \text{if the resolvent } C \otimes D \text{ exists}$$

$$\frac{\nabla\vec{\varepsilon}.\ C}{\nabla\vec{\varepsilon}.\ C \cup D}\ \textsf{ow-sub} \qquad \text{if } \overline{C} \not\models D$$

$$\frac{\nabla\vec{\varepsilon}.\ [\,]}{[\,]}\ \textsf{ow-bot}$$

$$\frac{\nabla\vec{\varepsilon}.\ D \qquad \nabla\vec{\varepsilon}.\ C \cup D|_B}{\nabla\vec{\varepsilon}\big(B :\!-\, \overline{C}\big).\ D}\ \textsf{ow-wlog} \qquad \text{if } B \not\models D \text{ and } \overline{C} \not\models D|_B$$

$$\frac{\nabla\vec{\varepsilon}.\ D}{\nabla\vec{\varepsilon}\big(B :\!-\, \overline{C}\big).\ D}\ \textsf{ow-taut} \qquad \text{if } B \models D \text{ or } \overline{C} \models D|_B$$

$$\frac{}{\nabla\vec{\varepsilon}\big(B :\!-\, \overline{C}\big).\ C}\ \textsf{ow-ax} \qquad \text{if } B \models C$$

Figure 4: Proof rules for the overwrite resolution proof system

ignored: since we are working with tree-shaped, truth-preserving proofs, deletion in this setting fulfills the same role as in resolution or DRUP: they only represent promises that a proof will not use a given clause in the future, and so they do not carry semantic meaning. RUP introductions are unproblematic: every RUP clause C in F can be derived by a proof using resolution and subsumption from F. This can be simulated by deriving an analogous proof of $\nabla\vec{\varepsilon}. C$ from $\{\nabla\vec{\varepsilon}.\ D \mid D \in F\}$ using the ow-res and ow-sub inference rules. The PR introduction case is more interesting. If C is a PR clause in F upon a cube B then, as argued in Section 4.3, we need to derive $\nabla\vec{\varepsilon}\big(B :\!-\, \overline{C}\big).\ F \cup \{C\}$ from $\nabla\vec{\varepsilon}.\ F$. On the one hand, this involves deriving the overwrite clause $\nabla\vec{\varepsilon}\big(B :\!-\, \overline{C}\big).\ C$, but this is straightforward using the inference rule ow-ax. On the other hand, we need to derive $\nabla\vec{\varepsilon}\big(B :\!-\, \overline{C}\big).\ D$ for every clause $D \in F$. The following result implies that we can always do so:

Proposition 2. *Let C be a PR clause in a CNF formula F upon a cube B, and $D \in F$. Then, either of the following hold: $B \models D$, or $\overline{C} \models D|_B$, or $C \cup D|_B$ is a RUP in F.*

Indeed, the premise $\nabla\vec{\varepsilon}.\ D$ in both inference rules ow-wlog and ow-taut is automatically satisfied, because $D \in F$. Moreover, this is the only premise in the case of ow-taut; if the first or the second alternatives in Proposition 2 holds then we can simply apply that inference rule to derive $\nabla\vec{\varepsilon}\big(B :\!-\, \overline{C}\big). D$. Otherwise, the clause $C \cup D|_B$ is a RUP in F; once again, this means that we can find a proof of $\nabla\vec{\varepsilon}.\ C \cup D|_B$ from $\nabla\vec{\varepsilon}.\ F$, thus deriving the premises required by ow-wlog to derive $\nabla\vec{\varepsilon}\big(B :\!-\, \overline{C}\big).\ D$.

The overwrite resolution proof system thus allows to completely represent DPR proofs. Let us illustrate how a specific DPR proof looks like in this framework.

Example 6. Let F be the formula containing the following clauses:

$$\begin{array}{cccc}
[+x,\ +y,\ -z] & [+y,\ +z,\ -t] & [-x,\ -z,\ -t] & [-x,\ +y,\ +t] \\
[-x,\ -y,\ +z] & [-y,\ -z,\ +t] & [+x,\ +z,\ +t] & [+x,\ -y,\ -t]
\end{array}$$

We consider the following DPR (in fact, DRAT) refutation of F, which appeared in [41]:

$$\begin{array}{r l}
\textbf{i:} & \langle -x\rangle \blacktriangleright [-x] \\
\textbf{d:} & [-x,\ -y,\ z] \\
\textbf{d:} & [-x,\ -z,\ -t] \\
\textbf{d:} & [-x,\ +y,\ +t] \\
\textbf{i:} & [+y] \\
\textbf{d:} & [+x,\ +y,\ -z] \\
\textbf{d:} & [+y,\ +z,\ +t] \\
\textbf{i:} & [\,]
\end{array}$$

In this DPR proof, the clause $C = [-x]$ is introduced as a RAT, i.e. a PR clause in F upon the cube $B = \langle -x\rangle$. On the one hand, this means, by definition, that for every clause $D \in F \cup \{C\}$ we can derive every clause in $F|_B$ as a RUP in $F|_{\overline{C}}$. On the other hand, within the overwrite resolution framework, we need to derive $\nabla\varepsilon.D$ for every clause $D \in F\cup\{C\}$ (where we have taken $\varepsilon = (\langle -x\rangle :\!- \langle +x\rangle))$, which can only be done using the inference rules ow-taut and ow-wlog. Proposition 2 and the discussion above show that this can be done, but it may not be clear exactly how. For the clauses $D \in F$ such that $B \vDash D$ or $\overline{C} \vDash D|_B$ hold, this is straightforward: since the clause D is in F, we can simply apply the ow-taut rule:

$$\dfrac{[-x,\ -y,\ +z]}{\nabla\varepsilon.\,[-x,\ -y,\ +z]}\ \textsf{ow-taut} \qquad \dfrac{[-x,\ -z,\ -t]}{\nabla\varepsilon.\,[-x,\ -z,\ -t]}\ \textsf{ow-taut} \qquad \dfrac{[-x,\ +y,\ +t]}{\nabla\varepsilon.\,[-x,\ +y,\ +t]}\ \textsf{ow-taut}$$

The case for clauses $D \in F$ such that $B \nvDash D$ and $\overline{C} \nvDash D|_B$ is slightly more complicated. The first condition is needed so that the reduct clause $D|_B$ exists. In particular, $D|_B \in F|_B$, so by the definition of a PR clause $D|_B$ is a RUP in $F|_{\overline{C}}$. To apply the inference rule ow-wlog we need to derive the clause $C \cup D|_B$ from F; Proposition 2 tells us that it can be derived as a RUP. Thus, the question is: how can we transform the subsumption-merge chain that derives $D|_B$ from $F|_{\overline{C}}$ into a subsumption-merge chain that derives $C \cup D|_B$ from F? To explain this, let us compute the reduct $F|_{\overline{C}}$:

$$[-y,\ +z] \qquad [+y,\ +z,\ -t] \qquad [-y,\ -z,\ +t] \qquad [-z,\ -t] \qquad [+y,\ +t]$$

Let us take as an example the clause $D = [+x,\ +y,\ -z]$. The reduct of this clause under B is $D|_B = [+y,\ -z]$, and as explained above, it is a RUP on $F|_{\overline{C}}$, and so a subsumption-merge chain deriving it from $F|_{\overline{C}}$ can be found, shown below on the left. Every premise used is the reduct $D'|_{\overline{C}}$ for some clause $D' \in F$. By replacing these reducts by the original clauses, we obtain a subsumption-merge chain that derives the clause $C \cup D|_B = [-x,\ +y,\ -z]$. Then, the ow-wlog inference rule can be applied to derive $\nabla\varepsilon.\,D = \nabla\varepsilon.\,[+x,\ +y,\ -z]$, shown shaded:

$$\textsf{res}\ \dfrac{\textsf{sub}\ \dfrac{[+y,\ +t]}{[+y,\ -z,\ +t]} \quad [-z,\ -t]}{[+y,\ -z]} \quad \longrightarrow \quad \dfrac{[+x,\ +y,\ -z] \quad \dfrac{\dfrac{[-x,\ +y,\ +t]}{[-x,\ +y,\ -z,\ +t]}\ \textsf{ow-sub} \quad [-x,\ -z,\ -t]}{[-x,\ +y,\ -z]}\ \textsf{ow-res}}{\nabla\varepsilon.\,[+x,\ +y,\ -z]}\ \textsf{ow-wlog}$$

We can similarly derive the overwrite clauses corresponding to every clause in F:

$$\dfrac{[+x,\ +y,\ +t] \qquad \dfrac{\dfrac{[-x,\ -y,\ +z]}{[-x,\ -y,\ +z,\ +t]}\,\textsf{ow-sub} \qquad [-x,\ +y,\ +z]}{[-x,\ +z,\ +t]}\,\textsf{ow-res}}{\nabla\varepsilon.\,[+x,\ +y,\ +t]}\,\textsf{ow-wlog} \qquad \dfrac{\dfrac{[+y,\ +z,\ -t]}{[-x,\ +y,\ +z,\ -t]}\,\textsf{ow-sub} \qquad [+y,\ +z,\ -t]}{\nabla\varepsilon.\,[+y,\ +z,\ -t]}\,\textsf{ow-wlog}$$

$$\dfrac{[+x,\ -y,\ -t] \qquad \dfrac{\dfrac{[-x,\ -y,\ +z]}{[-x,\ -y,\ +z,\ -t]}\,\textsf{ow-sub} \qquad [-x,\ -z,\ -t]}{[-x,\ -y,\ -t]}\,\textsf{ow-res}}{\nabla\varepsilon.\,[+x,\ -y,\ -t]}\,\textsf{ow-wlog} \qquad \dfrac{\dfrac{[-y,\ -z,\ +t]}{[-x,\ -y,\ -z,\ +t]}\,\textsf{ow-sub} \qquad [-y,\ -z,\ +t]}{\nabla\varepsilon.\,[-y,\ -z,\ +t]}\,\textsf{ow-wlog}$$

Now it only remains to derive the overwrite clause corresponding to the RAT clause C itself. As with any PR clause, it can be derived directly with the ow-ax inference rule:

$$\dfrac{}{\nabla\varepsilon.\,[-x]}\,\textsf{ow-wlog}$$

The other two introduction inferences in the DPR proof are RUP clause introductions, so in particular they can be derived from previous clauses by subsumption-merge chains. Since we have derived the corresponding overwrite clauses and all of them have the same overwrite prefix, we can simply replace each clause by their overwrite version and replace resolutions and subsumptions by ow-res and ow-sub. In our proof system, we will not directly obtain the empty clause, but instead it will be preceded by an overwrite prefix. The inference rule ow-bot allows in this case to derive the empty clause.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\nabla\varepsilon.\,[+x,\ +z,\ +t]}{\nabla\varepsilon.\,[+x,\ -y,\ +z,\ +t]} \qquad \nabla\varepsilon.\,[-y,\ -z,\ +t]}{\nabla\varepsilon.\,[+x,\ -y,\ +t]} \qquad \nabla\varepsilon.\,[+x,\ -y,\ -t]}{\nabla\varepsilon.\,[+x,\ -y]} \qquad \dfrac{\dfrac{\dfrac{\dfrac{\nabla\varepsilon.\,[+y,\ +z,\ -t]}{\nabla\varepsilon.\,[+x,\ +y,\ +z,\ -t]} \qquad \nabla\varepsilon.\,[+x,\ +y,\ +t]}{\nabla\varepsilon.\,[+x,\ +y,\ +z]} \qquad \nabla\varepsilon.\,[+x,\ +y,\ -z]}{\nabla\varepsilon.\,[+x,\ +y]} \qquad \nabla\varepsilon.\,[-x]}{\nabla\varepsilon.\,[y]}}{\nabla\varepsilon.\,[+x]} \qquad \nabla\varepsilon.\,[-x]}{\dfrac{\nabla\varepsilon.\,[\,]}{[\,]}\,\textsf{ow-bot}}$$

■

Limitations of interference-based proof systems Our framework presents an inconspicuous feature: it can perform inferences which cannot be obtained within DPR proofs, due to the notion of accumulated formula inherent to interference-based proofs. Let us first express the problem within an interference framework and then we show how this limitation can be overcome with the use of the finer-grained reasoning of tree-shaped proofs over overwrite clauses. Given a CNF formula F and a clause C candidate to be PR in Q, the notion of PR clause forces us to derive every clause in $F|_Q$ from $F|_{\overline{C}}$ as a RUP. By Proposition 2, this is equivalent to deriving $C \cup D|_Q$ as a RUP in F for every clause $D \in F$ for which the former expression makes sense, i.e. $Q \not\models D$ and $\overline{C} \not\models D|_Q$ must hold.

It might be the case that for some $D \in F$ the clause $C \cup D|_Q$ is not a RUP in F, but we may know how to derive it by RUP once a lemma D' has been introduced. Under the interference framework, one would then need to derive $C \cup D'|_Q$ by RUP, which may need an additional lemma D'', and so on. This influence of the whole formula, which interference is named after, makes it very hard to reason about PR introduction. This limitation is eliminated in our OCNF framework: since there is no notion of accumulated formula, there is no need for $\nabla(B :- \overline{C}).\, D$ to be derived for *all* clauses $D \in F$, and so we are able to derive more redundant clauses.

Example 7. Consider a clause C and a cube Q such that $Q \models C$, and let us assume that $Q \setminus C \neq \emptyset$. We provide a formula F and a clause $D \in F$ such that C is neither a RUP clause nor a PR clause in F *nor* in $F \setminus \{D\}$. In other words, C cannot be derived by neither of the following proof fragments:

$$\textbf{i:}\, C \qquad \textbf{i:}\, Q \blacktriangleright C \qquad \textbf{d:}\, D,\ \textbf{i:}\, C \qquad \textbf{d:}\, D,\ \textbf{i:}\, Q \blacktriangleright C$$

Nevertheless, it can be derived within our framework. Let us choose literals $l \in Q \cap C$ and $k \in Q \setminus C$, and variables x, y not occurring in C or Q. Consider the formula F given by clauses:

$$[l,\ k,\ -u,\ -v] \qquad [\overline{l},\ \overline{k},\ -u,\ -v] \qquad [\overline{l},\ k,\ +u,\ +v] \qquad [\overline{k},\ -u]$$

Then, C is not a RUP in F nor in $F \setminus \{D\}$, since unit propagation on $F \cup \overline{C}$ does not reach a contradiction. Furthermore, C is not a PR in F upon Q: although Q implies both clauses $[l,\ k,\ -u,\ -v]$ and $[\overline{l},\ k,\ +u,\ +v]$, and furthermore $C \cup [\overline{l},\ k,\ +u,\ +v]|_Q = C \cup [-u,\ -v]$ reaches contradiction by unit propagation using clauses $[\overline{k},\ -u]$ and $[l,\ k,\ -u,\ -v]$, the clause $C \cup [\overline{k},\ -u]|_Q = C \cup \{-u\}$ cannot itself be derived as a RUP in F. Moreover, C is not a PR either in $F \setminus \{[\overline{k},\ -u]\}$ upon Q, since the removed clause was needed to reach a contradiction before. However, in our framework, we can use rules ow-taut and ow-wlog to derive the overwrite clauses:

$$\nabla(Q :- \overline{C}).\, [l,\ k,\ -u,\ -v] \qquad \nabla(Q :- \overline{C}).\, [\overline{l},\ \overline{k},\ -u,\ -v] \qquad \nabla(Q :- \overline{C}).\, [\overline{l},\ k,\ +u,\ +v]$$

■

6 Conclusion

We introduced overwrite logic, a new extension of propositional logic with a connective which allows us to capture the meta-level reasoning behind interference-based proof systems.

We analyzed the logic from the point of view of expressivity and complexity. We learnt that while often more succinct than standard propositional logic, the satisfiability problem for overwrite logic remains in NP and can be efficiently reduced to SAT.

We shed new light on DPR and DRAT proofs by proposing to understand the non-monotone PR introduction (and its restriction RAT introduction) as formalisation of the proof technique "assumption without loss of generality" common in mathematics.

We then show that any DPR (DRAT) proof can be seen as a truth preserving proof in overwrite logic. This much stronger invariant than mere satisfiability preservation restores monotonicity: the clause deletion operation simply becomes a promise not to reuse the deleted clause later in the proof. This simplifies reasoning about proofs, opening possibilities for non-trivial processing of proofs such as interpolation.

This resolves the question of implicit semantics of DRAT proofs only partially answered in previous work [29] and extends to the more general DPR proof system as well. Moreover, we use this framework to identify an intrinsic limitation of interference proof systems and propose how it could be overcome in the more general setting using overwrite logic.

References

[1] Fadi A. Aloul, Arathi Ramani, Igor L. Markov, and Karem A. Sakallah. Solving difficult instances of boolean satisfiability in the presence of symmetry. *IEEE Trans. on CAD of Integrated Circuits and Systems*, 22(9):1117–1137, 2003.

[2] Paul Beame, Henry A. Kautz, and Ashish Sabharwal. Towards understanding and harnessing the potential of clause learning. *J. Artif. Intell. Res.*, 22:319–351, 2004.

[3] Maria Luisa Bonet, Toniann Pitassi, and Ran Raz. No feasible interpolation for tc0-frege proofs. In *38th Annual Symposium on Foundations of Computer Science, FOCS '97, Miami Beach, Florida, USA, October 19-22, 1997*, pages 254–263. IEEE Computer Society, 1997.

[4] Aaron R. Bradley. Understanding IC3. In Alessandro Cimatti and Roberto Sebastiani, editors, *Theory and Applications of Satisfiability Testing - SAT 2012 - 15th International Conference, Trento, Italy, June 17-20, 2012. Proceedings*, volume 7317 of *Lecture Notes in Computer Science*, pages 1–14. Springer, 2012.

[5] William Craig. Linear reasoning. A new form of the herbrand-gentzen theorem. *J. Symb. Log.*, 22(3):250–268, 1957.

[6] Allen Van Gelder. Producing and verifying extremely large propositional refutations - have your cake and eat it too. *Ann. Math. Artif. Intell.*, 65(4):329–372, 2012.

[7] Evguenii I. Goldberg and Yakov Novikov. Verification of proofs of unsatisfiability for CNF formulas. In *2003 Design, Automation and Test in Europe Conference and Exposition (DATE 2003), 3-7 March 2003, Munich, Germany*, pages 10886–10891. IEEE Computer Society, 2003.

[8] Arie Gurfinkel and Yakir Vizel. Druping for interpolates. In *Formal Methods in Computer-Aided Design, FMCAD 2014, Lausanne, Switzerland, October 21-24, 2014*, pages 99–106. IEEE, 2014.

[9] Armin Haken. The intractability of resolution. *Theor. Comput. Sci.*, 39:297–308, 1985.

[10] Marijn Heule, Matti Järvisalo, Florian Lonsing, Martina Seidl, and Armin Biere. Clause elimination for SAT and QSAT. *J. Artif. Intell. Res.*, 53:127–168, 2015.

[11] Marijn Heule, Warren A. Hunt Jr., and Nathan Wetzler. Trimming while checking clausal proofs. In *Formal Methods in Computer-Aided Design, FMCAD 2013, Portland, OR, USA, October 20-23, 2013*, pages 181–188. IEEE, 2013.

[12] Marijn Heule, Warren A. Hunt Jr., and Nathan Wetzler. Verifying refutations with extended resolution. In Maria Paola Bonacina, editor, *Automated Deduction - CADE-24 - 24th International Conference on Automated Deduction, Lake Placid, NY, USA, June 9-14, 2013. Proceedings*, volume 7898 of *Lecture Notes in Computer Science*, pages 345–359. Springer, 2013.

[13] Marijn Heule, Warren A. Hunt Jr., and Nathan Wetzler. Expressing symmetry breaking in DRAT proofs. In Amy P. Felty and Aart Middeldorp, editors, *Automated Deduction - CADE-25 - 25th International Conference on Automated Deduction, Berlin, Germany, August 1-7, 2015, Proceedings*, volume 9195 of *Lecture Notes in Computer Science*, pages 591–606. Springer, 2015.

[14] Marijn Heule and Benjamin Kiesl. The potential of interference-based proof systems. In Giles Reger and Dmitriy Traytel, editors, *ARCADE 2017, 1st International Workshop on Automated Reasoning: Challenges, Applications, Directions, Exemplary Achievements, Gothenburg, Sweden, 6th August 2017*, volume 51 of *EPiC Series in Computing*, pages 51–54. EasyChair, 2017.

[15] Marijn J. H. Heule and Armin Biere. Compositional propositional proofs. In Martin Davis, Ansgar Fehnker, Annabelle McIver, and Andrei Voronkov, editors, *Logic for Programming, Artificial Intelligence, and Reasoning - 20th International Conference, LPAR-20 2015, Suva, Fiji, November 24-28, 2015, Proceedings*, volume 9450 of *Lecture Notes in Computer Science*, pages 444–459. Springer, 2015.

[16] Marijn J. H. Heule and Armin Biere. What a difference a variable makes. In Dirk Beyer and Marieke Huisman, editors, *Tools and Algorithms for the Construction and Analysis of Systems - 24th International Conference, TACAS 2018, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2018, Thessaloniki, Greece, April 14-20, 2018, Proceedings, Part II*, volume 10806 of *Lecture Notes in Computer Science*, pages 75–92. Springer, 2018.

[17] Marijn J. H. Heule, Benjamin Kiesl, and Armin Biere. Short proofs without new variables. In Leonardo de Moura, editor, *Automated Deduction - CADE 26 - 26th International Conference on Automated Deduction, Gothenburg, Sweden, August 6-11, 2017, Proceedings*, volume 10395 of *Lecture Notes in Computer Science*, pages 130–147. Springer, 2017.

[18] Marijn J. H. Heule, Benjamin Kiesl, Martina Seidl, and Armin Biere. Pruning through satisfaction. In Ofer Strichman and Rachel Tzoref-Brill, editors, *Hardware and Software: Verification and Testing - 13th International Haifa Verification Conference, HVC 2017, Haifa, Israel, November 13-15, 2017, Proceedings*, volume 10629 of *Lecture Notes in Computer Science*, pages 179–194. Springer, 2017.

[19] Matti Järvisalo, Marijn Heule, and Armin Biere. Inprocessing rules. In Bernhard Gramlich, Dale Miller, and Uli Sattler, editors, *Automated Reasoning - 6th International Joint Conference, IJCAR 2012, Manchester, UK, June 26-29, 2012. Proceedings*, volume 7364 of *Lecture Notes in Computer Science*, pages 355–370. Springer, 2012.

[20] Benjamin Kiesl, Adrián Rebola-Pardo, and Marijn J. H. Heule. Extended resolution simulates DRAT. In Didier Galmiche, Stephan Schulz, and Roberto Sebastiani, editors, *Automated Reasoning - 9th International Joint Conference, IJCAR 2018, Held as Part of the Federated Logic Conference, FloC 2018, Oxford, UK, July 14-17, 2018, Proceedings*, volume 10900 of *Lecture Notes in Computer Science*, pages 516–531. Springer, 2018.

[21] Hans Kleine Büning and Theodor Lettmann. *Aussagenlogik - Deduktion und Algorithmen.* Leitfäden und Monographien der Informatik. Teubner, 1994.

[22] J. Krajicek, J. Krajíček, M.I.J. Krajicek, G.C. Rota, B. Doran, P. Flajolet, M. Ismail, T.Y. Lam, and E. Lutwak. *Bounded Arithmetic, Propositional Logic and Complexity Theory.* Encyclopedia of Mathematics and its Applications. Cambridge University Press, 1995.

[23] Jan Krajícek and Pavel Pudlák. Some consequences of cryptographical conjectures for $s^1{}_2$ and EF. *Inf. Comput.*, 140(1):82–94, 1998.

[24] Norbert Manthey, Marijn Heule, and Armin Biere. Automated reencoding of boolean formulas. In Armin Biere, Amir Nahir, and Tanja E. J. Vos, editors, *Hardware and Software: Verification and Testing - 8th International Haifa Verification Conference, HVC 2012, Haifa, Israel, November 6-8, 2012. Revised Selected Papers*, volume 7857 of *Lecture Notes in Computer Science*, pages 102–117. Springer, 2012.

[25] Norbert Manthey and Tobias Philipp. Formula simplifications as DRAT derivations. In Carsten Lutz and Michael Thielscher, editors, *KI 2014: Advances in Artificial Intelligence - 37th Annual German Conference on AI, Stuttgart, Germany, September 22-26, 2014. Proceedings*, volume 8736 of *Lecture Notes in Computer Science*, pages 111–122. Springer, 2014.

[26] Kenneth L. McMillan. An interpolating theorem prover. In Kurt Jensen and Andreas Podelski, editors, *Tools and Algorithms for the Construction and Analysis of Systems, 10th International Conference, TACAS 2004, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2004, Barcelona, Spain, March 29 - April 2, 2004, Proceedings*, volume 2988 of *Lecture Notes in Computer Science*, pages 16–30. Springer, 2004.

[27] Kenneth L. McMillan. Interpolants and symbolic model checking. In Byron Cook and Andreas

Podelski, editors, *Verification, Model Checking, and Abstract Interpretation, 8th International Conference, VMCAI 2007, Nice, France, January 14-16, 2007, Proceedings*, volume 4349 of *Lecture Notes in Computer Science*, pages 89–90. Springer, 2007.

[28] Tobias Philipp and Adrian Rebola-Pardo. DRAT proofs for XOR reasoning. In Loizos Michael and Antonis C. Kakas, editors, *Logics in Artificial Intelligence - 15th European Conference, JELIA 2016, Larnaca, Cyprus, November 9-11, 2016, Proceedings*, volume 10021 of *Lecture Notes in Computer Science*, pages 415–429, 2016.

[29] Tobias Philipp and Adrián Rebola-Pardo. Towards a semantics of unsatisfiability proofs with inprocessing. In Thomas Eiter and David Sands, editors, *LPAR-21, 21st International Conference on Logic for Programming, Artificial Intelligence and Reasoning, Maun, Botswana, May 7-12, 2017*, volume 46 of *EPiC Series in Computing*, pages 65–84. EasyChair, 2017.

[30] David A. Plaisted and Steven Greenbaum. A structure-preserving clause form translation. *J. Symb. Comput.*, 2(3):293–304, 1986.

[31] Adrián Rebola-Pardo. Unsatisfiability proofs in SAT solving with parity reasoning. Master's thesis, TU Dresden, 2015.

[32] Adrian Rebola-Pardo and Armin Biere. Two flavors of DRAT. In *Pragmatics of SAT 2018*, 2018.

[33] Matthias Schlaipfer and Georg Weissenbacher. Labelled interpolation systems for hyper-resolution, clausal, and local proofs. *J. Autom. Reasoning*, 57(1):3–36, 2016.

[34] João P. Marques Silva and Karem A. Sakallah. GRASP: A search algorithm for propositional satisfiability. *IEEE Trans. Computers*, 48(5):506–521, 1999.

[35] Mate Soos, Karsten Nohl, and Claude Castelluccia. Extending SAT solvers to cryptographic problems. In Oliver Kullmann, editor, *Theory and Applications of Satisfiability Testing - SAT 2009, 12th International Conference, SAT 2009, Swansea, UK, June 30 - July 3, 2009. Proceedings*, volume 5584 of *Lecture Notes in Computer Science*, pages 244–257. Springer, 2009.

[36] Martin Suda and Bernhard Gleiss. Local soundness for QBF calculi. In Olaf Beyersdorff and Christoph M. Wintersteiger, editors, *Theory and Applications of Satisfiability Testing - SAT 2018 - 21st International Conference, SAT 2018, Held as Part of the Federated Logic Conference, FloC 2018, Oxford, UK, July 9-12, 2018, Proceedings*, volume 10929 of *Lecture Notes in Computer Science*, pages 217–234. Springer, 2018.

[37] Alasdair Urquhart. Hard examples for resolution. *J. ACM*, 34(1):209–219, 1987.

[38] Alasdair Urquhart. The symmetry rule in propositional logic. *Discrete Applied Mathematics*, 96-97:177–193, 1999.

[39] Dirk van Dalen. *Logic and structure (3. ed.)*. Universitext. Springer, 1994.

[40] Georg Weissenbacher. Interpolant strength revisited. In *SAT*, volume 7317 of *Lecture Notes in Computer Science*, pages 312–326. Springer, 2012.

[41] Nathan Wetzler, Marijn Heule, and Warren A. Hunt Jr. Drat-trim: Efficient checking and trimming using expressive clausal proofs. In Carsten Sinz and Uwe Egly, editors, *Theory and Applications of Satisfiability Testing - SAT 2014 - 17th International Conference, Held as Part of the Vienna Summer of Logic, VSL 2014, Vienna, Austria, July 14-17, 2014. Proceedings*, volume 8561 of *Lecture Notes in Computer Science*, pages 422–429. Springer, 2014.

[42] Edward Zulkoski, Ruben Martins, Christoph M. Wintersteiger, Jia Hui Liang, Krzysztof Czarnecki, and Vijay Ganesh. The effect of structural measures and merges on SAT solver performance. In John N. Hooker, editor, *Principles and Practice of Constraint Programming - 24th International Conference, CP 2018, Lille, France, August 27-31, 2018, Proceedings*, volume 11008 of *Lecture Notes in Computer Science*, pages 436–452. Springer, 2018.