A Unifying Principle for Clause Elimination in First-Order Logic[⋆]

Benjamin Kiesl and Martin Suda

Institute of Information Systems, Vienna University of Technology

Abstract. Preprocessing techniques for formulas in conjunctive normal form play an important role in first-order theorem proving. To speed up the proving process, these techniques simplify a formula without affecting its satisfiability or unsatisfiability. In this paper, we introduce the principle of implication modulo resolution, which allows us to lift several preprocessing techniques—in particular, several clause-elimination techniques—from the SAT-solving world to first-order logic. We analyze confluence properties of these new techniques and show how implication modulo resolution yields short soundness proofs for the existing first-order techniques of predicate elimination and blocked-clause elimination.

1 Introduction

Automatic theorem provers often have to deal with formulas that contain a considerable amount of redundant information. To speed up the proving process, they therefore usually employ dedicated preprocessing methods that aim at simplifying formulas as much as possible [1, 2]. Since most provers are based on proof systems that require formulas to be in conjunctive normal form (CNF), preprocessing techniques operating on the clause level play a particularly important role. Research on SAT and on quantified Boolean formulas has given rise to a wide variety of CNF preprocessing techniques that significantly improve the performance of modern solvers [3], but for many of these techniques it was not clear whether they could be lifted to the level of first-order logic.

In this paper, we address this issue and introduce the principle of *implication modulo resolution*—a first-order generalization of *quantified implied outer resolvents* as introduced by Heule et al. [4] in the context of quantified Boolean formulas. Informally, a clause C is *implied modulo resolution* by a CNF formula F (which can be seen as a set of clauses) if C contains a literal such that all resolvents upon this literal are implied by $F \setminus \{C\}$. Here, by *all resolvents* we mean all resolvents with clauses in $F \setminus \{C\}$. In other words, although $F \setminus \{C\}$ might not necessarily *imply* the clause C, it implies all the conclusions that can

⋆ This work has been supported by the Austrian Science Fund (FWF) under projects W1255-N23, S11403-N23, and S11409-N23, by the ERC Starting Grant 2014 SYMCAR 639270, and by the National Science Foundation (NSF) under grant number CCF-1618574.

be derived with C via resolution upon one of its literals. We show that this suffices to ensure that C can be removed from F without affecting the satisfiability or unsatisfiability of F.

The importance of implication modulo resolution lies in the fact that it allows us to construct soundness proofs for numerous preprocessing techniques. We therefore use the principle of implication modulo resolution to lift several SAT-preprocessing techniques to first-order logic without equality. These techniques, which have not been available in first-order logic so far, include clause-elimination procedures for *covered clauses* (CC) [5], *asymmetric tautologies* (AT) [6], *resolution asymmetric tautologies* (RAT) [7], and *resolution subsumed clauses* (RS) [7]. Moreover, we show how the use of implication modulo resolution yields short soundness proofs for the existing preprocessing techniques of *blocked-clause elimination* [8, 9] and *predicate elimination* [2], again in the restricted case of first-order logic *without* equality.

Covered clauses are a generalization of the above-mentioned blocked clauses (which we discuss briefly in Section 4). To detect whether a clause is covered, one first adds a number of so-called *covered literals* to it and then checks whether the resulting clause is a blocked clause. Covered-clause elimination is more powerful than blocked-clause elimination in the sense that it implicitly removes all blocked clauses from a formula. As blocked-clause elimination leads to significant performance improvements of first-order theorem provers [8] and since the elimination of covered clauses has been shown to speed up modern SAT solvers [10], we expect covered-clause elimination to further boost prover performance.

Asymmetric tautologies and resolution asymmetric tautologies owe their popularity to the fact that their addition and elimination can simulate most of the reasoning techniques employed by state-of-the-art SAT solvers [7]. Because of this, they provide the basis for the well-known DRAT proof system [11], which is the de-facto standard for unsatisfiability proofs in practical SAT solving. Finally, the elimination of resolution subsumed clauses is another promising technique from the SAT world whose soundness on the first-order level can be easily shown using the principle of implication modulo resolution.

The main contributions of this paper are as follows: (1) We introduce the principle of implication modulo resolution. (2) We use implication modulo resolution to lift several clause-elimination techniques from the SAT world to first-order logic. (3) We show how implication modulo resolution yields short soundness proofs for existing preprocessing techniques from the literature. (4) We analyze confluence properties of the preprocessing techniques.

2 Preliminaries

We assume the reader to be familiar with the basics of first-order logic. As usual, formulas of a first-order language $\mathcal{L}$ are built using predicate symbols, function symbols, and constants from some given alphabet together with logical connectives, quantifiers, and variables. We use the letters $P, Q, R, S, \ldots$ as predicate symbols and the letters $f, g, h, \ldots$ as non-constant function symbols. Moreover, we use the letters $a, b, c, \ldots$ for constants and the letters $x, y, z, u, v, \ldots$ for variables (possibly with subscripts). An expression (i.e., a term, literal, formula, etc.) is *ground* if it contains no variables.

A *literal* is an atom or the negation of an atom, and a disjunction of literals is a *clause*. For a literal L, we define $\bar{L} = \neg P$ if $L = P$ and $\bar{L} = P$ if $L = \neg P$, where P is an atom. In the former case, L is of *positive polarity*; in the latter case, it is of *negative polarity*. If not stated otherwise, formulas are assumed to be in conjunctive normal form (CNF), i.e., a conjunction of clauses. Without loss of generality, clauses are assumed to be variable disjoint. Variables occurring in a CNF formula are implicitly universally quantified. We treat CNF formulas as sets of clauses and clauses as multisets of literals. A clause is a *tautology* if it contains both L and $\bar{L}$ for some literal L.

Regarding the semantics, we use the standard notions of *interpretation*, *model*, *validity*, *satisfiability*, and *logical equivalence*. We say that two formulas are *equisatisfiable* if they are either both satisfiable or both unsatisfiable. A *propositional assignment* is a mapping from ground atoms to the truth values 1 (*true*) and 0 (*false*). Accordingly, a set of ground clauses is *propositionally satisfiable* if there exists a propositional assignment that satisfies F under the usual semantics for the logical connectives. We sometimes write assignments as sequences of literals where a positive (negative) polarity of a literal indicates that the truth value 1 (0, respectively) is assigned to the literal's atom.

A *substitution* is a mapping from variables to terms that agrees with the identity function on all but finitely many variables. Let σ be a substitution. The domain, $dom(\sigma)$, of σ is the set of variables for which $\sigma(x) \neq x$. The range, $ran(\sigma)$, of σ is the set $\{\sigma(x) \mid x \in dom(\sigma)\}$. We denote the *inverse substitution* of σ, which is just the inverse function of σ, by σ^{-1}. A substitution is *ground* if its range consists only of ground terms. As common, $E\sigma$ denotes the result of applying σ to the expression E. If $E\sigma$ is ground, it is a *ground instance* of E. Juxtaposition of substitutions denotes their composition, i.e., $x\sigma\tau$ stands for $\tau(\sigma(x))$. The substitution σ is a *unifier* of the expressions $E_1, \ldots, E_n$ if $E_1\sigma = \cdots = E_n\sigma$. For substitutions σ and τ, we say that σ is *more general* than τ if there exists a substitution λ such that $\sigma\lambda = \tau$. Furthermore, σ is a *most general unifier* (*mgu*) of $E_1, \ldots, E_n$ if, for every unifier τ of $E_1, \ldots, E_n$, σ is more general than τ. It is well-known that whenever a set of expressions is unifiable, there exists an idempotent most general unifier of this set. In the rest of the paper, we use a popular variant of Herbrand's Theorem [12]:

Theorem 1. *A formula F is satisfiable if and only if every finite set of ground instances of clauses in F is propositionally satisfiable.*

Next, we introduce a formal notion of clause redundancy. Intuitively, a clause C is redundant w.r.t. a formula F if its removal from F does not affect the satisfiability or unsatisfiability of F [4]:

Definition 1. *A clause C is* redundant *w.r.t. a formula F if F and $F \setminus \{C\}$ are equisatisfiable.*

Note that this notion of redundancy does not require *logical* equivalence of F and $F \setminus \{C\}$, and that it differs from other well-known redundancy notions such as the one of Bachmair and Ganzinger that is usually employed within the context of ordered resolution [13]. It provides the basis for clause-elimination procedures. Note also that the redundancy of a clause C w.r.t. a formula F can be shown by proving that the satisfiability of $F \setminus \{C\}$ implies the satisfiability of F.

Finally, given two clauses $C = L_1 \vee \cdots \vee L_k \vee C'$ and $D = N_1 \vee \cdots \vee N_l \vee D'$ such that the literals $L_1, \dots, L_k, \bar{N}_1, \dots, \bar{N}_l$ are unifiable by an *mgu* σ, the clause $C'\sigma \vee D'\sigma$ is said to be a *resolvent* of C and D. If $k = l = 1$, it is a *binary resolvent* of C and D upon L_1.

3 Implication Modulo Resolution

In this section, we introduce the central concept of this paper—the principle of *implication modulo resolution* for first-order logic. We use the results of this section in subsequent sections to prove the soundness of various first-order preprocessing techniques. The definition of implication modulo resolution relies on the notion of L-resolvents:

Definition 2. *Given two clauses $C = L \vee C'$ and $D = N_1 \vee \cdots \vee N_l \vee D'$ such that the literals $L, \bar{N}_1, \dots, \bar{N}_l$ are unifiable by an mgu σ, the clause $C'\sigma \vee D'\sigma$ is called* L-resolvent *of C and D.*

Example 1. Let $C = P(x) \vee Q(x)$, $D = \neg P(y) \vee \neg P(z) \vee R(y, z)$, and $L = P(x)$. Then, the substitution $\sigma = \{y \mapsto x, z \mapsto x\}$ is an *mgu* of $P(x)$, $P(y)$, and $P(z)$. Therefore, $Q(x) \vee R(x, x)$ is an L-resolvent of C and D. □

Before we next define the principle of implication modulo resolution, we want to highlight that whenever we say that a formula F *implies* a clause C, we mean that every model of F is a model of C, that is, $F \models C$.

Definition 3. *A clause C is* implied modulo resolution *upon $L \in C$ by a formula F if all L-resolvents of C, with clauses in $F \setminus \{C\}$, are implied by $F \setminus \{C\}$.*

We say that a clause C is implied modulo resolution by F if F implies C modulo resolution upon one of its literals. A simple example for clauses that are implied modulo resolution are clauses with *pure literals*. A pure literal is a literal whose predicate symbol occurs in only one polarity in the whole formula. Since there are no resolvents upon such a literal, the containing clause is trivially implied modulo resolution. The following example is a little more involved:

Example 2. Let $C = P(x) \vee Q(x)$ and

$$F = \{P(x) \vee Q(x),\ \neg P(y) \vee R(y),\ R(z) \vee S(z),\ \neg S(u) \vee Q(u)\}.$$

There is one $P(x)$-resolvent of C, namely $Q(x) \vee R(x)$, obtained by resolving C with $\neg P(y) \vee R(y)$. Clearly, this resolvent is implied by the clauses $R(z) \vee S(z)$ and $\neg S(u) \vee Q(u)$. Therefore, F implies C modulo resolution upon $P(x)$. □

In the following, we prove that implication modulo resolution ensures redundancy, i.e., if a clause C is implied modulo resolution by a formula F, then C is redundant w.r.t. F. The proof relies on Herbrand's Theorem (Theorem 1), which tells us that a formula F is satisfiable if and only if all finite sets of ground instances of clauses in F are propositionally satisfiable.

To prove that the satisfiability of $F \setminus \{C\}$ implies the satisfiability of F, we proceed as follows: Given a finite set of ground instances of clauses in F, we can obtain a satisfying propositional assignment of this set from an assignment that satisfies all the ground instances of clauses in $F \setminus \{C\}$. The latter assignment is guaranteed to exist because $F \setminus \{C\}$ is satisfiable. The key idea behind the modification of this assignment is to *flip* (interchange) the truth values of certain ground literals. We illustrate this on the following example:

Example 3. Consider C and F from Example 2, let $C' = P(a) \vee Q(a)$ be a ground instance of C, and $F' = \{P(a) \vee Q(a), \neg P(a) \vee R(a),\ R(a) \vee S(a),\ \neg S(a) \vee Q(a)\}$ a finite set of ground instances of F (in fact, F' is even a ground instance of F). Clearly, $F' \setminus \{C'\}$ is propositionally satisfied by the assignment $\alpha = \neg P(a) R(a) \neg S(a) \neg Q(a)$, but α falsifies C'. However, we can turn α into a satisfying assignment of C' by flipping the truth value of $P(a)$—the instance of the literal upon which C is implied modulo resolution. The resulting assignment $\alpha' = P(a) R(a) \neg S(a) \neg Q(a)$ could possibly falsify the clause $\neg P(a) \vee R(a)$ since it contains $\neg P(a)$ which is not satisfied anymore. But, the clause stays true since $R(a)$ is satisfied by α'. Therefore, α' satisfies F'. □

In the above example, it is not a coincidence that $\neg P(a) \vee R(a)$ is still satisfied after flipping the truth value of $P(a)$. The intuitive explanation is as follows: The clause $Q(a) \vee R(a)$ is a ground instance of the $P(x)$-resolvent $Q(x) \vee R(x)$ (of C and $\neg P(y) \vee R(y)$) which is implied by $F \setminus \{C\}$. Therefore, since α satisfies all the ground instances of $F \setminus \{C\}$, it should also satisfy $Q(a) \vee R(a)$. But, since α does not satisfy $Q(a)$ (because α falsifies $C' = P(a) \vee Q(a)$), it must satisfy $R(a)$, and so it satisfies $\neg P(a) \vee R(a)$. Finally, since α' disagrees with α only on $P(a)$, it also satisisfies $R(a)$. The following lemma formalizes this observation:

Lemma 2. *Let C be a clause that is implied modulo resolution upon L by F. Let furthermore α be an assignment that propositionally satisfies all ground instances of clauses in $F \setminus \{C\}$ but falsifies a ground instance $C\lambda$ of C. Then, the assignment α', obtained from α by flipping the truth value of $L\lambda$, still satisfies all ground instances of clauses in $F \setminus \{C\}$.*

Proof. Let $D\tau$ be a ground instance of a clause $D \in F \setminus \{C\}$ and suppose α satisfies $D\tau$. If $D\tau$ does not contain $\bar{L}\lambda$, it is trivially satisfied by α'. Assume therefore that $\bar{L}\lambda \in D\tau$ and let $N_1, \dots, N_l$ be all the literals in D such that $N_i\tau = \bar{L}\lambda$ for $1 \leq i \leq l$. Then, the substitution $\lambda\tau = \lambda \cup \tau$ (note that C and D are variable disjoint by assumption) is a unifier of $L, \bar{N}_1, \dots, \bar{N}_l$. Hence, $R = (C \setminus \{L\})\sigma \vee (D \setminus \{N_1, \dots, N_l\})\sigma$, with σ being an *mgu* of $L, \bar{N}_1, \dots, \bar{N}_l$, is an L-resolvent of C and thus implied by $F \setminus \{C\}$.

As σ is most general, it follows that there exists a substitution γ such that $\sigma\gamma = \lambda\tau$. Therefore,

$$\begin{aligned} &(C \setminus \{L\})\sigma\gamma \vee (D \setminus \{N_1, \dots, N_l\})\sigma\gamma \\ = &(C \setminus \{L\})\lambda\tau \vee (D \setminus \{N_1, \dots, N_l\})\lambda\tau \\ = &(C \setminus \{L\})\lambda \quad \vee (D \setminus \{N_1, \dots, N_l\})\tau \end{aligned}$$

is a ground instance of R and so it must be satisfied by α. Thus, since α falsifies $C\lambda$, it must satisfy a literal $L'\tau \in (D \setminus \{N_1, \dots, N_l\})\tau$. But, as all the literals in $(D \setminus \{N_1, \dots, N_l\})\tau$ are different from $\bar{L}\lambda$, flipping the truth value of $L\lambda$ does not affect the truth value of $L'\tau$. It follows that α' satisfies $L'\tau$ and thus it satisfies $D\tau$. □

We can therefore satisfy a ground instance $C\lambda$ of C without falsifying ground instances of clauses in $F \setminus \{C\}$, by flipping the truth value of $L\lambda$—the ground instance of the literal L upon which C is implied modulo resolution. Still, as the following example shows, there could be other ground instances of C that contain the complement $\bar{L}\lambda$ of $L\lambda$:

Example 4. Suppose some formula F implies the clause $C = \neg P(x) \vee P(f(x))$ modulo resolution upon the literal $P(f(x))$ and consider two possible ground instances $C_1 = \neg P(a) \vee P(f(a))$ and $C_2 = \neg P(f(a)) \vee P(f(f(a)))$ of C. The assignment $P(a)\neg P(f(a))\neg P(f(f(a)))$ falsifies C_1, but we can satisfy C_1 by flipping the truth value of $P(f(a))$—the ground instance of $P(f(x))$—to obtain the assignment $P(a)P(f(a))\neg P(f(f(a)))$. However, by flipping the truth value of $P(f(a))$, we falsified the other ground instance C_2 of C. □

That this is not a serious problem is shown in the proof of our main result below. The key idea is to repeatedly satisfy ground instances of the literal upon which the clause is implied modulo resolution. In the above example, for instance, we can continue by also flipping the truth value of $P(f(f(a))$ to obtain a satisfying assignment of both C_1 and C_2.

Theorem 3. *If a clause C is implied modulo resolution by a formula F, it is redundant w.r.t. F.*

Proof. Assume that F implies C modulo resolution upon $L \in C$ and that $F \setminus \{C\}$ is satisfiable. We show that F is satisfiable. By Herbrand's theorem (Theorem 1), it suffices to show that every finite set of ground instances of clauses in F is propositionally satisfiable. Let therefore F' and F_C be finite sets of ground instances of clauses in $F \setminus \{C\}$ and $\{C\}$, respectively. Since $F \setminus \{C\}$ is satisfiable, there exists an assignment α that propositionally satisfies all ground instances of clauses in $F \setminus \{C\}$ and thus it clearly satisfies F'. Assume now that α falsifies some ground instances of C that are contained in F_C.

By Lemma 2, for every falsified ground instance $C\lambda$ of C, we can turn α into a satisfying assignment of $C\lambda$ by flipping the truth value of $L\lambda$, and this flipping does not falsify any ground instances of clauses in $F \setminus \{C\}$. The only clauses that could possibly be falsified are other ground instances of C that contain the literal $\bar{L}\lambda$. But, once an instance $L\tau$ of L is true in a ground instance $C\tau$ of C, $L\tau$ cannot (later) be falsified by making other instances of L true. As there are only finitely many clauses in F_C, we can therefore turn α into a satisfying assignment of $F' \cup F_C$ by repeatedly making ground instances of C true by flipping the truth values of their instances of L. Hence, all finite sets of ground instances of clauses in F are propositionally satisfiable and so F is satisfiable. $\square$

For example, the clause C in Example 2 is redundant w.r.t. F since it is implied modulo resolution by F. In what follows, we use Theorem 3 to prove soundness of several first-order preprocessing techniques. We start with blocked-clause elimination, as both resolution asymmetric tautologies and covered clauses (which we introduce later) can be seen as generalizations of blocked clauses.

4 Blocked Clauses

Blocked clauses have been introduced by Kullmann [14], and their elimination significantly improves the performance of SAT [9] and QSAT solvers [15, 16]. Also the first-order variant of blocked-clause elimination speeds up automatic theorem provers, especially on satisfiable formulas [8]. In propositional logic, a clause C is *blocked* in a formula F if it contains a literal L such that all binary resolvents of C upon L, with clauses in $F \setminus \{C\}$, are tautologies. In first-order logic, the notion of binary resolvents is replaced by L-resolvents [8]:

Definition 4. *A clause C is* blocked *by a literal $L \in C$ in a formula F if all L-resolvents of C, with clauses in $F \setminus \{C\}$, are tautologies.*

Example 5. Let $C = P(x) \lor \neg Q(x)$ and $F = \{P(x) \lor \neg Q(x),\ \neg P(y) \lor Q(y)\}$. There is only one $P(x)$-resolvent of C, namely the tautology $\neg Q(x) \lor Q(x)$, obtained by using the *mgu* $\sigma = \{y \mapsto x\}$. Therefore, C is blocked in F. $\square$

Since tautologies are trivially implied by every formula, blocked clauses are implied modulo resolution. The redundancy of blocked clauses, and therefore the soundness of blocked-clause elimination, is thus a consequence of the fact that implication modulo resolution ensures redundancy (Theorem 3):

Theorem 4. *If a clause is blocked in a formula F, it is redundant w.r.t. F.*

5 Asymmetric Tautologies and RATs

In this section, we first discuss the propositional notions of asymmetric tautologies and resolution asymmetric tautologies before lifting them to first-order logic. We start with asymmetric tautologies, which we use later to define resolution asymmetric tautologies. An asymmetric tautology is a clause that can be turned into a tautology by repeatedly adding so-called *asymmetric literals* to it. In propositional logic, a literal L is an *asymmetric literal* w.r.t. a clause C in a formula F if there exists a clause $D \lor \bar{L} \in F \setminus \{C\}$ such that D subsumes

C, i.e., $D \subseteq C$. The addition of an asymmetric literal L to a clause C yields a clause that is logically equivalent in the sense that $F \setminus \{C\} \models (C \equiv C \vee L)$ [6]. Consider, for instance, the following example:

Example 6. Let $C = P \vee Q$ and $F = \{P \vee Q,\ Q \vee R,\ \neg R \vee S,\ P \vee \neg R \vee \neg S\}$. Since the subclause Q of $Q \vee R$ subsumes C, the literal $\neg R$ is an asymmetric literal w.r.t. C. We thus add it to C to obtain $C_1 = P \vee Q \vee \neg R$. We then use $\neg R \vee S$ to add $\neg S$ to C_1 and obtain $C_2 = P \vee Q \vee \neg R \vee \neg S$. Finally, we use $P \vee \neg R \vee \neg S$ to add $\neg P$ to C_2, and so we end up with $C_3 = P \vee Q \vee \neg R \vee \neg S \vee \neg P$, which is a tautology. It follows that C is an asymmetric tautology in F. Moreover, by transitivity, $F \setminus \{C\} \models (C \equiv C_3)$ and thus C is redundant w.r.t. F. $\square$

In first-order logic, a clause C *subsumes* a clause D if there exists a substitution λ such that $C\lambda \subseteq D$. This motivates the following first-order variants of asymmetric literals and asymmetric tautologies.

Definition 5. *A literal L is an* asymmetric literal *w.r.t. a clause C in a formula F if there exist a clause $D \vee \bar{L}' \in F \setminus \{C\}$ and a substitution λ such that $D\lambda \subseteq C$ and $L = \bar{L}'\lambda$.*

Example 7. Consider the clause $C = P(x) \vee Q(x) \vee R(x)$ and the formula $F = \{P(x) \vee Q(x) \vee R(x),\ P(y) \vee Q(y) \vee \neg S(y)\}$. Then, $S(x)$ is an asymmetric literal w.r.t. C in F since, for $\lambda = \{y \mapsto x\}$, $(P(y) \vee Q(y))\lambda \subseteq C$ and $S(x) = S(y)\lambda$. $\square$

First-order asymmetric-literal addition is harmless, because the original clause C can be obtained from $C \vee L$ and $D \vee \bar{L}'$ via resolution, as shown in the proof of the following lemma:

Lemma 5. *Let F be a formula, C a clause, and L an asymmetric literal w.r.t. C in F. Then, $F \setminus \{C\} \models (C \equiv C \vee L)$.*

Proof. Clearly, $C \rightarrow C \vee L$ is valid. It therefore suffices to prove that C is implied by $(F \setminus \{C\}) \cup \{C \vee L\}$. Since L is an asymmetric literal w.r.t. C in F, there exist a clause $D \vee L' \in F \setminus \{C\}$ and a substitution λ such that $D\lambda \subseteq C$ and $\bar{L}'\lambda = L$. But then C is a binary resolvent of $C \vee L$ and $D\lambda \vee L'\lambda$ upon L. It follows that C is implied by $(F \setminus \{C\}) \cup \{C \vee L\}$. $\square$

An asymmetric tautology is now a clause that can be turned into a tautology by repeatedly adding asymmetric literals (*asymmetric-literal addition*, ALA):

Definition 6. *A clause C is an* asymmetric tautology *in a formula F if there exists a sequence $L_1, \dots, L_n$ of literals such that each L_i is an asymmetric literal w.r.t. $C \vee L_1 \vee \dots \vee L_{i-1}$ in $F \setminus \{C\}$ and $C \vee L_1 \vee \dots \vee L_n$ is a tautology.*

Example 8. Consider the clause $C = Q(x) \vee R(x)$ and the following formula $F = \{Q(x) \vee R(x),\ R(z) \vee S(z),\ \neg S(u) \vee Q(u)\}$. The subclause $R(z)$ of $R(z) \vee S(z)$ subsumes $R(x)$ via $\{z \mapsto x\}$ and so $\neg S(x)$ is an asymmetric literal w.r.t. C. We add it and obtain the clause $Q(x) \vee R(x) \vee \neg S(x)$. After this, $\neg S(u)$ subsumes $\neg S(x)$ via $\{u \mapsto x\}$ and thus $\neg Q(x)$ can be added to obtain the tautology $Q(x) \vee R(x) \vee \neg S(x) \vee \neg Q(x)$. Thus, C is an asymmetric tautology in F. $\square$

Note that in automatic theorem proving, we prefer short clauses over long ones, since the short clauses are usually stronger. Therefore, when performing asymmetric-tautology elimination, the asymmetric-literal additions are not meant to be permanent: We first add the literals and then test whether the resulting clause is a tautology. If so, we remove the clause; if not, we undo the asymmetric-literal additions to shrink the clause back to its original size. We next show that asymmetric tautologies are implied:

Theorem 6. *If C is an asymmetric tautology in F, it is implied by $F \setminus \{C\}$.*

Proof. Suppose C is an asymmetric tautology in F, i.e., there exists a sequence $L_1, \dots, L_n$ of literals such that each L_i is an asymmetric literal w.r.t. the clause $C \vee L_1 \vee \cdots \vee L_{i-1}$ in $F \setminus \{C\}$ and $C \vee L_1 \vee \cdots \vee L_n$ is a tautology. By the repeated application of Lemma 5 (an easy induction argument), it follows that $F \setminus \{C\} \models (C \equiv C \vee L_1 \vee \cdots \vee L_n)$. But then, since $C \vee L_1 \vee \cdots \vee L_n$ is a tautology, it trivially holds that $F \setminus \{C\} \models C \vee L_1 \vee \cdots \vee L_n$. Therefore, $F \setminus \{C\} \models C$. □

Unlike in propositional logic, the first-order variant of asymmetric-literal addition is not guaranteed to terminate. Consider the following example:

Example 9. Let $C = P(a)$ and $F = \{P(x) \vee \neg P(f(x))\}$. Then, since $P(x)$ subsumes $P(a)$ via $\lambda = \{x \mapsto a\}$, we can add the asymmetric literal $P(f(a))$ to obtain $P(a) \vee P(f(a))$. After this, we can add $P(f(f(a))$ via $\lambda = \{x \mapsto f(a)\}$, then $P(f(f(f(a))))$ and so on. This can be repeated infinitely many times. □

A resolution asymmetric tautology in first-order logic is then a clause C that contains a literal L such that all L-resolvents of C are asymmetric tautologies:

Definition 7. *A clause C is a* resolution asymmetric tautology (RAT) *on a literal $L \in C$ w.r.t. a formula F if all L-resolvents of C, with clauses in $F \setminus \{C\}$, are asymmetric tautologies in $F \setminus \{C\}$.*

Example 10. Consider the clause $C = P(x) \vee Q(x)$ and the following formula $F = \{P(x) \vee Q(x),\ \neg P(y) \vee R(y),\ R(z) \vee S(z),\ \neg S(u) \vee Q(u)\}$ (cf. Example 2). There is one $P(x)$-resolvent of C, namely $Q(x) \vee R(x)$. The formula $F \cup \{Q(x) \vee R(x)\}$ is a superset of the formula from Example 8 in which $Q(x) \vee R(x)$ is an asymmetric tautology. Thus, $Q(x) \vee R(x)$ is also an asymmetric tautology here: The literal $R(z)$ subsumes $R(x)$ via $\{z \mapsto x\}$ and so $\neg S(x)$ is an asymmetric literal w.r.t. $Q(x) \vee R(x)$. We add it to obtain $Q(x) \vee R(x) \vee \neg S(x)$. After this, $\neg S(u)$ subsumes $\neg S(x)$ via $\{u \mapsto x\}$ and so $\neg Q(x)$ can be added to obtain the tautology $Q(x) \vee R(x) \vee \neg S(x) \vee \neg Q(x)$. It follows that C is a RAT w.r.t. F. □

Theorem 7. *If a clause C is a* RAT *w.r.t. a formula F, then it is redundant w.r.t. F.*

Proof. Assume that C is a RAT w.r.t. F. Then, every L-resolvent of C with clauses in $F \setminus \{C\}$ is an asymmetric tautology in $F \setminus \{C\}$ and therefore, by Theorem 6, implied by $F \setminus \{C\}$. It follows that C is implied modulo resolution upon L by F and thus, by Theorem 3, C is redundant w.r.t. F. □

6 Covered Clauses

In this section, similar to the preceding one, we first recapitulate the notions of covered literals and covered clauses from propositional logic and then lift them to the first-order level. Informally, a clause C is *covered* in a propositional formula F, if the addition of so-called *covered literals* to C turns C into a blocked clause. A clause C *covers* a literal L' in F if C contains a literal L such that all non-tautological resolvents of C upon L contain L'. The crucial property of covered literals is, that they can be added to C without affecting satisfiability [5]. More precisely, given a formula F, a clause $C \in F$, and a literal L' that is covered by C in F, it holds that F and the formula F', obtained from F by replacing C with $C \vee L'$, are equisatisfiable.

Example 11. Consider the clause $C = P$ and the propositional formula $F = \{P,\ \neg P \vee \neg Q \vee R,\ \neg P \vee \neg Q \vee S\}$. There are two resolvents of C upon P, namely $\neg Q \vee R$ and $\neg Q \vee S$. As $\neg Q$ is contained in both resolvents, it is covered by C in F. Therefore, replacing C with $C \vee \neg Q$ in F does not affect satisfiability. □

We next introduce a first-order variant of covered literals. Our definition guarantees that covered-literal addition (CLA) has no effect on satisfiability:

Definition 8. *A clause C* covers *a literal L' in a formula F if C contains a literal L such that all non-tautological L-resolvents of C, with clauses in $F \cup \{C\}$, contain L'.*

Note that resolvents of C with itself are required to contain the literal L'. Moreover, when talking about resolvents of C with itself, we mean resolvents of C with an instance $C\tau$ of C, where τ is a renaming that maps the variables in C to fresh variables that do not occur in F.

Example 12. Consider the clause $C = P(f(x))$ and the formula

$$F = \{\neg P(y) \vee Q(y) \vee R(y),\ \neg P(z) \vee Q(z) \vee S(z)\}.$$

There are two $P(f(x))$-resolvents of C: $Q(f(x)) \vee R(f(x))$, obtained by using the *mgu* $\{y \mapsto f(x)\}$, and $Q(f(x)) \vee S(f(x))$, obtained by using the *mgu* $\{z \mapsto f(x)\}$. Since $Q(f(x))$ is contained in both resolvents, it is covered by C in F. □

As we will show below, the addition of a covered literal to the clause that covers it has no effect on satisfiability. The following example illustrates that this would not be the case if we did not require the covered literal to be contained in resolvents of the clause with itself:

Example 13. Consider the clause $C = \neg P(x) \vee P(f(x))$ and the formula $F = \{\neg P(x) \vee P(f(x)),\ \neg P(y) \vee Q(y),\ P(a),\ \neg Q(f(f(a))\}$. The literal $Q(f(x))$ is contained in the (only) $P(f(x))$-resolvent $\neg P(x) \vee Q(f(x))$ of C with clauses in F *that are different from C itself.* However, F is unsatisfiable whereas the formula F', obtained from F by replacing C with $C \vee Q(f(x))$, is satisfiable. □

Lemma 8. *If a clause C covers a literal L' in a formula F, then F and the formula F', obtained from F by replacing C with $C \vee L'$, are equisatisfiable.*

Proof. Assume that C covers L' in F, i.e., L' is contained in all non-tautological L-resolvents of C with clauses in F. First, we add $C\tau \vee L'\tau$ to F, with τ being a renaming that replaces the variables in $C \vee L'$ by fresh variables not occurring in F. Since $C\tau \vee L'\tau$ is subsumed by C, the formulas F and $F \cup \{C\tau \vee L'\tau\}$ are equisatisfiable. We next show that C is redundant w.r.t. $F \cup \{C\tau \vee L'\tau\}$ and that it can therefore by removed. To do so, we show that C is implied modulo resolution upon L by $F \cup \{C\tau \vee L'\tau\}$. As $F \cup \{C\tau \vee L'\tau\}$ and $F \cup \{C \vee L'\}$ are clearly equivalent, the claim then follows.

We show that all L-resolvents of C with clauses in F are implied by the formula $(F \setminus \{C\}) \cup \{C\tau \vee L'\tau\}$. Showing that the L-resolvents of C with $C\tau \vee L'\tau$ are also implied is done in a similar way. Since tautological L-resolvents are trivially implied, we consider only non-tautological ones. Let $C'\sigma \vee D'\sigma$ be a non-tautological L-resolvent of $C = C' \vee L$ with a clause $D = D' \vee N_1 \vee \dots \vee N_k \in F$, where σ is an (idempotent) *mgu* of the literals $L, \bar{N}_1, \dots, \bar{N}_k$. Since L' is covered by C in F, the resolvent $C'\sigma \vee D'\sigma$ contains L', and L' is of the form $P\sigma$ for some literal $P \in C' \vee D'$.

To prove that $C'\sigma \vee D'\sigma$ is implied by $(F \setminus \{C\}) \cup \{C\tau \vee L'\tau\}$, we show that it can be obtained from clauses in $(F \setminus \{C\}) \cup \{C\tau \vee L'\tau\}$ via resolution, instantiation, and factoring: Consider the clauses $C\tau \vee L'\tau = C'\tau \vee L\tau \vee L'\tau$ and $D = D' \vee N_1 \vee \dots \vee N_k$. Since the literals $L, \bar{N}_1, \dots, \bar{N}_k$ are unified by σ and since $dom(\tau^{-1}) \cap \text{var}(D) = \emptyset$, it follows that $L\tau$ and $\bar{N}_1, \dots, \bar{N}_k$ are unified by $\tau^{-1}\sigma$. Therefore, there exists an *mgu* σ' of $L\tau$ and $\bar{N}_1, \dots, \bar{N}_k$. Hence, the clause $(C'\tau \vee L'\tau \vee D')\sigma'$ is an $L\tau$-resolvent. Now, since σ' is most general, there exists a substitution γ such that $\sigma'\gamma = \tau^{-1}\sigma$. But then,

$$\begin{aligned}
&(C'\tau \vee L'\tau \vee D')\sigma'\gamma \\
&= (C'\tau \vee L'\tau \vee D')\tau^{-1}\sigma \\
&= C'\sigma \vee L'\sigma \vee D'\sigma,
\end{aligned}$$

from which we obtain $C'\sigma \vee D'\sigma$ by factoring, since $L' \in C'\sigma \vee D'\sigma$ and $L'\sigma = P\sigma\sigma = P\sigma = L'$. □

Similar to asymmetric-literal addition, the addition of covered literals in first-order logic is also not guaranteed to terminate. Consider the following example:

Example 14. Let $C = P(a)$ and $F = \{P(a), \neg P(x) \vee P(f(x))\}$. Then, there exists one $P(a)$-resolvent of C, namely $P(f(a))$. Therefore, $P(f(a))$ is covered by C and thus it can be added to C to obtain $C' = P(a) \vee P(f(a))$. Now, there is one $P(f(a))$-resolvent of C', namely $P(f(f(a)))$, and thus $P(f(f(a)))$ can be added. This addition of covered literals can be repeated infinitely often. □

Now, a clause C is covered in a formula F if the repeated addition of covered literals can turn it into a blocked clause. In the following, we denote by $F[C/D]$ the formula obtained from F by replacing the clause C with the clause D:

Definition 9. *A clause C is* covered *in a formula F if there exists a sequence $L_1, \dots, L_n$ of literals such that each L_i is covered by $C_{i-1} = C \vee L_1 \vee \cdots \vee L_{i-1}$ in $F[C/C_{i-1}]$ and C_n is blocked in $F[C/C_n]$.*

Example 15. Consider the clause $C = P(a) \vee \neg Q(a)$ which is contained in the formula $F = \{P(a) \vee \neg Q(a),\ \neg P(y) \vee R(y),\ \neg R(z) \vee Q(z)\}$. Although C is not blocked in F, we can add the literal $R(a)$ since it is contained in its only $P(a)$-resolvent, obtained by resolving with $\neg P(y) \vee R(y)$. The resulting clause $P(a) \vee \neg Q(a) \vee R(a)$ is then blocked by $R(a)$ since there is only the tautological $R(a)$-resolvent $P(a) \vee \neg Q(a) \vee Q(a)$, obtained by resolving with $\neg R(z) \vee Q(z)$. Therefore, C is covered in F. □

As in the case of asymmetric tautologies, the covered-literal additions used during covered-clause elimination are not meant to be permanent: We first add some covered literals and then test whether the resulting clause is blocked (and therefore covered). If so, we remove the clause; if not, we undo the literal additions.

Theorem 9. *If a clause C is covered in a formula F, it is redundant w.r.t. F.*

Proof. Assume that C is covered in F, i.e., we can add covered literals to C to obtain a clause C' that is blocked in F. Now, let F' be obtained from F by replacing C with C'. Then, by Lemma 8, F and F' are equisatisfiable. Moreover, since C' is blocked in F', it follows that $F' \setminus \{C'\}$ and F' are equisatisfiable. But then, as $F \setminus \{C\} = F' \setminus \{C'\}$, it follows that F and $F \setminus \{C\}$ are equisatisfiable and so C is redundant w.r.t. F. □

7 Resolution Subsumption and More

The redundancy notion of *resolution subsumption* (RS) from SAT [7] can also be straightforwardly lifted to first-order logic, where redundancy is again an immediate consequence of Theorem 3 since subsumption ensures implication:

Definition 10. *A clause C is* resolution subsumed (RS) *on a literal $L \in C$ in a formula F if all non-tautological L-resolvents of C, with clauses in $F \setminus \{C\}$, are subsumed in $F \setminus \{C\}$.*

Theorem 10. *If a clause is resolution subsumed in a formula F, then it is redundant w.r.t. F.*

With the methods presented in this paper, we can define even more types of redundant clauses that have been considered in the SAT literature. We can do so by combining asymmetric-literal addition or covered-literal addition with tautology or subsumption checks. These checks can be performed either directly on the clause or for all resolvents of the clause upon one of its literals. The latter can be seen as some kind of "look-ahead" via resolution. Fig. 1 illustrates possible combinations of techniques. Every path from the left to the right gives rise to a particular redundancy notion. We remark that ALA stands for asymmetric-literal addition and CLA stands for covered-literal addition.

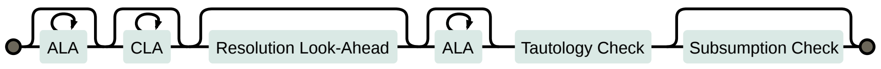

Fig. 1. Combination of Techniques to Obtain Redundancy Notions.

For instance, to detect whether a clause is an asymmetric tautology, we first perform some asymmetric-literal additions and then check whether the resulting clause is a tautology. Another example are blocked clauses, where we ask whether all L-resolvents of the clause are tautologies. Similarly, we can obtain covered clauses, resolution subsumed clauses, and resolution asymmetric tautologies via such combinations. This gives rise to various other types of clauses like *asymmetric blocked clauses*, *asymmetric subsumed clauses* [7], or *asymmetric covered clauses* [3]. The redundancy of these clauses follows from the results in this paper, most importantly from the principle of implication modulo resolution.

8 Predicate Elimination

In this section, we show how the principle of implication modulo resolution allows us to construct a short soundness proof for the predicate elimination technique of Khasidashvili and Korovin [2]. Predicate elimination is a first-order variant of variable elimination, which is successfully used during preprocessing and inprocessing in SAT solving [17]. The elimination of a predicate P from a formula F is computed as follows: First, we add all the non-tautological binary resolvents upon literals with predicate symbol P to F. After this, all original clauses containing P are removed. To guarantee that this procedure does not affect satisfiability, the original definition requires P to be *non-recursive*, meaning that it must not occur more than once per clause.

Theorem 11. *If a formula F' is obtained from a formula F by eliminating a non-recursive predicate P, then F and F' are equisatisfiable.*

Proof. Let F_P be obtained from F by adding all non-tautological resolvents upon P. Clearly, F_P and F are equivalent. Now, let C be a clause that contains a literal L with predicate symbol P. Since all non-tautological L-resolvents of C with clauses in $F_P \setminus \{C\}$ are contained in $F_P \setminus \{C\}$, C is implied modulo resolution by F_P and so it is redundant w.r.t. F_P. Thus, after removing from F_P all clauses containing P, the resulting formula F' and F_P are equisatisfiable. Therefore, F' and F are equisatisfiable. □

We want to highlight that Khasidashvili and Korovin [2] proved soundness of predicate elimination for first-order logic *with* equality while we restrict ourselves to first-order logic *without* equality.

9 Confluence Properties

In this section, we analyze confluence properties of the clause-elimination and literal-addition techniques discussed in this paper. Intuitively, confluence of a technique tells us that the order in which we perform the clause eliminations or the literal additions is not relevant to the final outcome of the technique.

To analyze confluence formally, we interpret our techniques as *abstract reduction systems* [18]. For instance, to analyze the confluence of a clause-elimination technique CE, we define the (reduction) relation $\rightarrow_{\mathsf{CE}}$ over formulas as follows: $F_1 \rightarrow_{\mathsf{CE}} F_2$ if and only if the technique CE allows us to obtain F_2 from F_1 by removing a clause. Likewise, for a literal-addition technique LA, we define the relation $\rightarrow_{\mathsf{LA}}$ over clauses as $C_1 \rightarrow_{\mathsf{LA}} C_2$ if and only if the technique LA allows us to obtain C_2 from C_1 by adding a literal. Hence, when we ask whether a certain preprocessing technique is confluent, what we actually want to know is whether its corresponding reduction relation is confluent [18]:

Definition 11. *Let $\rightarrow$ be a relation and $\rightarrow_*$ its reflexive transitive closure. Then, $\rightarrow$ is* confluent *if, for all x, y_1, y_2 with $x \rightarrow_* y_1$ and $x \rightarrow_* y_2$, there exists an element z such that $y_1 \rightarrow_* z$ and $y_2 \rightarrow_* z$.*

In our context, this means that whenever the elimination of certain clauses from a formula F yields a formula F_1, and the elimination of certain other clauses from F yields another formula F_2, then there is still a formula F_z that we can obtain from both F_1 and F_2. Likewise for the addition of literals to a clause. Therefore, we do not need to worry about "missed opportunities" caused by a bad choice of the elimination order. For some techniques in this paper, we can show the stronger *diamond property* which implies confluence [18]:

Definition 12. *A relation $\rightarrow$ has the* diamond property *if, for all x, y_1, y_2 with $x \rightarrow y_1$ and $x \rightarrow y_2$, there exists a z such that $y_1 \rightarrow z$ and $y_2 \rightarrow z$.*

Next, we present the confluence results. We start with blocked-clause elimination, for which confluence is easily shown. Define $F_1 \rightarrow_{\mathsf{BCE}} F_2$ iff the formula F_2 can be obtained from the formula F_1 by removing a clause that is blocked in F_1.

Theorem 12. *Blocked-clause elimination is confluent, i.e., $\rightarrow_{\mathsf{BCE}}$ is confluent.*

Proof. If a clause C is blocked in a formula F, it is also blocked in every subset F' of F, since the L-resolvents of C with clauses in $F' \setminus \{C\}$ are a subset of the L-resolvents with clauses in $F \setminus \{C\}$. Therefore, if all L-resolvents of C with clauses in $F \setminus \{C\}$ are tautologies, so are those with clauses in $F' \setminus \{C\}$. Hence, the relation $\rightarrow_{\mathsf{BCE}}$ has the diamond property and thus it is confluent. □

As in the propositional case, where covered-clause elimination is confluent [3], we can prove the confluence of its first-order variant. Define $F_1 \rightarrow_{\mathsf{CCE}} F_2$ iff the formula F_2 can be obtained from F_1 by removing a clause that is covered in F_1.

Theorem 13. *Covered-clause elimination is confluent, i.e., $\rightarrow_{\mathsf{CCE}}$ is confluent.*

Proof. We show that $\rightarrow_{\mathsf{CCE}}$ has the diamond property. Let F be a formula and let $F \setminus \{C\}$ and $F \setminus \{D\}$ be obtained from F by respectively removing the covered clauses C and D. It suffices to prove that C is covered in $F \setminus \{D\}$ and D is covered in $F \setminus \{C\}$. We show that C is covered in $F \setminus \{D\}$. The other case is symmetric. Since C is covered in F, we can perform a sequence of covered-literal additions to turn C into a clause $C_n = C \lor L_1 \lor \cdots \lor L_n$ that is blocked in F_n, where by F_i we denote the formula obtained from F by replacing C with $C_i = C \lor L_1 \lor \cdots \lor L_i$ $(0 \leq i \leq n)$.

Now, if in $F \setminus \{D\}$, the clause C_n can be obtained from C by performing the same sequence of covered-literal additions, then C_n is also blocked in $F_n \setminus \{D\}$ and thus C is covered in $F \setminus \{D\}$. Assume now to the contrary that there exists a literal L_i that is not covered by C_{i-1} in $F_{i-1} \setminus \{D\}$ and suppose w.l.o.g. that L_i is the first such literal. It follows that there exists a non-tautological L-resolvent of C_{i-1} (with a clause in $F_{i-1} \setminus \{D\}$) that does not contain L_i. But then L_i is not covered by C_{i-1} in F_{i-1}, a contradiction. $\square$

Covered-literal addition is confluent. Let F be a formula and define $C_1 \rightarrow_{\mathsf{CLA}} C_2$ iff C_2 can be obtained from C_1 by adding a literal L that is covered by C_1 in F.

Theorem 14. *Covered-literal addition is confluent, i.e.,* $\rightarrow_{\mathsf{CLA}}$ *is confluent.*

Proof. We show that the relation $\rightarrow_{\mathsf{CLA}}$ has the diamond property. Let F be formula and C a clause. Let furthermore $C_1 = C \lor L_1$ and $C_2 = C \lor L_2$ be obtained from C by respectively adding literals L_1 and L_2 that are both covered by C in F. We have to show that C_1 covers L_2 and, analogously, that C_2 covers L_1. Since C covers L_2, it follows that C contains a literal L such that L_2 is contained in all non-tautological L-resolvents of C. But, as $L \in C_1$, every non-tautological L-resolvent of C_1 must also contain L_2. It follows that C_1 covers L_2. The argument for L_1 being covered by C_2 is symmetric. $\square$

Asymmetric-literal addition is also confluent. Let F be a formula and define $C_1 \rightarrow_{\mathsf{ALA}} C_2$ iff C_2 can be obtained from C_1 by adding a literal L that is an asymmetric literal w.r.t. C_1 in F.

Theorem 15. *Asymmetric-literal addition is confluent, i.e., the relation* $\rightarrow_{\mathsf{ALA}}$ *is confluent.*

Proof. If L_1 is an asymmetric literal w.r.t. a clause C in a formula F, then there exists a clause $D \lor \bar{L} \in F \setminus \{C\}$ and a substitution λ such that $D\lambda \subseteq C$ and $L_1 = \bar{L}\lambda$. Thus, $D\lambda \subseteq C \lor L_2$ for each $C \lor L_2$ that was obtained from C by adding some asymmetric literal L_2, and so L_1 is an asymmetric literal w.r.t. every such clause. Hence, $\rightarrow_{\mathsf{ALA}}$ has the diamond property and so it is confluent. $\square$

For asymmetric-tautology elimination, the non-confluence result from propositional logic [3] implies non-confluence of the first-order generalization. Finally, the following example shows that RS and RAT elimination are not confluent:

Example 16. Let $F = \{\neg Q \vee P,\ \neg R \vee Q,\ \neg P \vee R,\ \neg Q \vee R\}$. Then, $\neg Q \vee R$ is a RAT and RS on the literal R as there is only one R-resolvent, namely the tautology $\neg Q \vee Q$, obtained by resolving with $Q \vee \neg R$. If we remove $\neg Q \vee R$, none of the remaining clauses of F is a RAT or RS. In contrast, suppose we start by removing $\neg Q \vee P$, which is a RAT and RS on P, then all the other clauses can be removed, because they become RAT and RS: The clause $\neg R \vee Q$ becomes both RAT and RS on the literal Q as there is only a tautological resolvent upon Q, namely $\neg R \vee R$. For $\neg P \vee R$, there are no resolvents upon $\neg P$ and so it trivially becomes RAT and RS on $\neg P$. Finally, $\neg Q \vee R$ becomes RAT and RS on both R and $\neg Q$ as there are only tautological resolvents upon these two literals. □

A summary of the confluence results is given in Table 1. Note that for all the confluent techniques, we could show that they also have the diamond property.

Technique	**Confluent**
Blocked-Clause Elimination	yes
Covered-Clause Elimination	yes
Asymmetric-Tautology Elimination	no
Resolution-Asymmetric-Tautology Elimination	no
Resolution-Subsumed-Clause Elimination	no
Covered-Literal Addition	yes
Asymmetric-Literal Addition	yes

Table 1. Confluence Properties of the First-Order Preprocessing Techniques.

10 Conclusion

We introduced the principle of implication modulo resolution for first-order logic and showed that if a clause C is implied modulo resolution by a formula F, then C is redundant with respect to F. Using implication modulo resolution, we lifted several SAT-preprocessing techniques to first-order logic, proved their soundness, and analyzed their confluence properties. We furthermore demonstrated how implication modulo resolution yields short soundness proofs for the existing first-order techniques of predicate elimination and blocked-clause elimination.

For now, we have only considered first-order logic without equality. A variant of implication modulo resolution that guarantees redundancy in first-order logic with equality requires a refined notion of L-resolvents, possibly based on flat resolvents [2] as in the definition of equality-blocked clauses [8]. The focus of this paper is mainly theoretical, laying the groundwork for practical applications of the new first-order techniques. We plan to implement and empirically evaluate the preprocessing techniques proposed in this paper within the next year, since we expect them to improve the performance of first-order theorem provers.

References

1. Hoder, K., Khasidashvili, Z., Korovin, K., Voronkov, A.: Preprocessing techniques for first-order clausification. In: Proc. of the 12th Conference on Formal Methods in Computer-Aided Design (FMCAD 2012), IEEE (2012) 44–51
2. Khasidashvili, Z., Korovin, K.: Predicate elimination for preprocessing in first-order theorem proving. In: Proc. of the 19th Int. Conference on Theory and Applications of Satisfiability Testing (SAT 2016). Volume 9710 of LNCS., Cham, Springer (2016) 361–372
3. Heule, M.J.H., Järvisalo, M., Lonsing, F., Seidl, M., Biere, A.: Clause elimination for SAT and QSAT. Journal of Artificial Intelligence Research **53** (2015) 127–168
4. Heule, M.J.H., Seidl, M., Biere, A.: Solution validation and extraction for QBF preprocessing. J. Autom. Reasoning **58**(1) (2017) 97–125
5. Heule, M.J.H., Järvisalo, M., Biere, A.: Covered clause elimination. In: Short papers for the 17th Int. Conference on Logic for Programming, Artificial intelligence, and Reasoning (LPAR-17-short). Volume 13 of EPiC Series., EasyChair (2010) 41–46
6. Heule, M.J.H., Järvisalo, M., Biere, A.: Clause elimination procedures for CNF formulas. In: Proc. of the 17th Int. Conference on Logic for Programming, Artificial Intelligence, and Reasoning (LPAR-17). Volume 6397 of LNCS., Heidelberg, Springer (2010) 357–371
7. Järvisalo, M., Heule, M.J.H., Biere, A.: Inprocessing rules. In: Proc. of the 6th Int. Joint Conference on Automated Reasoning (IJCAR 2012). Volume 7364 of LNCS., Heidelberg, Springer (2012) 355–370
8. Kiesl, B., Suda, M., Seidl, M., Tompits, H., Biere, A.: Blocked clauses in first-order logic. In: Proceedings of the 21st International Conference on Logic for Programming, Artificial Intelligence and Reasoning (LPAR-21). Volume 46 of EPiC Series in Computing., EasyChair (2017) 31–48
9. Järvisalo, M., Biere, A., Heule, M.J.H.: Blocked clause elimination. In: Proc. of the 16th Int. Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS 2010). Volume 6015 of LNCS., Heidelberg, Springer (2010) 129–144
10. Biere, A.: Splatz, Lingeling, Plingeling, Treengeling, YalSAT Entering the SAT Competition 2016. In: Proc. of SAT Competition 2016 – Solver and Benchmark Descriptions. Volume B-2016-1 of Dep. of Computer Science Series of Publications B., University of Helsinki (2016) 44–45
11. Wetzler, N., Heule, M.J.H., Hunt Jr., W.A.: DRAT-trim: Efficient checking and trimming using expressive clausal proofs. In: Proc. of the 17th Int. Conference on Theory and Applications of Satisfiability Testing (SAT 2014). Volume 8561 of LNCS., Cham, Springer (2014) 422–429
12. Fitting, M.: First-Order Logic and Automated Theorem Proving. 2 edn. Springer (1996)
13. Bachmair, L., Ganzinger, H.: Resolution theorem proving. In Robinson, J.A., Voronkov, A., eds.: Handbook of Automated Reasoning (in 2 volumes). Elsevier and MIT Press (2001) 19–99
14. Kullmann, O.: On a generalization of extended resolution. Discrete Applied Mathematics **96-97** (1999) 149–176
15. Biere, A., Lonsing, F., Seidl, M.: Blocked clause elimination for QBF. In: Proc. of the 23rd Int. Conference on Automated Deduction (CADE 2011). Volume 6803 of LNCS., Heidelberg, Springer (2011) 101–115

16. Lonsing, F., Bacchus, F., Biere, A., Egly, U., Seidl, M.: Enhancing search-based QBF solving by dynamic blocked clause elimination. In: Proc. of the 20th Int. Conference on Logic for Programming, Artificial Intelligence, and Reasoning (LPAR-20). Volume 9450 of LNCS., Heidelberg, Springer (2015) 418–433
17. Eén, N., Biere, A.: Effective preprocessing in SAT through variable and clause elimination. In: Proc. of the 8th Int. Conference on Theory and Applications of Satisfiability Testing (SAT 2005). Volume 3569 of LNCS., Heidelberg, Springer (2005) 61–75
18. Baader, F., Nipkow, T.: Term Rewriting and All That. Cambridge University Press (1998)