



Extending SMT Solving with Non-Ground Clause Learning

Yasmine Briefs^{1,2} and Christoph Weidenbach¹

¹ Max Planck Institute for Informatics, Saarbrücken, Germany

{ybriefs,weidenbach}@mpi-inf.mpg.de

² Graduate School of Computer Science,
Saarland Informatics Campus, Saarbrücken, Germany

Abstract

Quantifier instantiation is currently the main approach to non-ground SMT solving: solvers generate ground instances and solve the resulting ground SMT problems with CDCL(T)-style reasoning. When a conflict is found, conflict analysis learns only a ground clause, even though the conflict comes from instances of non-ground clauses. Yet non-ground reasoning can give exponentially shorter proofs than purely ground reasoning. We propose a calculus that consists of ground instantiations, CDCL(T)-style rules, and non-ground conflict analysis. The solver reasons on ground instances, but the resolution steps of conflict analysis are performed on their original non-ground clauses. This produces learned clauses that are typically more general than the ground conflict. With a suitable strategy, the learned clauses are even non-redundant. We also show how chronological backtracking can be included in SMT solving. Our calculus gives a common setting for CDCL(T)-style SMT solving, a range of instantiation-based procedures, and non-ground clause learning, and we prove that it simulates CDCL, SCL(FOL), SCL(T), and even Resolution.

1 Introduction

Satisfiability Modulo Theories (SMT) has become a central tool in automated reasoning, both in its ground form and in the presence of quantifiers. At the ground level, much of this success is due to CDCL(T) [30, 4], which combines conflict-driven clause learning (CDCL) [26] with dedicated theory reasoning. To handle non-ground problems, this architecture is commonly extended by quantifier instantiation [15, 19, 20, 35, 6]: quantified formulas are instantiated, and the resulting ground problems are solved by CDCL(T). Treating solver components, such as CDCL(T), as black boxes is a powerful design principle in automated reasoning. At the same time, many advances in solver efficiency come from integrating these components more tightly with the surrounding search procedure. This observation also motivates a closer integration of quantifier instantiation with CDCL(T). In current SMT solving, quantified clauses are instantiated and the resulting ground conflicts are analyzed by CDCL(T), yielding learned ground clauses. Our goal is to lift this learning step to the non-ground clauses from which the conflicting instances originate. The motivation is not only conceptual: non-ground learning can yield exponentially shorter refutations than purely ground learning [32].

To formalize this idea, we introduce the calculus ICLF (Instantiation-based Clause Learning Framework). The input to ICLF is a first-order clause set modulo theories. A state consists of

three main components: a set N of possibly non-ground clauses, initially containing the input clauses; a set G of ground instances of clauses in N , initially empty; and a ground partial model Γ , called the *trail*, also initially empty. The rules of ICLF fall into four groups. Instantiation rules create and delete ground instances of clauses in N . Ground rules build a partial model for G in the style of CDCL, by decisions and propagations. Theory rules, such as $\mathcal{T}$ -Propagate and $\mathcal{T}$ -Conflict, connect this ground search to a theory solver. Here ICLF abstracts from the concrete theory, or combination of theories, and from the complications involved in theory combination [29, 39, 33, 22]. Finally, conflict analysis rules derive learned clauses. The key point is that, although the clauses in G are ground, they remain linked to their non-ground origins in N . During conflict analysis, resolution is therefore performed on these original clauses, not only on their ground instances.

For example, assume that the trail contains, among others, the literals $h(a) = g(a)$, $h(a) = a$, and $g(a) \neq a$. These literals are inconsistent in the theory of equality, since they contradict transitivity. In ICLF, this can be detected by applying $\mathcal{T}$ -Conflict with the closure $x_2 \neq x_3 \vee x_2 \neq x_4 \vee x_3 = x_4 \cdot \{x_2 \mapsto h(a), x_3 \mapsto g(a), x_4 \mapsto a\}$, a theory lemma expressing transitivity of equality. We write $C \cdot \sigma$ for a clause C together with a grounding substitution σ . The instantiated clause $C\sigma$ is used for the ground search, while the associated non-ground clause C is retained for conflict analysis and resolution. Now assume that $h(a) = g(a)$ was propagated from the closure $c \neq d \vee h(x_1) = g(x_1) \cdot \{x_1 \mapsto a\}$. An ICLF resolution step unifies the contradictory literals $h(x_1) = g(x_1)$ and $x_2 \neq x_3$, for instance using the most general unifier $\{x_2 \mapsto h(x_1), x_3 \mapsto g(x_1)\}$, and derives the clause $c \neq d \vee h(x_1) \neq x_4 \vee g(x_1) = x_4$. Thus, conflict analysis derives a non-ground clause from a ground conflict.

For first-order clause learning, redundancy is a central issue: a learned clause should ideally not be implied by smaller clauses that are already available. In superposition-based theorem proving [1, 31], redundancy elimination is essential for controlling the search space, but tests such as forward subsumption can be expensive. Simple Clause Learning (SCL) [16, 13], a recent approach that lifts ideas from CDCL to first-order logic by guiding non-ground resolution with a ground partial model, avoids this problem in a different way: its learned clauses are non-redundant by construction. This makes SCL an important starting point for our work. At the same time, SCL is designed around direct reasoning with non-ground clauses; all non-ground clauses have to be considered for propagations and conflicts and conflict analysis leaves little freedom. The goal of ICLF is to keep the useful non-redundancy guarantees of SCL while allowing the ground search to remain close to CDCL(T). Our basic strategy [Definition 2] restricts the rules only enough to ensure that learned clauses are non-redundant with respect to the current set G of ground instances [Theorem 11]. Thus, implementations can focus propagation and conflict search on G and use efficient techniques for ground reasoning, without constantly inspecting the full non-ground clause set N . We also define a stronger, first-order aware strategy [Definition 17], which additionally takes N into account and guarantees non-redundancy with respect to N [Theorem 18]. This strategy is closer in spirit to SCL, but still leaves more freedom in conflict analysis. Finally, ICLF contains multiple backtracking rules. They capture CDCL/CDCL(T)-style backtracking, first-order backtracking in the spirit of SCL, and a way to incorporate chronological backtracking [28, 27] into SMT solving.

This flexibility makes ICLF a common setting for several forms of model-based reasoning. In the propositional case, it simulates CDCL. For first-order logic, it simulates SCL and Resolution. Moreover, it simulates clause learning in SCL(T) [11], an adaptation of SCL to the fragment of first-order logic modulo theories without uninterpreted functions and constants. It also captures CDCL(T)-style reasoning on ground clauses modulo theories. Finally, ICLF can be seen as a formal setting for studying a range of instantiation-based SMT procedures: although

quantifier instantiation is a standard practical approach, it is usually presented through concrete algorithms and solver architectures rather than as one fixed calculus. In this sense, ICLF is intended as a foundational framework: it identifies which rule restrictions are needed for soundness, non-redundant learning, and termination under suitable restrictions, and which choices can instead be left to strategies and implementations.

The paper is organized as follows. In Section 3, we present the calculus ICLF and establish, among other properties, soundness [Theorem 6], termination under suitable restrictions [Theorem 14], and completeness under suitable restrictions [Corollary 16]. In Section 4, we show that ICLF simulates the other calculi mentioned above. Further examples, lemmas, and explanations, as well as all omitted proofs, can be found in the extended version of the paper [9].

2 Preliminaries

We consider the following standard notions of many-sorted first-order logic without equality [11]. Let $\Sigma = (\mathcal{S}, \Omega, \Pi)$ be a many-sorted signature where $\mathcal{S}$ is a finite, non-empty set of *sort symbols*, Ω is a non-empty set of *function symbols* over $\mathcal{S}$ and Π is a finite, non-empty set of *predicate symbols* over $\mathcal{S}$. We assume that Ω contains at least one constant of each sort. We additionally assume a set of variables $\mathcal{X}$ that contains infinitely many variables of each sort. First-order terms, atoms, literals and substitutions are defined in the usual way. We assume that substitutions are well-sorted and that they have finite *domain* $\text{dom}(\sigma) = \{x \mid x\sigma \neq x\}$. We further define the *codomain* of a substitution as $\text{codom}(\sigma) = \{x\sigma \mid x \in \text{dom}(\sigma)\}$. A *clause* is a set of literals, which means that duplicate literals are implicitly deleted. If we write a clause as $L_1 \vee \dots \vee L_n$, where the L_i are literals, we mean $\{L_1\} \cup \dots \cup \{L_n\}$. Likewise, $C \vee L$, where C is a clause and L is a literal, denotes $C \cup \{L\}$, and $C \vee D$, where C and D are clauses, denotes $C \cup D$. $\perp$ denotes the *empty clause*. We say that a term is *ground* if it does not contain any variables. We extend this definition to sets of terms, atoms, literals, clauses and clause sets in the obvious way. A substitution σ is called *grounding* for a term t if $t\sigma$ is a ground term. We also extend this definition to atoms, literals and clauses in the obvious way. By $\text{comp}(L)$, we denote the *complement* of a literal L . We further define a function atom that maps literals to their corresponding atoms. We extend this function to map clauses to the set of atoms of the literals occurring in the clause and clause sets to the set of atoms of the clauses in the clause set. A *closure* is a pair $C \cdot \sigma$ where C is a clause or $\top$ and σ is a grounding substitution for C . We assume that variables that do not occur in C are implicitly dropped from σ , i.e., that $\text{dom}(\sigma)$ only contains variables that occur in C . Moreover, we assume that $\text{codom}(\sigma)$ is ground. If $C \in \{\top, \perp\}$, we may write $\top$ to refer to the closure $\top \cdot \{\}$ and $\perp$ to refer to the closure $\perp \cdot \{\}$. Given a clause C , a closure $C \cdot \sigma$ is called an *instance* of C . For a clause C , $\text{gnd}(C)$ denotes the set $\text{gnd}(C) = \{C\sigma \mid \sigma \text{ is grounding for } C\}$. For a clause set N , $\text{gnd}(N) = \bigcup_{C \in N} \text{gnd}(C)$. The function mgu denotes the *most general unifier* of two terms, atoms or literals. We assume that the mgu does not introduce any fresh variables and is idempotent. Given a ground literal L and a closure $C \cdot \sigma$, let $C = C' \vee L_1 \vee \dots \vee L_n$ such that $L_i\sigma = L$ for $i \in \{1, \dots, n\}$ and L does not occur in $C'\sigma$. Further, let $\mu = \text{mgu}(L_1, \dots, L_n)$. Then, the result of *exhaustively factorizing* L in $C \cdot \sigma$ is $(C' \vee L_1)\mu \cdot \sigma$.

The semantics of many-sorted first-order logic without equality are given by the notion of an algebra. A Σ -*algebra* $\mathcal{A}$ is a mapping that assigns (i) a non-empty carrier set $S^{\mathcal{A}}$ to every sort $S \in \mathcal{S}$, so that $S_1^{\mathcal{A}} \cap S_2^{\mathcal{A}} = \emptyset$ for any distinct sorts $S_1, S_2 \in \mathcal{S}$, (ii) a total function $f^{\mathcal{A}} : S_1^{\mathcal{A}} \times \dots \times S_n^{\mathcal{A}} \rightarrow S^{\mathcal{A}}$ to every function symbol $f : S_1 \times \dots \times S_n \rightarrow S \in \Omega$, (iii) a relation $P^{\mathcal{A}} \subseteq S_1^{\mathcal{A}} \times \dots \times S_n^{\mathcal{A}}$ to every predicate symbol $P \in \Pi$ of arity n . We assume $\mathcal{T}$ to be a *theory*, i.e., a non-empty set of Σ -algebras. $\mathcal{T}$ can also be the combination of multiple theories. For

instance, as we consider first-order logic without equality, if needed, equality reasoning has to be part of the theory. The semantic entailment relation $\models$ is defined in the usual way. We write $\models_{\mathcal{T}}$ to denote the semantic entailment relation that considers only the Σ -algebras $\mathcal{A} \in \mathcal{T}$.

A *trail* is a sequence of annotated, ground literals. Given a trail Γ , a ground literal L is called *propositionally true/false* in Γ if $L/\text{comp}(L)$ occurs in Γ . Otherwise, L is called *propositionally undefined* in Γ . The trail in ICLF is always *consistent*, i.e., no ground literal is propositionally true and propositionally false at the same time. A ground clause C is called *propositionally true* in Γ if there is a literal $L \in C$ that is propositionally true in Γ . It is called *propositionally false* in Γ if all literals $L \in C$ are propositionally false in Γ . Otherwise, it is called *propositionally undefined* in Γ . Trails are interpreted as the conjunction of their literals, clauses are interpreted as the disjunction of their literals and clause sets are interpreted as the conjunction of their clauses. A clause set N is called *satisfiable* if there exists a Σ -algebra $\mathcal{A}$ such that $\mathcal{A} \models N$. It is called *$\mathcal{T}$ -satisfiable* if there exists a Σ -algebra $\mathcal{A} \in \mathcal{T}$ such that $\mathcal{A} \models N$. Similarly, the $\mathcal{T}$ -satisfiability of a trail Γ is defined. Given a ground set of clauses N , a trail Γ is called a *model* of N if all clauses in N are propositionally true in Γ . Two clause sets N_1 and N_2 are called *equi-satisfiable* if N_1 is satisfiable if and only if N_2 is satisfiable. They are called *equi-satisfiable in $\mathcal{T}$* if N_1 is $\mathcal{T}$ -satisfiable if and only if N_2 is $\mathcal{T}$ -satisfiable.

Definition 1 (Clause Redundancy). *Let $\prec$ be a well-founded, total, strict ordering on ground literals. We lift this ordering to clauses by its multiset extension. We further define $\preceq$ to be the reflexive closure of $\prec$ and $N \preceq^C = \{D \in N \mid D \preceq C\}$ for a ground clause C and a ground clause set N .*

A ground clause C is called redundant with respect to a ground clause set N and $\prec$ if $N \preceq^C \models C$. A clause C is called redundant with respect to a clause set N and $\prec$ if for all $C' \in \text{gnd}(C)$, it holds that C' is redundant with respect to $\text{gnd}(N)$.

3 The Calculus

We define ICLF as an abstract rewrite system. Its inference rules operate on a state that is a six-tuple $(\Theta; \Gamma; N; G; k; D \cdot \tau)$ where Θ is a set of ground atoms, Γ is the trail, N is a clause set, G is a set of closures, k is a natural number called the *decision level*, and $D \cdot \tau$ is the *conflict closure*. We define $\text{clauses}(G) = \{C\sigma \mid C \cdot \sigma \in G\}$. The purpose of Θ is to restrict the set of ground atoms that are currently available in the search for a model of $\text{clauses}(G)$. Θ always contains at least all atoms that occur on the trail or in $\text{clauses}(G)$ [Lemma 5]. The literals in Γ are either annotated with a natural number, in which case they are called *decisions*, or with a closure, in which case they are called *propagations*, or with $\mathcal{T}$, in which case they are called *theory propagations*. The decisions in Γ are annotated with $1, \dots, k$ in this order [Lemma 5]. The *decision level* of a decision is its annotation, and the decision level of any other literal in Γ is the annotation of the decision closest to its left. If there is no such decision, the decision level of the literal is 0. The decision level of Γ is the maximal decision level of a literal in it, or 0 if $\Gamma = \varepsilon$. For a simpler presentation, N contains both the initial and the learned clauses and G contains both the instantiated and the learned closures. If $D \cdot \tau = \top$, the algorithm is in the instantiation and model building phase. If $D \cdot \tau = \perp$, $\mathcal{T}$ -unsatisfiability has been derived. Otherwise, the algorithm is in the conflict analysis phase. The start state for a set of clauses N' is $(\emptyset; \varepsilon; N'; \emptyset; 0; \top)$. We assume that the initial set of clauses N' does not contain the empty clause. Moreover, we require that the $\mathcal{T}$ -satisfiability of conjunctions of ground literals is decidable. Instead of using an abstraction function, we write the ground theory literals on the trail directly. An implementation can still use an abstraction function for the ground reasoning.

We first present the four instantiation rules that are used to manage the instances.

Instantiate $(\Theta; \Gamma; N; G; k; \top) \Rightarrow_{\text{ICLF}} (\Theta \cup \text{atom}(C\sigma); \Gamma; N; G \cup \{C \cdot \sigma\}; k; \top)$
 given that $C \in N$, σ is grounding for C , $C\sigma \notin \text{clauses}(G)$ and either $C\sigma$ is not propositionally false in Γ or $\Gamma = \Gamma_1 K$ with $K \in \{L^{D \cdot \tau}, L^{\mathcal{T}}\}$ and $C\sigma$ is not propositionally false for Γ_1 .

Typically, a run starts by applying Instantiate with the empty substitution to all clauses in N that are already ground. To avoid learning redundant clauses, it is crucial that no instance in $\text{clauses}(G)$ is false for a proper trail prefix or becomes false by a decision [Definition 2]. The condition on $C\sigma$ being propositionally false ensures that newly added instances preserve this property. Allowing propositionally false instances to be instantiated at all is necessary for following a first-order aware strategy [Definition 17]. To add an arbitrary propositionally false instance, Restart can first be used to return to a trail prefix at which Instantiate is applicable.

ClauseDel $(\Theta; \Gamma; N \uplus \{C\}; G; k; \top) \Rightarrow_{\text{ICLF}} (\Theta; \Gamma; N; G; k; \top)$
 given that $N \uplus \{C\}$ and N are equi-satisfiable in $\mathcal{T}$ and there is no grounding σ with $C \cdot \sigma \in G$.

Even though it is undecidable in general whether the rule ClauseDel is applicable, there are some sufficient conditions for its applicability, e.g., that C is subsumed by another clause in N .

InstanceDel $(\Theta; \Gamma; N; G \uplus \{C \cdot \sigma\}; k; \top) \Rightarrow_{\text{ICLF}} (\Theta; \Gamma; N; G; k; \top)$
 given that there is no $L^{C \cdot \sigma}$ in Γ .

Although it can make sense to only apply InstanceDel if $G \uplus \{C \cdot \sigma\}$ and G are equi-satisfiable in $\mathcal{T}$, we would like to allow restarts with fresh instances.

AtomDel $(\Theta \uplus \{A\}; \Gamma; N; G; k; \top) \Rightarrow_{\text{ICLF}} (\Theta; \Gamma; N; G; k; \top)$
 given that $A \notin \text{atom}(\Gamma) \cup \text{atom}(\text{clauses}(G))$.

The following four rules form the ground rules.

Decide $(\Theta; \Gamma; N; G; k; \top) \Rightarrow_{\text{ICLF}} (\Theta; \Gamma L^{k+1}; N; G; k+1; \top)$
 given that $\text{atom}(L) \in \Theta$ and L is propositionally undefined in Γ .

Propagate $(\Theta; \Gamma; N; G; k; \top) \Rightarrow_{\text{ICLF}} (\Theta; \Gamma L^{C \cdot \sigma}; N; G; k; \top)$
 given that $C \cdot \sigma \in G$, $C\sigma = C' \vee L$, C' is propositionally false in Γ and L is propositionally undefined in Γ .

Conflict $(\Theta; \Gamma; N; G; k; \top) \Rightarrow_{\text{ICLF}} (\Theta; \Gamma; N; G; k; D \cdot \tau)$
 given that $D \cdot \tau \in G$ and $D\tau$ is propositionally false in Γ .

Restart $(\Theta; \Gamma; N; G; k; \top) \Rightarrow_{\text{ICLF}} (\Theta; \Gamma_1; N; G; j; \top)$
 given that $\Gamma = \Gamma_1 \Gamma_2$ and j is the decision level of Γ_1 .

Applying the rule Restart with $\Gamma_1 = \varepsilon$ corresponds to a restart in CDCL(T), i.e., a restart on the ground level. We allow restarting with arbitrary trail prefixes, enabling the subsequent application of Instantiate with clauses that would have been propositionally false in Γ .

Next, we present the five theory rules.

$\mathcal{T}$ -Propagate $(\Theta; \Gamma; N; G; k; \top) \Rightarrow_{\text{ICLF}} (\Theta; \Gamma L^{\mathcal{T}}; N; G; k; \top)$
 given that $\text{atom}(L) \in \Theta$, $\Gamma \models_{\mathcal{T}} L$ and L is propositionally undefined in Γ .

Theory propagations are annotated with $\mathcal{T}$ and can be explained later when they become relevant during conflict analysis. The condition $\Gamma \models_{\mathcal{T}} L$ makes sure that there exists a tautology in the theory from which L can be propagated.

$\mathcal{T}$ -Learn $(\Theta; \Gamma; N; G; k; \top) \Rightarrow_{\text{ICLF}} (\Theta; \Gamma; N \cup \{C\}; G; k; \top)$
 given that $C \notin N$ and $\models_{\mathcal{T}} C$.

Note that $\mathcal{T}$ -Learn adds a non-ground theory lemma in general. This can then be instantiated by the rule Instantiate. If the added theory lemma is already ground, it can be instantiated with the empty substitution $\{\}$.

$\mathcal{T}$ -Atom $(\Theta; \Gamma; N; G; k; \top) \Rightarrow_{\text{ICLF}} (\Theta \cup \{A\}; \Gamma; N; G; k; \top)$
 given that $A \notin \Theta$ and A is a ground atom.

$\mathcal{T}$ -Conflict $(\Theta; \Gamma; N; G; k; \top) \Rightarrow_{\text{ICLF}} (\Theta; \Gamma; N; G; k; D \cdot \tau)$
 given that $\models_{\mathcal{T}} D$, τ is grounding for D and $D\tau$ is propositionally false in Γ .

Just like in $\mathcal{T}$ -Learn, the conflict clause produced by $\mathcal{T}$ -Conflict can be non-ground.

Explain $(\Theta; \Gamma_1 L^{\mathcal{T}} \Gamma_2; N; G; k; D \cdot \tau) \Rightarrow_{\text{ICLF}} (\Theta; \Gamma_1 L^{C \cdot \sigma} \Gamma_2; N; G; k; D \cdot \tau)$
 given that $\models_{\mathcal{T}} C$, σ is grounding for C , $C\sigma = C' \vee L$ and C' is propositionally false in Γ_1 .

$\top$ is a short form for the conflict closure $\top \cdot \{\}$, i.e., the rule Explain can be applied both in the model building and in the conflict analysis phase. Just like in $\mathcal{T}$ -Learn and $\mathcal{T}$ -Conflict, the reason can be non-ground. For each theory propagation on the trail, there exists a clause to apply Explain with, for example $\neg \Gamma_1 \vee L \cdot \{\}$, which is always valid in the theory by Lemma 5.

Finally, the following five rules are the conflict analysis rules.

Resolve $(\Theta; \Gamma_1 L^{C \cdot \sigma} \Gamma_2; N; G; k; (D \vee L') \cdot \tau) \Rightarrow_{\text{ICLF}} (\Theta; \Gamma_1 L^{C \cdot \sigma} \Gamma_2; N; G; k; \text{resolve}(C \cdot \sigma, D \cdot \tau, L', L))$
 given that $L'\tau = \text{comp}(L)$.

The function resolve takes as arguments two variable disjoint closures $C \cdot \sigma$ and $D \cdot \tau$, a literal L' that does not occur in D , may share variables with D , but not with C , and a ground literal L such that $L'\tau = \text{comp}(L)$. Let $C = C' \vee L_1 \vee \dots \vee L_n$ such that $L_i\sigma = L$ for all $i \in \{1, \dots, n\}$ and L does not occur in $C'\sigma$. If $\mu = \text{mgu}(\text{comp}(L'), L_1, \dots, L_n)$, then $\text{resolve}(C \cdot \sigma, D \cdot \tau, L', L) = (C' \vee D)\mu \cdot \sigma\tau$. In other words, the function resolve exhaustively factorizes L in C and computes a resolution inference between the two clauses.

The rule Resolve allows resolving any literal that is on the trail. This generality makes standard simplifications such as unit reduction possible. Additionally, it allows modeling conflict analysis procedures such as in standard CDCL with UIP learning [26] or in SCL [13] that work through the trail backwards, performing either Skip or Resolve for each encountered literal, as well as procedures like chronological backtracking [28, 27] that only resolve with a subsequence of the relevant literals on the trail.

Factorize $(\Theta; \Gamma; N; G; k; (D \vee L \vee L') \cdot \tau) \Rightarrow_{\text{ICLF}} (\Theta; \Gamma; N; G; k; (D \vee L)\eta \cdot \tau)$
 given that $L\tau = L'\tau$ and $\eta = \text{mgu}(L, L')$.

The ground conflict clause does not change by applying Factorize. If the literal that Resolve is applied to can be factorized, then it does not disappear from the ground conflict clause after resolving. While the resulting ground conflict clause is the same regardless of whether a literal is exhaustively factorized before it is resolved or whether Resolve is applied multiple times, this is, in general, not true for the non-ground conflict clause [13, Example 3].

BacktrackClassic $(\Theta; \Gamma_1 K^j \Gamma_2; N; G; k; D \cdot \tau) \Rightarrow_{\text{ICLF}} (\Theta; \Gamma_1 L^{D \cdot \tau}; N \cup \{D\}; G \cup \{D \cdot \tau\}; j - 1; \top)$

given that $D\tau = D' \vee L$, D' is propositionally false in Γ_1 and L is propositionally undefined in Γ_1 .

BacktrackFOL $(\Theta; \Gamma_1 \Gamma_2; N; G; k; D \cdot \tau) \Rightarrow_{\text{ICLF}} (\Theta; \Gamma_1; N \cup \{D\}; G \cup \{D \cdot \tau\}; j; \top)$

given that there is no grounding σ such that $D\sigma$ is propositionally false in Γ_1 , j is the decision level of Γ_1 and Γ_2 contains a decision literal.

BacktrackCB $(\Theta; \Gamma; N; G; k; D \cdot \tau) \Rightarrow_{\text{ICLF}} (\Theta; \Gamma' L^{D \cdot \tau}; N \cup \{D\}; G \cup \{D \cdot \tau\}; j; \top)$

given that $\Gamma' = \text{cb}(\Gamma, D\tau, j)$, $D\tau = D' \vee L$, D' is propositionally false in Γ' and L is propositionally undefined in Γ' .

Here, CB stands for chronological backtracking [28, 27], an alternative backtracking scheme for CDCL that we adapt to ICLF in this paper. We now define the function cb . To this end, given a trail Γ , we first define the *conflict graph* as the directed acyclic graph whose nodes are the literals on the trail Γ . Decision literals L^k and theory propagations L^τ do not have any predecessors, and the predecessors of a propagation $L^{C' \cdot \sigma}$ are the literals $\text{comp}(L')$ for all $L' \in C' \cdot \sigma$ with $L' \neq L$. If L is a propagation (or an explained theory propagation), all of these predecessors are indeed literals on the trail. For each literal L on the trail Γ , we define $\text{reach}(L)$ as the set of literals that can be reached from L in the conflict graph, including L itself. By $\text{reach}^{-1}(L)$, we denote the set of literals that can reach L in the conflict graph.

The arguments of the function cb are a trail Γ , a ground clause C and a destination decision level j that is smaller than the decision level of Γ . Let $\Gamma = \Gamma_1 L'^{j+1} \Gamma_2$. cb is only applicable if the following three conditions are met. First, C should be propositionally false in Γ , i.e., for all $L \in C$, $\text{comp}(L)$ should be in Γ . Second, $|\{\text{comp}(L) \mid L \in C\} \cap \text{reach}(L')| = 1$, i.e., exactly one literal in C should depend on the $j+1$ -st decision. Third, in $\bigcup_{L \in C} \text{reach}^{-1}(\text{comp}(L)) \cap \Gamma_2$, there should be no literal annotated as a decision or theory propagation, i.e., no conflict literal should depend on a decision or theory propagation after L' on the trail. Then, let $C = C' \vee L''$ such that $\{\text{comp}(L) \mid L \in C'\} \cap \text{reach}(L') = \emptyset$, i.e., L'' is the only literal in C that depends on L' . Further, let Γ_3 be the subsequence of Γ_2 that contains only the literals $\bigcup_{L \in C'} \text{reach}^{-1}(\text{comp}(L)) \cap \Gamma_2$. It is crucial to note that Γ_3 only contains propagations (and explained theory propagations), and that none of the literals in Γ_3 was propagated from L' . Then $\text{cb}(\Gamma, C, j) = \Gamma_1 \Gamma_3$. By the aforementioned properties of Γ_3 , all propagations in Γ_3 are still justified in $\Gamma_1 \Gamma_3$ and C' stays propositionally false in $\Gamma_1 \Gamma_3$. Figure 1 illustrates this definition of the function cb .

Definition 2. A strategy for ICLF is called *reasonable* if the following are true:

1. The rule *Conflict* is preferred over the rules *Decide*, *Propagate*, $\mathcal{T}$ -*Propagate* and $\mathcal{T}$ -*Conflict*.
2. No application of the rule *Decide* enables an immediate application of the rule *Conflict*.
3. If a conflict state is reached by the rule *Conflict*, then *Resolve* and *Factorize* are applied in such a way that the complement of the rightmost literal on the trail does not occur in the ground conflict clause anymore before applying any of the rules *BacktrackClassic*, *BacktrackFOL* or *BacktrackCB*.

These requirements do not cause any stuck states by Theorem 15. Although this does not affect correctness or termination, to avoid that previously resolved literals are reintroduced into the conflict clause, *Resolve* should be applied from right to left on the trail.

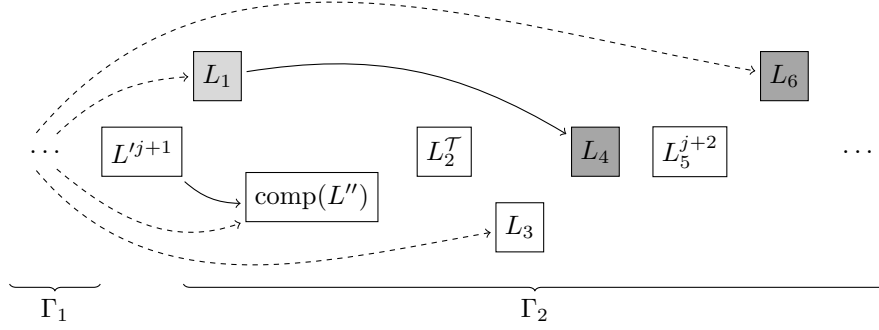


Figure 1: An example to illustrate the definition of the function cb . C' contains the complements of the literals L_4 and L_6 . Then $\Gamma_3 = L_1 L_4 L_6$.

The goal of restricting the strategy is to learn only clauses that are non-redundant with respect to the current set of instantiations G and a trail induced ordering [Definition 9]. Most SAT solvers implement CDCL with 1UIP learning [26]. In CDCL, the 1UIP clause is non-redundant because exhaustive propagation maintains the necessary invariants [41]. In ICLF, however, these invariants can be broken by theory rules and instantiations. For example, a newly instantiated clause may propagate with respect to an earlier trail prefix, making it possible for the 1UIP clause to be redundant. Moreover, for theory propagations and for literals propagated from them, the exact implying literals are not known before a theory explanation is computed, which may be expensive. For these reasons, 1UIP learning is not straightforward in our setting. Therefore, the restrictions of a reasonable strategy capture only what is needed for non-redundant learning and otherwise leave the conflict analysis rules general.

The following example shows ICLF in action modulo the theory of equality for the sake of simple theory reasoning. Of course, ICLF works modulo any suitable theory combination in the same way as CDCL(T). This run closely resembles quantifier instantiation combined with CDCL(T) and shows how ICLF can model this standard approach.

Example 3. *We present a reasonable ICLF run on the clause set*

$$\begin{aligned}
 N_0 = \{ & (N1) \ f(x_0) = g(x_0) \vee f(x_1) = h(x_1), & (N2) \ g(b) = c, \\
 & (N3) \ f(a) \neq g(a) \vee c = b, & (N4) \ f(b) \neq h(b) \vee f(c) \neq h(c), \\
 & (N5) \ f(a) \neq g(a) \vee g(x_2) \neq x_2 \}.
 \end{aligned}$$

$$\begin{aligned}
 & (\emptyset; \varepsilon; N_0; \emptyset; 0; \top) \\
 \Rightarrow_{ICLF}^{Instantiate \times 5} & (\Theta_0; \varepsilon; N_0; G_0; 0; \top)
 \end{aligned}$$

Here, $\Theta_0 = \{f(a) = g(a), f(b) = h(b), g(b) = c, c = b, f(c) = h(c), g(b) = b\}$ and

$$\begin{aligned}
G_0 = & \{(G1) f(x_0) = g(x_0) \vee f(x_1) = h(x_1) \cdot \{x_0 \mapsto a, x_1 \mapsto b\}, \\
& (G2) g(b) = c \cdot \{\}, \\
& (G3) f(a) \neq g(a) \vee c = b \cdot \{\}, \\
& (G4) f(b) \neq h(b) \vee f(c) \neq h(c) \cdot \{\}, \\
& (G5) f(a) \neq g(a) \vee g(x_2) \neq x_2 \cdot \{x_2 \mapsto b\}\}. \\
\Rightarrow_{ICLF}^{Propagate} & (\Theta_0; \Gamma_1 := (g(b) = c)^{(G2)}; N_0; G_0; 0; \top) \\
\Rightarrow_{ICLF}^{Decide} & (\Theta_0; \Gamma_2 := \Gamma_1(f(a) = g(a))^1; N_0; G_0; 1; \top) \\
\Rightarrow_{ICLF}^{Propagate} & (\Theta_0; \Gamma_3 := \Gamma_2(c = b)^{(G3)}; N_0; G_0; 1; \top) \\
\Rightarrow_{ICLF}^{Propagate} & (\Theta_0; \Gamma_4 := \Gamma_3(g(b) \neq b)^{(G5)}; N_0; G_0; 1; \top) \\
\Rightarrow_{ICLF}^{\mathcal{T}\text{-Conflict}} & (\Theta_0; \Gamma_4; N_0; G_0; 1; g(b) \neq c \vee c \neq b \vee g(b) = b \cdot \{\}) \\
\Rightarrow_{ICLF}^{Resolve(G5)} & (\Theta_0; \Gamma_4; N_0; G_0; 1; f(a) \neq g(a) \vee g(b) \neq c \vee c \neq b \cdot \{\}) \\
\Rightarrow_{ICLF}^{Resolve(G3)} & (\Theta_0; \Gamma_4; N_0; G_0; 1; f(a) \neq g(a) \vee g(b) \neq c \cdot \{\}) \\
\Rightarrow_{ICLF}^{BacktrackClassic} & (\Theta_0; \Gamma_5 := (g(b) = c)^{(G2)}(f(a) \neq g(a))^{(G6)}; N_1; G_1; 0; \top)
\end{aligned}$$

Here, $N_1 = N_0 \cup \{(N6) f(a) \neq g(a) \vee g(b) \neq c\}$ and $G_1 = G_0 \cup \{(G6) f(a) \neq g(a) \vee g(b) \neq c \cdot \{\}\}$.

$$\begin{aligned}
\Rightarrow_{ICLF}^{Propagate} & (\Theta_0; \Gamma_6 := \Gamma_5(f(b) = h(b))^{(G1)}; N_1; G_1; 0; \top) \\
\Rightarrow_{ICLF}^{Propagate} & (\Theta_0; \Gamma_7 := \Gamma_6(f(c) \neq h(c))^{(G4)}; N_1; G_1; 0; \top)
\end{aligned}$$

Γ_7 is a $\mathcal{T}$ -satisfiable model of G_1 . A possible next step is to apply *Instantiate* with the clause $(N1) \cdot \{x_0 \mapsto a, x_1 \mapsto c\}$. This immediately enables the application of *Conflict*, and after several applications of *Resolve*, $\perp$ will be derived, showing the unsatisfiability of the clause set N_0 .

The following example shows how ICLF can learn non-ground clauses. Again, we work modulo the theory of equality to keep the theory reasoning simple.

Example 4. We present a reasonable ICLF run on the clause set

$$N_0 = \{(N1) c = d, \quad (N2) c \neq d \vee h(x_0) = x_0, \quad (N3) c \neq d \vee h(x_1) = g(x_1)\}.$$

$$\begin{aligned}
& (\emptyset; \varepsilon; N_0; \emptyset; 0; \top) \\
\Rightarrow_{ICLF}^{Instantiate \times 3} & (\Theta_0; \varepsilon; N_0; G_0; 0; \top)
\end{aligned}$$

Here, $\Theta_0 = \{c = d, h(a) = a, h(a) = g(a)\}$ and $G_0 = \{(G1) c = d, (G2) c \neq d \vee h(x_0) = x_0 \cdot \{x_0 \mapsto a\}, (G3) c \neq d \vee h(x_1) = g(x_1) \cdot \{x_1 \mapsto a\}\}$.

$$\begin{aligned}
\Rightarrow_{ICLF}^{Propagate} & (\Theta_0; \Gamma_1 := (c = d)^{(G1)}; N_0; G_0; 0; \top) \\
\Rightarrow_{ICLF}^{Propagate} & (\Theta_0; \Gamma_2 := \Gamma_1(h(a) = a)^{(G2)}; N_0; G_0; 0; \top) \\
\Rightarrow_{ICLF}^{Propagate} & (\Theta_0; \Gamma_3 := \Gamma_2(h(a) = g(a))^{(G3)}; N_0; G_0; 0; \top) \\
\Rightarrow_{ICLF}^{\mathcal{T}\text{-Atom}} & (\Theta_1 := \Theta_0 \cup \{g(a) = a\}; \Gamma_3; N_0; G_0; 0; \top) \\
\Rightarrow_{ICLF}^{Decide} & (\Theta_1; \Gamma_4 := \Gamma_3(g(a) \neq a)^1; N_0; G_0; 1; \top) \\
\Rightarrow_{ICLF}^{\mathcal{T}\text{-Conflict}} & (\Theta_1; \Gamma_4; N_0; G_0; 1; x_2 \neq x_3 \vee x_2 \neq x_4 \vee x_3 = x_4)
\end{aligned}$$

$$\begin{aligned}
& \cdot \{x_2 \mapsto h(a), x_3 \mapsto g(a), x_4 \mapsto a\}) \\
\Rightarrow_{\text{ICLF}}^{\text{Resolve}(G3)} & (\Theta_1; \Gamma_4; N_0; G_0; 1; c \neq d \vee h(x_1) \neq x_4 \vee g(x_1) = x_4 \cdot \{x_1 \mapsto a, x_4 \mapsto a\}) \\
\Rightarrow_{\text{ICLF}}^{\text{Resolve}(G2)} & (\Theta_1; \Gamma_4; N_0; G_0; 1; c \neq d \vee g(x_4) = x_4 \cdot \{x_4 \mapsto a\}) \\
\Rightarrow_{\text{ICLF}}^{\text{Resolve}(G1)} & (\Theta_1; \Gamma_4; N_0; G_0; 1; g(x_4) = x_4 \cdot \{x_4 \mapsto a\}) \\
\Rightarrow_{\text{ICLF}}^{\text{BacktrackClassic}} & (\Theta_1; \Gamma_3(g(a) = a)^{(G4)}; N_1; G_1; 0; \top)
\end{aligned}$$

Here, $N_1 = N_0 \cup \{(N4) \ g(x_4) = x_4\}$ and $G_1 = G_0 \cup \{(G4) \ g(x_4) = x_4 \cdot \{x_4 \mapsto a\}\}$. Of course, after the third propagation, Γ_3 is already a $\mathcal{T}$ -satisfiable model of G_0 . However, consider a larger problem that contains the three clauses $N1$, $N2$, and $N3$ among other clauses. In such a problem, another clause might have motivated the decision $g(a) \neq a$. A $\text{CDCL}(T)$ -style approach with quantifier instantiation and ground 1UIP conflict analysis would learn the ground theory conflict clause $h(a) \neq g(a) \vee h(a) \neq a \vee g(a) = a$ from this conflict. By contrast, in this particular run, the generality of *Resolve* allows *ICLF* to resolve further with the three propagations and learn the more general clause $g(x_4) = x_4$. For these further resolution steps to yield the non-ground clause $g(x_4) = x_4$, the non-ground theory lemma $x_2 \neq x_3 \vee x_2 \neq x_4 \vee x_3 = x_4$ is essential: starting from its ground instance instead, the same resolution steps would only yield the ground clause $g(a) = a$. The learned non-ground clause can be useful later in the run: if $g(b)$ becomes relevant, it can be instantiated as $g(x_4) = x_4 \cdot \{x_4 \mapsto b\}$, whereas the ground clause $g(a) = a$ carries no information about $g(b)$.

Next, we prove the correctness of *ICLF* and further properties, including non-redundant learning as well as termination and completeness under suitable restrictions. We start with invariants that are maintained by all *ICLF* rules. The proof is by induction on the length of the derivation, with a case distinction on the last applied rule.

Lemma 5 (Invariants). *Let $(\Theta; \Gamma; N; G; k; D \cdot \tau)$ be a state reached from a start state $(\emptyset; \varepsilon; N'; \emptyset; 0; \top)$ by applying the rules of *ICLF*. Then the following hold:*

1. For all $C \cdot \sigma \in G$, it holds that $C \in N$.
2. $\text{atom}(\Gamma) \cup \text{atom}(\text{clauses}(G)) \subseteq \Theta$ and for all $A \in \Theta$, A is a ground atom.
3. If $\Gamma = \Gamma_1 K \Gamma_2$ then K is propositionally undefined in Γ_1 .
4. If $\Gamma = \Gamma_1 L^{C \cdot \sigma} \Gamma_2$ then $C \sigma$ is ground, $C \sigma = C' \vee L$, C' is propositionally false in Γ_1 and either $C \cdot \sigma \in G$ or $\models_{\mathcal{T}} C$.
5. If $\Gamma = \Gamma_1 L^{\top} \Gamma_2$ then $\Gamma_1 \models_{\mathcal{T}} L$.
6. Let $k_1, \dots, k_l$ be the annotations of decisions in Γ . If there are no decisions, then $k = 0$; otherwise $1 = k_1$, $k = k_l$ and for all $i \in \{2, \dots, l\}$, $k_i = k_{i-1} + 1$.
7. N' and N are equi-satisfiable in $\mathcal{T}$.
8. If $D \neq \top$ then $D \tau$ is ground, $\text{atom}(D \tau) \subseteq \Theta$ and $D \tau$ is propositionally false in Γ .
9. $\text{clauses}(G) \models_{\mathcal{T}} D \tau$.
10. $N \models_{\mathcal{T}} D$.

The soundness of *ICLF* follows immediately from Properties 7 and 10.

Theorem 6 (Soundness). *If $(\emptyset; \varepsilon; N'; \emptyset; 0; \top) \Rightarrow_{ICLF}^* (\Theta; \Gamma; N; G; k; \perp)$, then $N' \models_{\mathcal{T}} \perp$.*

The two previous statements hold regardless of the strategy used to apply the rules of ICLF. The following properties require the rules to be applied according to a reasonable strategy, as defined in Definition 2. We first show two invariants maintained by a reasonable strategy. The first invariant, Lemma 7, states that there is no propositional conflict for a proper trail prefix, and if the last literal on the trail is a decision, then there is no propositional conflict. This invariant is used to prove non-redundant learning with respect to G in Theorem 11; it implies that, after conflict analysis, the learned clause is the only false clause for a suitable trail prefix [Lemma 10]. It also helps us show that whenever Conflict is applicable, the last literal on the trail is a propagation or a theory propagation whose complement occurs in the conflict clause [Lemma 8]. This is essential for satisfying the third requirement of a reasonable strategy, which requires resolving with the last literal on the trail before backtracking. The lemma is therefore used to show that there are no stuck states in Theorem 15.

Lemma 7. *If $(\emptyset; \varepsilon; N'; \emptyset; 0; \top) \Rightarrow_{ICLF}^* (\Theta; \Gamma'; N; G; k; D \cdot \tau)$ using a reasonable strategy, then if $\Gamma' = \Gamma K$, there is no $C \cdot \sigma \in G$ such that $C\sigma$ is propositionally false in Γ , and if K is a decision, then there is no $C \cdot \sigma \in G$ such that $C\sigma$ is propositionally false in ΓK .*

Lemma 8. *If $(\emptyset; \varepsilon; N'; \emptyset; 0; \top) \Rightarrow_{ICLF}^* (\Theta; \Gamma; N; G; k; \top)$ using a reasonable strategy and Conflict is applicable to this state with $D \cdot \tau \in G$, then $\Gamma = \Gamma_1 K$, K is not a decision literal and $D\tau$ is not propositionally false in Γ_1 .*

Now we are ready to show that ICLF learns only non-redundant clauses when a reasonable strategy is used. First, we define an ordering with respect to which the learned clauses are non-redundant.

Definition 9 (Trail Induced Ordering). *Given a trail $\Gamma = L_1 \dots L_n$, we define the trail induced ordering $\prec_{\Gamma}$ as a well-founded, total and strict ordering in which $L_1 \prec_{\Gamma} \text{comp}(L_1) \prec_{\Gamma} L_2 \prec_{\Gamma} \text{comp}(L_2) \prec_{\Gamma} \dots \prec_{\Gamma} L_n \prec_{\Gamma} \text{comp}(L_n)$ and all literals that are propositionally undefined in Γ are larger than $\text{comp}(L_n)$.*

Lemma 10. *If $(\emptyset; \varepsilon; N'; \emptyset; 0; \top) \Rightarrow_{ICLF}^* (\Theta; \Gamma; N; G; k; D \cdot \tau)$ using a reasonable strategy and BacktrackClassic, BacktrackFOL or BacktrackCB is applicable to this state following a reasonable strategy, then $\Gamma = \Gamma_1 \Gamma_2$, $D\tau$ is propositionally false in Γ_1 and no clause in $\text{clauses}(G)$ is propositionally false in Γ_1 .*

Theorem 11 (Non-Redundant Learning). *Whenever ICLF learns a ground clause (using BacktrackClassic, BacktrackFOL or BacktrackCB) in a reasonable run, this clause is non-redundant with respect to the current set of ground instances and the trail induced ordering.*

The non-ground learned clause D , however, can be redundant with respect to N and the trail-induced ordering $\prec_{\Gamma}$. This can happen, for example, if there is an instance of a clause in N that is propositionally false for a trail prefix, but that is not contained in G ; one such case is shown in Example 12. Theorem 18 shows that if a stronger strategy, called first-order aware strategy [Definition 17], is followed, then the non-ground learned clause D is also non-redundant with respect to N . Moreover, Theorem 19 shows that following a first-order aware strategy does not cause any stuck states.

Example 12. Consider the clause set

$$N_0 = \{(N1) P(x_0), \quad (N2) P(x_1) \vee R(x_1), \quad (N3) Q(x_2) \vee \neg R(x_2)\}.$$

$$\begin{aligned} & (\emptyset; \varepsilon; N_0; \emptyset; 0; \top) \\ \Rightarrow_{ICLF}^{Instantiate \times 2} & (\Theta_0; \varepsilon; N_0; G_0; 0; \top) \end{aligned}$$

Here, $\Theta_0 = \{P(a), R(a), Q(a)\}$ and $G_0 = \{(G1) P(x_1) \vee R(x_1) \cdot \{x_1 \mapsto a\}, (G2) Q(x_2) \vee \neg R(x_2) \cdot \{x_2 \mapsto a\}\}$.

$$\begin{aligned} & \Rightarrow_{ICLF}^{Decide \times 2} (\Theta_0; \neg P(a)^1 \neg Q(a)^2; N_0; G_0; 2; \top) \\ & \Rightarrow_{ICLF}^{Propagate} (\Theta_0; \neg P(a)^1 \neg Q(a)^2 R(a)^{(G1)}; N_0; G_0; 2; \top) \\ & \Rightarrow_{ICLF}^{Conflict} (\Theta_0; \neg P(a)^1 \neg Q(a)^2 R(a)^{(G1)}; N_0; G_0; 2; Q(x_2) \vee \neg R(x_2) \cdot \{x_2 \mapsto a\}) \\ & \Rightarrow_{ICLF}^{Resolve} (\Theta_0; \neg P(a)^1 \neg Q(a)^2 R(a)^{(G1)}; N_0; G_0; 2; P(x_2) \vee Q(x_2) \cdot \{x_2 \mapsto a\}) \end{aligned}$$

The clause $P(x_2) \vee Q(x_2)$ that can now be learned is non-redundant with respect to G_0 , but redundant with respect to N_0 due to the clause $P(x_0)$. This also illustrates the difference between a reasonable and a first-order aware strategy. A first-order aware strategy prevents the decision $\neg P(a)$ since it makes an instance of $P(x_0) \in N_0$ propositionally false, whereas a reasonable strategy allows this decision because $P(a) \notin \text{clauses}(G_0)$.

Corollary 13. If $(\emptyset; \varepsilon; N'; \emptyset; 0; \top) \Rightarrow_{ICLF}^* (\Theta; \Gamma; N; G; k; D \cdot \tau)$ using a reasonable strategy, then there are no two $C \cdot \sigma, C' \cdot \sigma' \in G$ such that $C \cdot \sigma \neq C' \cdot \sigma'$, but $C\sigma = C'\sigma'$.

Without restrictions, termination cannot be expected for ICLF: Instantiate may generate infinitely many ground instances, $\mathcal{T}$ -Learn may add infinitely many theory lemmas, and $\mathcal{T}$ -Atom may add arbitrary ground atoms. Even ground reasoning on a fixed set of ground instances may diverge due to Restart and InstanceDel, the latter being analogous to Forget in CDCL(T). We therefore prove termination when these rules are applied only finitely often. Such restrictions are standard in related settings. For example, SCL restricts trail literals to be smaller than a fixed bound β , thereby obtaining only finitely many relevant instances [13]. In CDCL, restarts can be performed with increasing periodicity to ensure termination [30]. The following theorem states the corresponding termination condition for ICLF.

Theorem 14 (Termination). If ICLF is executed from a start state $(\emptyset; \varepsilon; N'; \emptyset; 0; \top)$ using a reasonable strategy, and the rules Instantiate, InstanceDel, Restart, $\mathcal{T}$ -Learn, and $\mathcal{T}$ -Atom are applied only finitely often, then it terminates.

Theorem 15 (No Stuck States). Let $(\Theta; \Gamma; N; G; k; D \cdot \tau)$ be a state reached by ICLF from a start state $(\emptyset; \varepsilon; N'; \emptyset; 0; \top)$ using a reasonable strategy. Then either $D = \perp$, or $D = \top$ and Γ is a $\mathcal{T}$ -satisfiable model of clauses(G), or a rule from the set $\{\text{Decide}, \text{Propagate}, \text{Conflict}, \text{T-Conflict}, \text{Explain}, \text{Resolve}, B\}$ is applicable following a reasonable strategy, where B is any of the rules BacktrackClassic, BacktrackFOL or BacktrackCB.

In full generality, completeness cannot be expected for arbitrary theories and theory combinations, for example for linear arithmetic combined with uninterpreted function symbols [2, 40, 24]. Nevertheless, Theorems 14 and 15 imply completeness whenever a finite $\mathcal{T}$ -unsatisfiable set of ground instances is available.

Corollary 16 (Completeness). *Let N be a clause set and $S \subseteq \text{gnd}(N)$ be finite and $\mathcal{T}$ -unsatisfiable. Then ICLF can derive $\perp$.*

Definition 17. *A strategy for ICLF is called first-order aware if it is reasonable and the following are true:*

1. *If Instantiate is applicable with a propositionally false instance, then this is always preferred over applying $\mathcal{T}$ -Conflict.*
2. *At no point during the run is there a clause in the clause set that has an instance that is propositionally false for a proper trail prefix.*
3. *Whenever the rightmost literal on the trail is a decision, no clause in the clause set has an instance that is propositionally false on the trail.*

Theorem 18. *If $(\emptyset; \varepsilon; N'; \emptyset; 0; \top) \Rightarrow_{ICLF}^* (\Theta; \Gamma; N; G; k; D \cdot \tau)$ using a first-order aware strategy and one of the rules *BacktrackClassic*, *BacktrackFOL* and *BacktrackCB* is applicable to this state following a first-order aware strategy, then the clause D added to N by the respective backtrack rule is non-redundant with respect to N and the trail induced ordering $\prec_\Gamma$.*

Theorem 19. *If $(\emptyset; \varepsilon; N'; \emptyset; 0; \top) \Rightarrow_{ICLF}^* (\Theta; \Gamma; N; G; k; D \cdot \tau)$ using a first-order aware strategy, then either $D = \perp$, or $D = \top$, Γ is a $\mathcal{T}$ -satisfiable model of $\text{clauses}(G)$ and no instance of a clause in N is propositionally false in Γ , or a rule from the set $\{\text{Instantiate}, \text{Decide}, \text{Propagate}, \text{Conflict}, \mathcal{T}\text{-Conflict}, \text{Explain}, \text{Resolve}, \text{BacktrackFOL}\}$ is applicable following a first-order aware strategy. Additionally, *Instantiate* only has to be applied with instances whose atoms are already contained in Θ .*

It should be noted that although following a first-order aware strategy does not cause any stuck states [Theorem 19], it limits the applicability of some rules. For instance, *Propagate* and $\mathcal{T}$ -*Propagate* cannot be applied if there is an uninstantiated false instance; instead, this false instance has to be instantiated. Additionally, *Decide* cannot be applied if doing so would make some instance false. In this sense, the requirements of a first-order aware strategy in ICLF are very similar to the requirements of a regular run in SCL(FOL) [13]. Moreover, the first-order aware strategy also restricts the applicability of $\mathcal{T}$ -Learn. If a clause that has a propositionally false instance should be added by $\mathcal{T}$ -Learn, *Restart* can first be used to go back to a point on the trail at which no instance is propositionally false. Beyond that, while detecting conflicts and propagations with respect to ground clause sets such as G is straightforward, both tasks are NP-complete in the worst case with respect to first-order clause sets such as N ; nevertheless, the recent lifting of the two-watched literal scheme to first-order logic shows that such clause sets can still be handled efficiently in practice [8].

4 Simulation of Other Calculi

In this section, we show that ICLF can simulate various other calculi. To avoid confusion, we mark variables from the simulated calculi with an overline, e.g., $\overline{N}$, and refer to variables from ICLF without any overlines, e.g., by N .

First, we show that ICLF can simulate SCL(FOL) [13]. This means that we associate each possible SCL(FOL) state with a set of ICLF states. Then, we show that for each state in a given SCL(FOL) run, we can find an associated ICLF state and ICLF rule applications between

these states. The simulation is linear in the sense that for each SCL(FOL) rule application, only a constant number of ICLF rule applications has to be performed. SCL(FOL) is an algorithm for first-order logic without equality, so we assume $\mathcal{T}$ to be the *free* theory, i.e., the theory that contains all Σ -algebras. For SCL(FOL), we write the status closure as $\overline{D} \cdot \overline{\tau}$, assuming that $\top = \top \cdot \{\}$ and $\perp = \perp \cdot \{\}$, just as we do for ICLF. Let $\prec_B$ be a well-founded, total and strict ordering on ground atoms such that for any ground atom A , there are only finitely many ground atoms A' with $A' \prec_B A$. $\prec_B$ is lifted to literals by comparing the respective atoms and to clauses by its multiset extension.

Definition 20. *We say that an ICLF state $(\Theta; \Gamma; N; G; k; D \cdot \tau)$ is associated with a given SCL(FOL) state $(\overline{\Gamma}; \overline{N}; \overline{U}; \overline{\beta}; \overline{k}; \overline{D} \cdot \overline{\tau})$ if all of the following hold:*

1. $D \cdot \tau = \overline{D} \cdot \overline{\tau}$.
2. For all $A \in \Theta$, $A \prec_B \overline{\beta}$ and there is an atom $A' \in \text{atom}(N)$ such that A is a ground instance of A' .
3. $N = \overline{N} \cup \overline{U}$.
4. $k \geq \overline{k}$. If $\overline{D} \cdot \overline{\tau} = \top$, then $k = \overline{k}$.
5. If $\Gamma = K_1 \dots K_n$ and $\overline{\Gamma} = \overline{K}_1 \dots \overline{K}_{\overline{n}}$, then $n \geq \overline{n}$ and for all $i \in \{1, \dots, \overline{n}\}$, K_i and $\overline{K}_i$ match. By match, we mean that either $K_i = \overline{K}_i = L^j$ or that $K_i = L^{C \cdot \sigma}$, $\overline{K}_i = L^{\overline{C} \cdot \overline{\sigma}}$ and $\overline{C} \cdot \overline{\sigma}$ is the result of exhaustively factorizing L in $C \cdot \sigma$. If $\overline{D} \cdot \overline{\tau} = \top$, then $n = \overline{n}$.

Theorem 21 (ICLF Simulates SCL(FOL)). *Let N' be a first-order clause set and β' be a ground literal. Then for any regular SCL(FOL) run*

$$(\varepsilon; N'; \emptyset; \beta'; 0; \top) = S_0 \Rightarrow_{\text{SCL(FOL)}} S_1 \Rightarrow_{\text{SCL(FOL)}} \dots \Rightarrow_{\text{SCL(FOL)}} S_m,$$

there exists a reasonable ICLF run

$$(\emptyset; \varepsilon; N'; \emptyset; 0; \top) = T_0 \Rightarrow_{\text{ICLF}}^{\leq 3} T_1 \Rightarrow_{\text{ICLF}}^{\leq 3} \dots \Rightarrow_{\text{ICLF}}^{\leq 3} T_m$$

such that for all $i \in \{0, \dots, m\}$, T_i is associated with S_i . Here, $\Rightarrow_{\text{ICLF}}^{\leq 3}$ means that at most three ICLF rules are applied to reach the following state. Only the ICLF rules Instantiate, InstanceDel, Decide, Propagate, Conflict, $\mathcal{T}$ -Atom, Resolve, Factorize and BacktrackFOL are needed.

SCL(FOL) has been shown to simulate non-redundant ground superposition [12, 10]. Since ICLF simulates SCL(FOL), this result also transfers to ICLF. For ICLF, however, we can show a stronger result: it can simulate any resolution inference that does not yield a tautology, provided that neither parent clause becomes tautological under the most general unifier of the resolved literals. If a parent clause does become tautological, the resolvent is already subsumed by the other parent clause. This does not contradict the non-redundancy guarantee for reasonable strategies, because learned clauses are guaranteed to be non-redundant only with respect to G . Since ICLF does not force all clauses to be instantiated, a clause that is redundant with respect to N may still be non-redundant with respect to G .

Remark 22 (ICLF Simulates Resolution). *Let $(\Theta; \Gamma; N; G; k; \top)$ be an ICLF state reached in a reasonable run, and let $C_1 \vee L_1, C_2 \vee L_2 \in N$ be such that L_1 and $\text{comp}(L_2)$ are unifiable. Let $\mu = \text{mgu}(L_1, \text{comp}(L_2))$. If the resolvent of the clauses is not a tautology and neither*

parent clause becomes tautological under μ , then ICLF can learn it or a clause subsuming it by following a reasonable strategy as follows. We first apply *Restart* with $\Gamma_1 = \varepsilon$ to empty the trail, and then *InstanceDel* on all instances in G to empty the set of ground instances. Next, we apply *Instantiate* with $(C_1 \vee L_1) \cdot (\mu\sigma)$ and $(C_2 \vee L_2) \cdot (\mu\sigma)$, where we pick σ such that no two atoms $A_1, A_2 \in \text{atom}(C_1 \cup C_2 \cup \{L_1, L_2\})$ with $A_1\mu \neq A_2\mu$ become equal. This can, for example, be achieved by instantiating all variables with different fresh constants. Then, we decide the complements of all literals except for $L_1\mu\sigma$ and $L_2\mu\sigma$. This is possible because the resolvent of the clauses is not a tautology, because neither parent clause is tautological under μ , and because of the way we chose σ . It also does not contradict the reasonable strategy because neither of the two ground instances can be false at this point. We then propagate $L_1\mu\sigma$ and apply *Conflict* with $(C_2 \vee L_2) \cdot (\mu\sigma)$. Resolving exhaustively with $L_1\mu\sigma$ yields the desired resolvent, or, due to exhaustive factorization, a clause subsuming it. The derived clause is either $\perp$ or can be learned using *BacktrackClassic*.

Now, we show that ICLF can simulate clause learning in $\text{SCL}(\text{T})$ [11]. Unlike in the simulation of $\text{SCL}(\text{FOL})$, we do not simulate every rule application. Instead, we simulate the sequence of rule applications that leads to the application of *Conflict* and show that ICLF can learn the same clause that $\text{SCL}(\text{T})$ learns. We assume that a constrained clause $\Lambda \parallel C$ is just a different notation for the clause $\neg\Lambda \vee C$ and note the status closure for $\text{SCL}(\text{T})$ as $\overline{D} \cdot \overline{\tau}$, assuming that $\top = \top \cdot \{\}$. Let N' be a pure, abstracted clause set according to the requirements of $\text{SCL}(\text{T})$ and B' be a finite sequence of constants of background sorts.

Definition 23. We say that an ICLF state $(\Theta; \Gamma; N; G; k; D \cdot \tau)$ is weakly associated with a given $\text{SCL}(\text{T})$ state $(\overline{M}; \overline{N}; \overline{U}; \overline{B}; \overline{k}; \overline{D} \cdot \overline{\tau})$ if $N = \overline{N} \cup \overline{U}$, $D \cdot \tau = \overline{D} \cdot \overline{\tau}$, and the background literals in Γ form a prefix of $\overline{\Gamma}$. It is called associated if additionally, the set of literals in $\overline{M}$ is a subset of the set of literals in Γ , the foreground literals in $\overline{M}$ form a subsequence of Γ , and for all propagations $L^{\overline{C} \cdot \overline{\sigma}}$ in $\overline{M}$, the corresponding literal L in Γ is annotated with a closure $C \cdot \sigma$ such that $\overline{C} \cdot \overline{\sigma}$ is the result of exhaustively factorizing L in $C \cdot \sigma$, and for all decisions L^k in $\overline{M}$, the corresponding literal L in Γ is also a decision.

Theorem 24 (Simulation of Conflict). For any regular $\text{SCL}(\text{T})$ run

$$(\varepsilon; N'; \emptyset; B'; 0; \top) \Rightarrow_{\text{SCL}(\text{T})}^* S_0 \Rightarrow_{\text{SCL}(\text{T})}^{*(\text{no Conf})} S_1 \Rightarrow_{\text{SCL}(\text{T})}^{\text{Conflict}} S_2$$

where S_0 is not a conflict state, and any state T_0 reached from $(\emptyset; \varepsilon; N'; \emptyset; 0; \top)$ in a reasonable ICLF run and weakly associated with S_0 , there exists a sequence of reasonable ICLF rule applications of the rules *Instantiate*, *InstanceDel*, *T-Atom*, *Restart*, *Decide* and *Propagate* from T_0 to a state T_1 that is associated with S_1 , and *Conflict* can be applied to T_1 following a reasonable strategy, reaching a state T_2 that is associated with S_2 .

Theorem 25 (Simulation of Learning). Consider a regular $\text{SCL}(\text{T})$ run

$$(\varepsilon; N'; \emptyset; B'; 0; \top) \Rightarrow_{\text{SCL}(\text{T})}^* S_0 \Rightarrow_{\text{SCL}(\text{T})}^{\text{Conflict}} S_1 \Rightarrow_{\text{SCL}(\text{T})}^{\{\text{Resolve}, \text{Factorize}, \text{Skip}\}^*} S_2$$

and a state T_1 reached from $(\emptyset; \varepsilon; N'; \emptyset; 0; \top)$ in a reasonable ICLF run and associated with S_1 . Then there is a state T_2 reachable from T_1 by applying the rules *Resolve* and *Factorize* in a reasonable way such that T_2 is associated with S_2 . Moreover, if *Backtrack* can be applied to S_2 in a regular run, reaching a state S_3 , then *BacktrackClassic* is applicable to T_2 following a reasonable strategy, reaching a state T_3 that is weakly associated with S_3 .

Together, Theorems 24 and 25 show how ICLF can simulate a given regular SCL(T) derivation of the empty clause. Clearly, the start state of ICLF is weakly associated with the start state of SCL(T). Whenever Conflict is applied in SCL(T), ICLF can reach an associated conflict state by Theorem 24. Then, all applications of Resolve, Factorize and Skip during conflict analysis in SCL(T) can be simulated in ICLF by Theorem 25. If SCL(T) derives the empty clause during conflict analysis, ICLF also derives it because the states are shown to be associated. If SCL(T) learns a clause, then by Theorem 25, ICLF can learn the same clause, reaching a weakly associated state from which the described procedure can be iterated.

Next, we show that ICLF can simulate the Isabelle/HOL verified calculus CDCL_W [7]. CDCL is an algorithm for pure propositional logic, so we assume $\mathcal{T}$ to be the free theory. For easier notation, given a ground clause D , we assume that $D = D \cdot \{\}$. To accommodate the different notations in ICLF and CDCL_W, we assume that $L^j = L^\dagger$ for ground literals L , and that the trail in CDCL_W grows to the right instead of to the left.

Definition 26. *We say that an ICLF state $(\Theta; \Gamma; N; G; k; D \cdot \tau)$ is associated with a given CDCL_W state $(\bar{M}; \bar{N}; \bar{U}; \bar{D})$ if $\text{clauses}(G) = \bar{N} \cup \bar{U}$, $D \cdot \tau = \bar{D}$, when $D \cdot \tau = \top$, then $\Gamma = \bar{M}$, and when $D \cdot \tau \neq \top$, then $\bar{M}$ is a prefix of Γ .*

From its start state, ICLF first has to instantiate all given clauses to reach a state that is associated with the CDCL_W start state. Then, each rule application in CDCL_W can be simulated by at most one rule application in ICLF.

Theorem 27 (ICLF Simulates CDCL_W). *Let N' be a propositional clause set. Then for any reasonable CDCL_W run*

$$(\varepsilon; N'; \emptyset; \top) = S_0 \Rightarrow_{\text{CDCL}_W} S_1 \Rightarrow_{\text{CDCL}_W} \dots \Rightarrow_{\text{CDCL}_W} S_m,$$

there exists a reasonable ICLF run

$$(\emptyset; \varepsilon; N'; \emptyset; 0; \top) \Rightarrow_{\text{ICLF}}^{\text{Instantiate} *} T_0 \Rightarrow_{\text{ICLF}}^{\leq 1} T_1 \Rightarrow_{\text{ICLF}}^{\leq 1} \dots \Rightarrow_{\text{ICLF}}^{\leq 1} T_m$$

such that for all $i \in \{0, \dots, m\}$, T_i is associated with S_i . After the initial applications of Instantiate, only the ICLF rules Propagate, Decide, Conflict, Restart, InstanceDel, Resolve and BacktrackClassic are needed.

In standard presentations of CDCL(T) [30, 4], conflict analysis is left flexible, allowing different learning mechanisms. We therefore do not state a formal simulation theorem for CDCL(T). Since ICLF learns only non-redundant clauses, such a theorem would have to impose a compatible condition on learning in CDCL(T). We can nevertheless describe how ICLF, given a theory solver for $\mathcal{T}$, can be used to decide the satisfiability of a ground clause set modulo $\mathcal{T}$.

Remark 28 (ICLF Simulates CDCL(T)). *Let N' be a set of ground clauses modulo a theory $\mathcal{T}$. We start in the state $(\emptyset; \varepsilon; N'; \emptyset; 0; \top)$ and first apply Instantiate with $\{\}$ to all clauses in N' . Now, by Theorems 14 and 15, applying the rules Decide, Propagate, Conflict, Restart, $\mathcal{T}$ -Propagate, $\mathcal{T}$ -Learn, $\mathcal{T}$ -Atom, $\mathcal{T}$ -Conflict, Explain, Resolve and BacktrackClassic according to a reasonable strategy and restricting Restart, $\mathcal{T}$ -Learn and $\mathcal{T}$ -Atom to a finite number of applications, ICLF either derives the empty clause, or it terminates in a state whose trail is a $\mathcal{T}$ -satisfiable model of N' .*

For instantiation-based SMT procedures combining quantifier instantiation with CDCL(T), we are not aware of any formal description. Nevertheless, ICLF can model the essential interaction between an instantiation module and a CDCL(T) solver. The process consists of

two phases: the instantiation phase and the CDCL(T) phase. First, the instantiation module generates some instances, including all of the initially ground clauses, using Instantiate. Then, CDCL(T) is run according to Remark 28. If it derives unsatisfiability, the process terminates. Otherwise, the instantiation module adds more instances. In ICLF, due to non-ground learning, the instantiation module can also instantiate learned clauses, which is not possible in classical instantiation-based SMT solving.

There are other calculi that integrate explicit model building and a form of clause learning. Theory instantiation [18] extends the InstGen [17] calculus with theories. Still, only instances of existing clauses are learned with respect to ground theory conflicts. The model-building in model evolution with lemma learning [5] goes beyond ground literals but does not involve theories. Lemma learning gets more complicated and no non-redundancy guarantees are presented. In conflict resolution [38] models are built by first-order decision literals and propagations computed by the unit propagation rule. The conflict resolution rule and the overall calculus are described in an abstract way, such that no quality guarantees are possible with respect to newly generated clauses.

5 Conclusion

We presented the calculus ICLF for non-ground SMT, which generalizes several model-based approaches for propositional logic, first-order logic, and first-order logic modulo theories. Most notably, ICLF extends quantifier instantiation with non-ground learning and a tighter integration of instantiation and model building. It also extends SCL by allowing a lazier treatment of non-ground clauses: conflicts only have to be considered for the current ground instances. In this way, ICLF provides a flexible setting for combining techniques from instantiation-based SMT solving and non-ground clause learning. The long-term goal is to use this flexibility to develop new methods for solving problems that are beyond the reach of current solvers.

The main direction for future work is to implement ICLF and integrate its ideas into existing solvers. Since ICLF generalizes multiple other calculi, it naturally leaves freedom in several choices. One such choice is which ground instances to create and when to create them. This difficulty already arises in quantifier instantiation and in SCL, and we hope that instantiation techniques that have proved useful in SMT solving [15, 14, 20, 37, 36] can also be adapted to ICLF. Another such choice is when, and to what extent, to check the applicability of theory rules, for instance when to perform complete checks. This depends on the specific theory, and we expect that strategies that work well in CDCL(T) will also work well in ICLF. As in all model-based calculi, there is also freedom in the choice of which literals to include in the model, that is, which decisions and propagations to perform. Unlike in SCL, however, decisions and propagations are restricted by the current ground instances in ICLF. Finally, there is freedom in the choice of how to explain theory propagations and conflicts. In CDCL(T), it is sufficient to explain a theory conflict by a minimal subset of contradictory trail literals, for example the literals $h(a) = g(a)$, $h(a) = a$, and $g(a) \neq a$ from Example 4. In ICLF, this ground explanation can be used, but it may be preferable to use the more general explanation $x_2 \neq x_3 \vee x_2 \neq x_4 \vee x_3 = x_4$, a theory lemma expressing the transitivity axiom for equality. Instantiated with the substitution $\{x_2 \mapsto h(a), x_3 \mapsto g(a), x_4 \mapsto a\}$, this yields the same ground explanation, but at the non-ground level it allows us to learn a more general clause from the conflict. If we use the ground explanation directly, by contrast, then all variables matched against the ground explanation are grounded, and this restricts our ability to learn non-ground clauses. Thus, besides instantiation heuristics, a central challenge for implementing ICLF is to develop theory solvers that can produce useful non-ground explanations [3, 25].

This gives another perspective on the role of theory explanations in ICLF. In saturation-based theorem proving, adding theory axioms as ordinary clauses can greatly enlarge the search space [34, 21, 23]. In ICLF, theory axioms need not be added to N as clauses available for arbitrary instantiation, propagation, and resolution. Instead, theory lemmas only have to be produced when they become relevant during conflict analysis. In this way, the ground search guides the use of theory lemmas in resolution, rather than making all theory axioms available throughout the search. This allows theory reasoning to contribute non-ground theory lemmas when the ground search needs them, combining the control of model-guided reasoning with the generality of first-order learning.

Acknowledgements We thank our anonymous reviewers for their constructive feedback.

References

- [1] Leo Bachmair and Harald Ganzinger. Rewrite-based equational theorem proving with selection and simplification. *Journal of Logic and Computation*, 4(3):217–247, 1994.
- [2] Leo Bachmair, Harald Ganzinger, and Uwe Waldmann. Refutational theorem proving for hierarchic first-order theories. *Appl. Algebra Eng. Commun. Comput.*, 5:193–212, 1994.
- [3] Haniel Barbosa, Andrew Reynolds, Gereon Kremer, Hanna Lachnitt, Aina Niemetz, Andres Nötzli, Alex Ozdemir, Mathias Preiner, Arjun Viswanathan, Scott Viteri, Yoni Zohar, Cesare Tinelli, and Clark W. Barrett. Flexible proof production in an industrial-strength SMT solver. In Jasmin Blanchette, Laura Kovács, and Dirk Pattinson, editors, *Automated Reasoning - 11th International Joint Conference, IJCAR 2022, Haifa, Israel, August 8-10, 2022, Proceedings*, volume 13385 of *Lecture Notes in Computer Science*, pages 15–35. Springer, 2022.
- [4] Clark W. Barrett, Roberto Sebastiani, Sanjit A. Seshia, and Cesare Tinelli. Satisfiability modulo theories. In Armin Biere, Marijn Heule, Hans van Maaren, and Toby Walsh, editors, *Handbook of Satisfiability - Second Edition*, volume 336 of *Frontiers in Artificial Intelligence and Applications*, pages 1267–1329. IOS Press, 2021.
- [5] Peter Baumgartner, Alexander Fuchs, and Cesare Tinelli. Lemma learning in the model evolution calculus. In Miki Hermann and Andrei Voronkov, editors, *Logic for Programming, Artificial Intelligence, and Reasoning, 13th International Conference, LPAR 2006, Phnom Penh, Cambodia, November 13-17, 2006, Proceedings*, volume 4246 of *Lecture Notes in Computer Science*, pages 572–586. Springer, 2006.
- [6] Nikolaj S. Bjørner. SMT solvers: Foundations and applications. In Javier Esparza, Orna Grumberg, and Salomon Sickert, editors, *Dependable Software Systems Engineering*, volume 45 of *NATO Science for Peace and Security Series - D: Information and Communication Security*, pages 24–32. IOS Press, 2016.
- [7] Jasmin Christian Blanchette, Mathias Fleury, Peter Lammich, and Christoph Weidenbach. A verified SAT solver framework with learn, forget, restart, and incrementality. *J. Autom. Reason.*, 61(1-4):333–365, 2018.
- [8] Yasmine Briefs, Martin Bromberger, Tobias Gehl, Lorenz Leutgeb, Simon Schwarz, and Christoph Weidenbach. A two-watched literal scheme for first-order logic. In Armin Biere, Carsten Lutz, and Sara Negri, editors, *Automated Reasoning - 13th International Joint Conference, IJCAR 2026, Lisbon, Portugal, July 26-29, 2026, Proceedings, Part II*, volume 16689 of *Lecture Notes in Computer Science*, pages 20–36. Springer, 2026.
- [9] Yasmine Briefs and Christoph Weidenbach. Extending SMT solving with non-ground clause learning. *CoRR*, abs/2609.11509, 2026.
- [10] Martin Bromberger, Martin Desharnais, and Christoph Weidenbach. A stepwise refinement proof that SCL(FOL) simulates ground ordered resolution. In Clark W. Barrett and Uwe Waldmann, ed-

- itors, *Automated Deduction - CADE 30 - 30th International Conference on Automated Deduction, Stuttgart, Germany, July 28-31, 2025, Proceedings*, volume 15943 of *Lecture Notes in Computer Science*, pages 302–322. Springer, 2025.
- [11] Martin Bromberger, Alberto Fiori, and Christoph Weidenbach. Deciding the Bernays-Schoenfinkel fragment over bounded difference constraints by simple clause learning over theories. In Fritz Henglein, Sharon Shoham, and Yakir Vizel, editors, *Verification, Model Checking, and Abstract Interpretation - 22nd International Conference, VMCAI 2021, Copenhagen, Denmark, January 17-19, 2021, Proceedings*, volume 12597 of *Lecture Notes in Computer Science*, pages 511–533. Springer, 2021.
 - [12] Martin Bromberger, Chaahat Jain, and Christoph Weidenbach. SCL(FOL) can simulate non-redundant superposition clause learning. In Brigitte Pientka and Cesare Tinelli, editors, *Automated Deduction - CADE 29 - 29th International Conference on Automated Deduction, Rome, Italy, July 1-4, 2023, Proceedings*, volume 14132 of *Lecture Notes in Computer Science*, pages 134–152. Springer, 2023.
 - [13] Martin Bromberger, Simon Schwarz, and Christoph Weidenbach. SCL(FOL) revisited. *CoRR*, abs/2302.05954, 2023.
 - [14] Leonardo Mendonça de Moura and Nikolaj S. Bjørner. Efficient e-matching for SMT solvers. In Frank Pfenning, editor, *Automated Deduction - CADE-21, 21st International Conference on Automated Deduction, Bremen, Germany, July 17-20, 2007, Proceedings*, volume 4603 of *Lecture Notes in Computer Science*, pages 183–198. Springer, 2007.
 - [15] David Detlefs, Greg Nelson, and James B. Saxe. Simplify: a theorem prover for program checking. *J. ACM*, 52(3):365–473, 2005.
 - [16] Alberto Fiori and Christoph Weidenbach. SCL clause learning from simple models. In Pascal Fontaine, editor, *Automated Deduction - CADE 27 - 27th International Conference on Automated Deduction, Natal, Brazil, August 27-30, 2019, Proceedings*, volume 11716 of *Lecture Notes in Computer Science*, pages 233–249. Springer, 2019.
 - [17] Harald Ganzinger and Konstantin Korovin. New directions in instantiation-based theorem proving. In Samson Abramsky, editor, *18th Annual IEEE Symposium on Logic in Computer Science, LICS'03*, pages 55–64. IEEE Computer Society, 2003.
 - [18] Harald Ganzinger and Konstantin Korovin. Theory instantiation. In Miki Hermann and Andrei Voronkov, editors, *Logic for Programming, Artificial Intelligence, and Reasoning, 13th International Conference, LPAR 2006, Phnom Penh, Cambodia, November 13-17, 2006, Proceedings*, volume 4246 of *Lecture Notes in Computer Science*, pages 497–511. Springer, 2006.
 - [19] Yeting Ge, Clark W. Barrett, and Cesare Tinelli. Solving quantified verification conditions using satisfiability modulo theories. In Frank Pfenning, editor, *Automated Deduction - CADE-21, 21st International Conference on Automated Deduction, Bremen, Germany, July 17-20, 2007, Proceedings*, volume 4603 of *Lecture Notes in Computer Science*, pages 167–182. Springer, 2007.
 - [20] Yeting Ge and Leonardo Mendonça de Moura. Complete instantiation for quantified formulas in satisfiability modulo theories. In Ahmed Bouajjani and Oded Maler, editors, *Computer Aided Verification, 21st International Conference, CAV 2009, Grenoble, France, June 26 - July 2, 2009. Proceedings*, volume 5643 of *Lecture Notes in Computer Science*, pages 306–320. Springer, 2009.
 - [21] Bernhard Gleiss and Martin Suda. Layered clause selection for theory reasoning - (short paper). In Nicolas Peltier and Viorica Sofronie-Stokkermans, editors, *Automated Reasoning - 10th International Joint Conference, IJCAR 2020, Paris, France, July 1-4, 2020, Proceedings, Part I*, volume 12166 of *Lecture Notes in Computer Science*, pages 402–409. Springer, 2020.
 - [22] Dejan Jovanovic and Clark W. Barrett. Polite theories revisited. In Christian G. Fermüller and Andrei Voronkov, editors, *Logic for Programming, Artificial Intelligence, and Reasoning - 17th International Conference, LPAR-17, Yogyakarta, Indonesia, October 10-15, 2010. Proceedings*, volume 6397 of *Lecture Notes in Computer Science*, pages 402–416. Springer, 2010.
 - [23] Konstantin Korovin, Laura Kovács, Giles Reger, Johannes Schoisswohl, and Andrei Voronkov.

- ALASCA: reasoning in quantified linear arithmetic. In Sriram Sankaranarayanan and Natasha Sharygina, editors, *Tools and Algorithms for the Construction and Analysis of Systems - 29th International Conference, TACAS 2023, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2023, Paris, France, April 22-27, 2023, Proceedings, Part I*, volume 13993 of *Lecture Notes in Computer Science*, pages 647–665. Springer, 2023.
- [24] Konstantin Korovin and Andrei Voronkov. Integrating linear arithmetic into superposition calculus. In Jacques Duparc and Thomas A. Henzinger, editors, *Computer Science Logic, 21st International Workshop, CSL 2007, 16th Annual Conference of the EACSL, Lausanne, Switzerland, September 11-15, 2007, Proceedings*, volume 4646 of *Lecture Notes in Computer Science*, pages 223–237. Springer, 2007.
- [25] Hendrik Leidinger and Christoph Weidenbach. Computing ground congruence classes. In Clark W. Barrett and Uwe Waldmann, editors, *Automated Deduction - CADE 30 - 30th International Conference on Automated Deduction, Stuttgart, Germany, July 28-31, 2025, Proceedings*, volume 15943 of *Lecture Notes in Computer Science*, pages 594–613. Springer, 2025.
- [26] João Marques-Silva, Inês Lynce, and Sharad Malik. Conflict-driven clause learning SAT solvers. In Armin Biere, Marijn Heule, Hans van Maaren, and Toby Walsh, editors, *Handbook of Satisfiability - Second Edition*, volume 336 of *Frontiers in Artificial Intelligence and Applications*, pages 133–182. IOS Press, 2021.
- [27] Sibylle Möhle and Armin Biere. Backing backtracking. In Mikolás Janota and Inês Lynce, editors, *Theory and Applications of Satisfiability Testing - SAT 2019 - 22nd International Conference, SAT 2019, Lisbon, Portugal, July 9-12, 2019, Proceedings*, volume 11628 of *Lecture Notes in Computer Science*, pages 250–266. Springer, 2019.
- [28] Alexander Nadel and Vadim Ryvchin. Chronological backtracking. 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 111–121. Springer, 2018.
- [29] Greg Nelson and Derek C. Oppen. Simplification by cooperating decision procedures. *ACM Trans. Program. Lang. Syst.*, 1(2):245–257, 1979.
- [30] Robert Nieuwenhuis, Albert Oliveras, and Cesare Tinelli. Solving SAT and SAT modulo theories: From an abstract Davis–Putnam–Logemann–Loveland procedure to DPLL(T). *J. ACM*, 53(6):937–977, 2006.
- [31] Robert Nieuwenhuis and Albert Rubio. Paramodulation-based theorem proving. In John Alan Robinson and Andrei Voronkov, editors, *Handbook of Automated Reasoning (in 2 volumes)*, pages 371–443. Elsevier and MIT Press, 2001.
- [32] Juan Antonio Navarro Pérez and Andrei Voronkov. Proof systems for effectively propositional logic. In Alessandro Armando, Peter Baumgartner, and Gilles Dowek, editors, *Automated Reasoning, 4th International Joint Conference, IJCAR 2008, Sydney, Australia, August 12-15, 2008, Proceedings*, *Lecture Notes in Computer Science*, pages 426–440. Springer, 2008.
- [33] Silvio Ranise, Christophe Ringeissen, and Calogero G. Zarba. Combining data structures with nonstably infinite theories using many-sorted logic. In Bernhard Gramlich, editor, *Frontiers of Combining Systems, 5th International Workshop, FroCoS 2005, Vienna, Austria, September 19-21, 2005, Proceedings*, volume 3717 of *Lecture Notes in Computer Science*, pages 48–64. Springer, 2005.
- [34] Giles Reger and Martin Suda. Set of support for theory reasoning. In Thomas Eiter, David Sands, Geoff Sutcliffe, and Andrei Voronkov, editors, *IWIL@LPAR 2017 Workshop and LPAR-21 Short Presentations, Maun, Botswana, May 7-12, 2017*, volume 1 of *Kalpa Publications in Computing*, pages 124–134. EasyChair, 2017.
- [35] Andrew Reynolds. Conflicts, models and heuristics for quantifier instantiation in SMT. In Laura Kovács and Andrei Voronkov, editors, *Vampire@IJCAR 2016. Proceedings of the 3rd Vampire*

- Workshop, Coimbra, Portugal, July 2, 2016*, volume 44 of *EPiC Series in Computing*, pages 1–15. EasyChair, 2016.
- [36] Andrew Reynolds, Haniel Barbosa, and Pascal Fontaine. Revisiting enumerative instantiation. 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 112–131. Springer, 2018.
 - [37] Andrew Reynolds, Cesare Tinelli, and Leonardo Mendonça de Moura. Finding conflicting instances of quantified formulas in SMT. In *Formal Methods in Computer-Aided Design, FMCAD 2014, Lausanne, Switzerland, October 21-24, 2014*, pages 195–202. IEEE, 2014.
 - [38] John K. Slaney and Bruno Woltzenlogel Paleo. Conflict resolution: A first-order resolution calculus with decision literals and conflict-driven clause learning. *J. Autom. Reason.*, 60(2):133–156, 2018.
 - [39] Cesare Tinelli and Calogero G. Zarba. Combining nonstably infinite theories. *J. Autom. Reason.*, 34(3):209–238, 2005.
 - [40] Uwe Waldmann. Superposition and chaining for totally ordered divisible abelian groups (Extended abstract). In Rajeev Goré, Alexander Leitsch, and Tobias Nipkow, editors, *Automated reasoning : First International Joint Conference, IJCAR 2001*, volume 2083 of *LNAI*, pages 226–241, Siena, Italy, 2001. Springer.
 - [41] Christoph Weidenbach. Automated reasoning building blocks. In Roland Meyer, André Platzer, and Heike Wehrheim, editors, *Correct System Design - Symposium in Honor of Ernst-Rüdiger Olderog on the Occasion of His 60th Birthday, Oldenburg, Germany, September 8-9, 2015. Proceedings*, *Lecture Notes in Computer Science*, pages 172–188. Springer, 2015.