SCL(FOL) Can Simulate Non-Redundant Superposition Clause Learning

Martin Bromberger, Chaahat Jain, Christoph Weidenbach

SCL(FOL) Can Simulate Non-Redundant Superposition Clause Learning

Martin Bromberger[1], Chaahat Jain[1,2], and Christoph Weidenbach[1(✉)]

[1] Max Planck Institute for Informatics, Saarbrücken, Germany
{mbromber,cjain,weidenbach}@mpi-inf.mpg.de
[2] Graduate School of Computer Science, Saarbrücken, Germany

Abstract. We show that SCL(FOL) can simulate the derivation of non-redundant clauses by superposition for first-order logic without equality. Superposition-based reasoning is performed with respect to a fixed reduction ordering. The completeness proof of superposition relies on the grounding of the clause set. It builds a ground partial model according to the fixed ordering, where minimal false ground instances of clauses then trigger non-redundant superposition inferences. We define a respective strategy for the SCL calculus such that clauses learned by SCL and superposition inferences coincide. From this perspective the SCL calculus can be viewed as a generalization of the superposition calculus.

Keywords: first-order reasoning · superposition · SCL · non-redundant clause learning

1 Introduction

Superposition [1,2,18] is currently considered as the prime calculus for first-order logic reasoning where all leading first-order theorem provers implement a variant thereof [14,16,20,22]. More recently, the family of SCL calculi (Clause Learning from Simple Models, or just Simple Clause Learning) [4,8,9,11,17] was introduced. There are first experimental results [3] available, and first steps towards an overall implementation [5,7].

The main differences between superposition and SCL for first-order logic without equality are: (i) superposition assumes a fixed ordering on literals whereas the ordering in SCL is dynamic and evolves out of the satisfiability of clauses, (ii) superposition performs single superposition left and factoring inferences whereas SCL typically performs several such inferences to derive a single learned clause, (iii) the superposition model operator is not effective on the non-ground clause level whereas the SCL model assumption is effective. For first-order logic without equality superposition reduces to ordered resolution combined with the powerful superposition redundancy criterion. Our simulation result cannot be one-to-one because an SCL learned clause is typically generated by several superposition inferences and superposition factoring inferences are performed by SCL only in the context of resolution inferences. The simulation result considers the

B. Pientka and C. Tinelli (Eds.): CADE 2023, LNAI 14132, pp. 134–152, 2023.
https://doi.org/10.1007/978-3-031-38499-8_8

ground case, where the superposition strategy used in the completeness proof only triggers non-redundant inferences [1]. We call this strategy SUP-MO, Definition 5. Overall first-order superposition completeness is then obtained by a lifting argument to the non-ground clause level. We actually show that a superposition refutation of some ground clause set can be simulated by an SCL refutation on the same clause set, such that they coincide on all superposition left (ordered resolution) inferences. For the superposition calculus we refer to [1] and for SCL to [9] where all main properties of both calculi have meanwhile been verified inside the Isabelle framework [10,19,21].

For example, consider a superposition refutation of the simple ground clause set

$$N^0_{\text{SUP}} = \{(C_1)\ P(a) \vee P(a), \quad (C_2)\ \neg P(a) \vee Q(b), \quad (C_3)\ \neg Q(b)\}$$

with respect to a KBO [13], where all symbols have weight one, and precedence $a \prec b \prec P \prec Q$. Superposition generates only non-redundant clauses. Then with respect to the usual superposition ordering extension to literals and clauses we get $(C_1) \prec_{\text{KBO-SUP}} (C_2) \prec_{\text{KBO-SUP}} (C_3)$ and the superposition model operator produces the Herbrand model $N^0_{\text{SUP},\mathcal{I}} = \emptyset$. Now clause (C_1) is the minimal false clause, triggering a factoring inference resulting in $(C_4)\ P(a)$ and clause set $N^1_{\text{SUP}} = N^0_{\text{SUP}} \cup \{(C_4)\ P(a)\}$. The clause $P(a)$ cannot be derived by SCL because factoring is only preformed in the context of resolution inferences. Now (C_4) is the smallest clause in N^1_{SUP} and the superposition model operator produces $N^1_{\text{SUP},\mathcal{I}} = \{P(a), Q(b)\}$ with minimal false clause (C_3). A superposition left inference between (C_3) and (C_2) generates $(C_5)\ \neg P(a)$ and $N^2_{\text{SUP}} = N^1_{\text{SUP}} \cup \{(C_5)\ \neg P(a)\}$. The generation of $\neg P(a)$ can now be simulated by SCL by constructing the SCL trail $[P(a)^1 Q(b)^{\{\neg P(a) \vee Q(b)\}}]$ out of $N^0_{\text{SUP}} = N^0_{\text{SCL}}$ leading to the learned clause $(C_5)\ \neg P(a)$ and respective clause set $N^2_{\text{SCL}} = N^0_{\text{SCL}} \cup \{(C_5)\ \neg P(a)\}$. Note that $P(a)$ could have also been propagated, see Sect. 2 rule Propagate, but this would eventually not lead to the learned clause $(C_5)\ \neg P(a)$ but $\bot$. Finally, the superposition model operator produces $N^2_{\text{SUP},\mathcal{I}} = \{P(a), Q(b)\}$ with minimal false clause (C_5) and infers $\bot$. The SCL simulation generates the trail $[P(a)^{\{P(a)\}}]$ and then learns $\bot$ as well out of a conflict with (C_5). Note that this SCL trail is based on a factoring of (C_1) to $P(a)$ that was the explicit first step of the superposition refutation. Recall that by using an exhaustive propagation strategy, SCL would start with the trail $[P(a)^{P(a)} Q(b)^{\{\neg P(a) \vee Q(b)\}}]$ and immediately derive $\bot$. Exhaustive propagation is not a good strategy in general, because first-order logic clauses may enable infinitely many propagations. Even together with the typical SCL restriction to finitely many ground instances, there are exponentially many propagations possible, in general. Therefore, the *regular* strategy defined in [9] does not require exhaustive propagation, but guarantees non-redundant clause learning. The SCL-SUP strategy, Definition 8, and Definition 10, simulating superposition SUP-MO runs is also a regular strategy, Lemma 17.

The paper is now organized as follows. After repetition of the needed concepts of SCL and superposition, Sect. 2, the simulation result is contained in Sect. 3. We show that any superposition refutation of a ground clause set producing only non-redundant inferences through the SUP-MO strategy, can be simulated via

the SCL-SUP strategy. Based on the 14 simulation invariants of Definition 7, we show the invariants by an inductive argument on the length of the superposition refutation, starting from the initial state, Lemma 13, for intermediate superposition inference steps Lemma 14, until the final refutation Lemma 15, and Lemma 16. For the simulation we do not consider selection in superposition inferences in favor of a less complicated presentation. The paper ends with a discussion of the obtained results. A full version of the paper including all proofs is available on arxiv [6].

2 Preliminaries

We assume a first-order language without equality where N denotes a clause set; C, D denote clauses; L, K, H denote literals; A, B denote atoms; P, Q, R denote predicates; t, s terms; f, g, h function symbols; a, b, c constants; and x, y, z variables. Atoms, literals, clauses and clause sets are considered as usual, where in particular clauses are identified both with their disjunction and multiset of literals [9]. The complement of a literal is denoted by the function comp. The function atom(L) denotes the atomic part of a literal. Semantic entailment $\models$ is defined as usual where variables in clauses are assumed to be universally quantified. Substitutions σ, τ are total mappings from variables to terms, where $\mathrm{dom}(\sigma) := \{x \mid x\sigma \neq x\}$ is finite and $\mathrm{codom}(\sigma) := \{t \mid x\sigma = t, x \in \mathrm{dom}(\sigma)\}$. Their application is extended to literals, clauses, and sets of such objects in the usual way. A term, atom, clause, or a set of these objects is *ground* if it does not contain any variable. A substitution σ is *ground* if $\mathrm{codom}(\sigma)$ is ground. A substitution σ is *grounding* for a term t, literal L, clause C if $t\sigma$, $L\sigma$, $C\sigma$ is ground, respectively. The function mgu denotes the *most general unifier* of two terms, atoms, literals. We assume that any mgu of two terms or literals does not introduce any fresh variables and is idempotent. A *closure* is denoted as $C \cdot \sigma$ and is a pair of a clause C and a substitution σ that is grounding for C. The function ground returns the set of all ground instances of a literal, clause, or clause set with respect to the signature of the respective clause set.

A *(partial) model* M for a clause set N is a satisfiable set of ground literals. A ground clause C is true in M, denoted $M \models C$, if $C \cap M \neq \emptyset$, and false otherwise. A ground clause set N is true in M, denoted $M \models N$ if all clauses from N are true in M. A *(partial) Herbrand model* I for a clause set N is a set of ground atoms. A ground clause C is true in I, denoted $I \models_H C$, if there is an atom $A \in C$ such that $A \in I$, or there is a negative literal $\neg A \in C$ such that $A \notin I$, and false otherwise. A ground clause set N entails a ground clause C, denoted $N \models C$, if $M \models C$ implies $M \models \{C\}$ for all models M.

We identify sets and sequences whenever appropriate. However, the trail of an SCL run is always a sequence of ground literals.

Let $\prec$ denote a well-founded, total, strict ordering on ground literals. This ordering is then lifted to clauses and clause sets by its respective multiset extension. We overload $\prec$ for literals, clauses, clause sets if the meaning is clear from the context. The ordering is lifted to the non-ground case via instantiation: we define $C \prec D$ if for all grounding substitutions σ it holds $C\sigma \prec D\sigma$. We define $\preceq$ as the reflexive closure of $\prec$ and $N^{\preceq C} := \{D \mid D \in N \text{ and } D \preceq C\}$.

Definition 1 (Clause Redundancy). *A ground clause C is* redundant *with respect to a ground clause set N and an order $\prec$ if $N^{\preceq C} \models C$. A clause C is* redundant *with respect to a clause set N and an order $\prec$ if for all $C' \in \text{ground}(C)$ it holds that C' is redundant with respect to* $\text{ground}(N)$.

Let $\prec_B$ denote a well-founded, total, strict ordering on ground atoms such that for any ground atom A there are only finitely many ground atoms B with $B \prec_B A$. For example, an instance of such an ordering could be KBO without zero-weight symbols. (Note that LPO does not satisfy the last condition of a $\prec_B$ ordering although it is a well-founded, total, strict ordering.) The ordering $\prec_B$ is lifted to literals by comparing the respective atoms and if the atoms of two literals are the same, then the negative version of the literal is larger than the positive version. It is lifted to clauses by a multiset extension.

The SCL(FOL) Calculus: The inference rules of SCL(FOL) [9] are represented by an abstract rewrite system. They operate on a problem state, a six-tuple $(\Gamma; N; U; \beta; k; D)$ where Γ is a sequence of annotated ground literals, the *trail*; N and U are the sets of *initial* and *learned* clauses; β is a ground literal limiting the size of the trail; k counts the number of decisions; and D is either $\top$, $\bot$ or a clause closure $C \cdot \sigma$ such that $C\sigma$ is ground and false in Γ. Literals in Γ are either annotated with a number, also called a level; i.e., they have the form L^k meaning that L is the k-th guessed decision literal, or they are annotated with a closure that propagated the literal to become true. A ground literal L is of *level* i with respect to a problem state $(\Gamma; N; U; \beta; k; D)$ if L or $\text{comp}(L)$ occurs in Γ and the first decision literal left from L ($\text{comp}(L)$) in Γ, including L, is annotated with i. If there is no such decision literal then its level is zero. A ground clause D is of *level* i with respect to a problem state $(\Gamma; N; U; \beta; k; D)$ if i is the maximal level of a literal in D. The level of the empty clause $\bot$ is 0. Recall D is a non-empty closure or $\top$ or $\bot$. Similarly, a trail Γ is of level i if the maximal literal in Γ is of level i.

A literal/atom L/A is *undefined* in Γ if neither L/A nor $\text{comp}(L)/\text{comp}(A)$ occur in Γ. The start state of SCL is $(\epsilon; N; \emptyset; \beta; 0; \top)$ for some initial clause set N and bound β. The below rules are exactly the rules from [9] and serve as a reference for our simulation proof in Sect. 3.

Propagate $(\Gamma; N; U; \beta; k; \top) \Rightarrow_{\text{SCL}} (\Gamma, L\sigma^{(C_0 \vee L)\delta \cdot \sigma}; N; U; \beta; k; \top)$
provided $C \vee L \in (N \cup U)$, $C = C_0 \vee C_1$, $C_1\sigma = L\sigma \vee \cdots \vee L\sigma$, $C_0\sigma$ does not contain $L\sigma$, δ is the mgu of the literals in C_1 and L, $(C \vee L)\sigma$ is ground, $(C \vee L)\sigma \prec_\beta \{\beta\}$, $C_0\sigma$ is false under Γ, and $L\sigma$ is undefined in Γ.

Decide $(\Gamma; N; U; \beta; k; \top) \Rightarrow_{\text{SCL}} (\Gamma, L\sigma^{k+1}; N; U; \beta; k+1; \top)$
provided $\text{atom}(L)$ occurs C for a $C \in (N \cup U)$, $L\sigma$ is a ground literal undefined in Γ, and $L\sigma \prec_\beta \beta$.

Conflict $(\Gamma; N; U; \beta; k; \top) \Rightarrow_{\text{SCL}} (\Gamma; N; U; \beta; k; D \cdot \sigma)$
provided $D \in (N \cup U)$, $D\sigma$ false in Γ for a grounding substitution σ.

Skip $(\Gamma, L; N; U; \beta; k; D \cdot \sigma) \Rightarrow_{\text{SCL}} (\Gamma; N; U; \beta; k - i; D \cdot \sigma)$
provided $\text{comp}(L)$ does not occur in $D\sigma$, if L is a decision literal then $i = 1$, otherwise $i = 0$.

Factorize $(\Gamma; N; U; \beta; k; (D \vee L \vee L') \cdot \sigma) \Rightarrow_{\text{SCL}} (\Gamma; N; U; \beta; k; (D \vee L)\eta \cdot \sigma)$
provided $L\sigma = L'\sigma$, $\eta = \text{mgu}(L, L')$.

Resolve $(\Gamma, L\delta^{(C \vee L) \cdot \delta}; N; U; \beta; k; (D \vee L') \cdot \sigma)$
$\Rightarrow_{\text{SCL}}$ $(\Gamma, L\delta^{(C \vee L) \cdot \delta}; N; U; \beta; k; (D \vee C)\eta \cdot \sigma\delta)$
provided $L\delta = \text{comp}(L'\sigma)$, $\eta = \text{mgu}(L, \text{comp}(L'))$.

Backtrack $(\Gamma_0, K, \Gamma_1, \text{comp}(L\sigma)^k; N; U; \beta; k; (D \vee L) \cdot \sigma)$
$\Rightarrow_{\text{SCL}}$ $(\Gamma_0; N; U \cup \{D \vee L\}; \beta; j; \top)$
provided $D\sigma$ is of level $i' < k$, and Γ_0, K is the minimal trail subsequence such that there is a grounding substitution τ with $(D \vee L)\tau$ is false in Γ_0, K but not in Γ_0, and Γ_0 is of level j.

A sequence of rule applications of a particular calculus is called a *run* of the calculus. A *strategy* for a calculus restricts the set of runs we actually allow by imposing further conditions on the allowed rule applications.

Definition 2 (SCL Runs). *A sequence of SCL rule applications is called a* reasonable run *if the rule Decide does not enable an immediate application of rule Conflict. A sequence of SCL rule applications is called a* regular run *if it is a reasonable run and the rule Conflict has precedence over all other rules.*

All regular SCL runs are sound, only derive non-redundant clauses, always terminate, and SCL with a regular strategy is refutationally complete (for first-order logic without equality) [9].

The Superposition Calculus: Superposition [1,2,18] is a calculus for first-order logic reasoning that also infers/learns new clauses like SCL. In contrast to SCL, it does these inferences based on a static ordering $\prec$ and, at the level of inference rules, independent of a partial model. A permissible ordering $\prec$ for the superposition calculus is always a well-founded, total, strict ordering on ground literals. This ordering is then lifted to clauses and clause sets by its respective multiset extension. A problem state in the superposition calculus is just a set N of clauses. The start state the initial clause set. Due to the restriction to first-order logic without equality, the most basic version of the superposition calculus consists just of the following two rules (without selection):

Superposition Left $(N \uplus \{C_1 \vee P(t_1, \ldots, t_n), C_2 \vee \neg P(s_1, \ldots, s_n)\}) \Rightarrow_{\text{SUP}}$
$(N \cup \{C_1 \vee P(t_1, \ldots, t_n), C_2 \vee \neg P(s_1, \ldots, s_n)\} \cup \{(C_1 \vee C_2)\sigma\})$
where (i) $P(t_1, \ldots, t_n)\sigma$ is strictly maximal in $(C_1 \vee P(t_1, \ldots, t_n))\sigma$
(ii) $\neg P(s_1, \ldots, s_n)\sigma$ is maximal, (iii) σ is the mgu of $P(t_1, \ldots, t_n)$ and $P(s_1, \ldots, s_n)$.

Factoring $(N \uplus \{C \vee P(t_1, \ldots, t_n) \vee P(s_1, \ldots, s_n)\}) \Rightarrow_{\text{SUP}}$
$(N \cup \{C \vee P(t_1, \ldots, t_n) \vee P(s_1, \ldots, s_n)\} \cup \{(C \vee P(t_1, \ldots, t_n))\sigma\})$
where (i) $P(t_1, \ldots, t_n)\sigma$ is maximal in $(C \vee P(t_1, \ldots, t_n) \vee P(s_1, \ldots, s_n))\sigma$
(ii) σ is the mgu of $P(t_1, \ldots, t_n)$ and $P(s_1, \ldots, s_n)$.

Let $\mathrm{sfac}(C)$ represent a clause obtained by exhaustively applying superposition Factoring on C. Recall, that superposition Factoring only applies to maximal positive literals. Let $\mathrm{sfac}(N)$ represent the clause set N after every clause has been exhaustively factorized by Superposition Factorization.

Although the superposition calculus itself is independent of a partial model and may learn non-redundant clauses, the completeness proof of superposition in [1] is based on a strategy that builds ground partial models according to the fixed ordering $\prec$, where minimal false ground instances of clauses then trigger non-redundant superposition inferences. Note that the completeness proof relies on a grounding of the clause set that may lead to infinitely many clauses. However, the strategy from the completeness proof can also be seen as a superposition strategy for an initial clause set, where all clauses are already ground. On ground, finite clause sets, superposition restricted to the strategy only infers non-redundant clauses, always terminates, and is complete. The partial model needed in each step of the strategy is constructed according to the following model operator:

Definition 3 (Superposition Model Operator). *Let N be a set of ground clauses. Then N_I is the Herbrand model according to the superposition model operator for clause set N and it is constructed recursively over the partial Herbrand models N_C for all $C \in N$:*

$$N_C = \bigcup_{D \prec C} \delta_D \qquad N_I = \bigcup_{C \in N} \delta_C$$

$$\delta_D = \begin{cases} \{B\} & \text{if } D = D' \vee B,\ B \text{ strictly maximal},\ N_D \not\models_H D \\ \emptyset & \text{otherwise} \end{cases}$$

We say that a clause C is productive *(wrt. the model construction of a clause set N) if $\delta_C \neq \emptyset$. We say that a clause C* produces *an atom B (wrt. the model construction of a clause set N) if $\delta_C = \{B\}$.*

After constructing the model N_I for a clause set N, the strategy selects the smallest clause in N that is false in N_I. The strategy then selects a fitting inference rule based on the reason why the clause is false in N_I. The newly inferred clause either changes the model in the next step or changes the smallest clause that is false. This is the strategy used in the superposition completeness proof [1].

Definition 4 (Minimal False Clause). *The minimal false clause $C \in N$ is the smallest clause in N according to $\prec$ such that $N_C \cup \delta_C \not\models_H C$.*

Definition 5 (Superposition Model-Operator Strategy: SUP-MO). *The* superposition model-operator strategy *is defined over the minimal false clause with regards to the current clause set N. The strategy can encounter the following cases:*

(1) *N has no minimal false clause. Then N is satisfied by N_I and we can stop the superposition run.*

(2) *The minimal false clause in N is $\bot$. Then N is unsatisfiable, which means we can also stop the superposition run.*

(3) *C is the minimal false clause in N, and it has a maximal literal L that is negative. Then there must be a clause $D \in N$ with $D \prec C$, a strictly maximal literal* $\mathrm{comp}(L)$*, and $\delta_D = \{\mathrm{comp}(L)\}$. In this case, the strategy applies as its next step Superposition Left to C and D.*

(4) *C is the minimal false clause in N, and it has a maximal literal L that is positive. Then L is not strictly maximal in C and the strategy applies Factoring to C.*

The first two cases of the SUP-MO strategy also describe its final states according to [1]. In all other states there is always exactly one rule applicable according to the SUP-MO strategy, which also means that SUP-MO is never stuck.

Lemma 6 (SUP-MO Applicability). *Let N be a set of ground clauses. If N has a minimal false clause $C \neq \bot$, then there exists exactly one rule applicable to N according to the SUP-MO strategy.*

3 SCL Simulates Superposition

In general, it is not possible to simulate all inferences of the superposition calculus with SCL because SCL only learns/infers non-redundant clauses, whereas syntactic superposition inferences have no such guarantees. Moreover, the inferences by SCL are all based on conflicts according to a partial model driven by the satisfiability of clause instances, whereas the inferences by superposition are based on a static ordering $\prec$. We can mitigate these differences by restricting superposition with the SUP-MO strategy because SUP-MO has non-redundancy guarantees and it infers new clauses based on minimal false clauses with respect to a ground partial model.

Let N^0 be a set of ground clauses, totally ordered by a superposition reduction ordering $\prec$. Let N^i (for $i > 0$) be the result of i steps of the superposition calculus applied to N^0 according to the SUP-MO strategy, i.e., $N^0 \Rightarrow_{\text{SUP-MO}} N^1 \Rightarrow_{\text{SUP-MO}} \dots \Rightarrow_{\text{SUP-MO}} N^i$. Again, all N^i are sets of ground clauses, totally ordered by a superposition reduction ordering $\prec$. The SCL strategy SCL-SUP that simulates superposition restricted to SUP-MO runs is defined inductively on the clause ordering $\prec$. To guide and to prove the correctness of our simulation, we assign to each SCL state and every clause some additional information. For this purpose, every SCL state is annotated with a triple (i, C, γ), where i is an integer that states that the SCL state simulates the superposition state N^i, C is the last clause that was used as a decision aid by the strategy, γ is a function such that $\gamma(C) = \mathrm{sfac}(C)$ if $\mathrm{sfac}(C) \in N^i$ and $\gamma(C) = C$ otherwise, the SCL state also simulates the model construction for N^i upto $N^i_{C'} \cup \delta_{C'}$, where $C' = \gamma(C)$. The annotated states are written $(\Gamma; N^0; U; \beta; k; E)_{(i,C,\gamma)}$. The overall start state is then $(\epsilon; N^0; \emptyset; \beta; 0; \top)_{(0,\bot,\gamma)}$, where we assume β large enough so $A \prec_\beta \beta$ for all $A \in \mathrm{atom}(N^0)$, $\bot \notin N^0$, and $\gamma(C) = \mathrm{sfac}(C)$ if $\mathrm{sfac}(C) \in N^0$ and $\gamma(C) = C$ otherwise. We will later see that the annotated integer is not relevant for the actual choice of SCL rules by the SCL-SUP strategy but only

to prove that the strategy actually simulates superposition. Moreover, we define a new ordering $\prec_\gamma$ based on our superposition ordering $\prec$ and function γ such that $C \prec_\gamma D$ if $\gamma(C) \prec \gamma(D)$.

Definition 7 (State Simulation). *Let $(\Gamma; N^0; U; \beta; k; E)_{(i,D,\gamma)}$ be an SCL state for the input clauses N^0. Let L be the maximal literal in D if $D \neq \bot$ and the minimal literal according to $\prec$ otherwise. Let $N^0 \Rightarrow_{SUP\text{-}MO} N^1 \Rightarrow_{SUP\text{-}MO} \ldots \Rightarrow_{SUP\text{-}MO} N^i$ be the superposition run following the SUP-MO strategy starting from the input clause set N^0. Let $D' = \gamma(D)$. Then we say that the SCL state $(\Gamma; N^0; U; \beta; k; E)_{(i,D,\gamma)}$* simulates *$N^i$ and the model construction upto $N^i_{D'} \cup \delta_{D'}$ if*

- **(i)** $\mathrm{atom}(N^0) = \mathrm{atom}(N^i) = \mathrm{atom}(N^0 \cup U)$, *$A \prec_\beta \beta$ for all $A \in \mathrm{atom}(N^0)$, and $D \in \{\bot\} \cup N^0 \cup U$*
- **(ii)** $\mathrm{sfac}(N^0 \cup U) \subseteq \mathrm{sfac}(N^i)$ *and $\gamma(C) \in N^i$ for all $C \in N^0 \cup U$ and $\gamma(C) = \mathrm{sfac}(C)$ or $\gamma(C) = C$.*
- **(iii)** *for all $C \in N^i$ there exists a $C' \in N^0 \cup U \cup \{E\}$ such that $\mathrm{sfac}(C) = \mathrm{sfac}(C')$ if the maximal literal in C is positive*
- **(iv)** *for all $C \in N^i$ there exists a $C' \in N^0 \cup U \cup \{E\}$ such that $C' \models C$ and $\gamma(C') \preceq C$*
- **(v)** *for all atoms A occurring in N^0: $A \in N_{D'} \cup \delta_{D'}$ iff $A \in \Gamma$*
- **(vi)** *for all atoms A: $\neg A \in \Gamma$ iff $A \prec L$ and $A \notin N_{D'}$*
- **(vii)** *for every literal L in Γ, i.e., $\Gamma = \Gamma', L, \Gamma''$, and all literals L' in Γ',* $\mathrm{atom}(L') \prec \mathrm{atom}(L)$
- **(viii)** *for every atom (= positive literal) B in Γ, i.e., $\Gamma = \Gamma', B, \Gamma''$, there exists $C \in N^0 \cup U$ and a $C' \in N^i$ such that $\gamma(C) = \mathrm{sfac}(C) = \mathrm{sfac}(C') = C'$, and C' produces B, i.e., $\delta_{C'} = \{B\}$*
- **(ix)** *for every clause $C \in N^i$ with $C \preceq \gamma(D)$ that produces an atom B, i.e., $\delta_C = \{B\}$, there exists $C' \in N^0 \cup U$ such that $C = \gamma(C')$ and $C \preceq_\gamma D$.*
- **(x)** *Γ contains only decisions if $E = \top$*
- **(xi)** *$E \notin \{\top, \bot\}$ iff $\Gamma = \Gamma' B^{\mathrm{sfac}(D)}$, Γ' contains only decisions, there exists $E' \in N^i$ where $\gamma(E) = E = E'$ is the minimal false clause in N^i, and $\neg B \in E$*
- **(xii)** *$\Gamma \models C$ for all $C \in N^0 \cup U$ with $C \preceq_\gamma D$*
- **(xiii)** *Conflict is not applicable to $(\Gamma; N^0; U; \beta; k; E)_{(i,D,\gamma)}$.*
- **(xiv)** *$\bot \notin N^0 \cup U$ and $E = \bot$ iff $\Gamma = \epsilon$ and $\bot \in N^i$*

The above invariants can be summarized as follows: (i) All ground atoms encountered are known from the start and the trail bound β is large enough so SCL can Decide/Propagate them. (ii)–(iv) Every initial clause C or inferred clause by SUP-MO must coincide with an initial clause C' or learned clause by SCL; this means on the one hand that for every clause C learned by SCL-SUP, SUP-MO infers a clause C' that is identical up to factoring; on the other hand it means that for every clause C inferred by SUP-MO, SCL-SUP learns a clause C' that entails C (i.e. $C' \models C$) and is at most as large as C wrt. γ. (v)–(ix) The partial model constructed by SCL-SUP and SUP-MO coincide and any atom B in $N_C \cup \delta_C$ produced by clause D has a clause D' on the SCL side that could

propagate B and vice versa. (x)–(xiii) Ensure that any Conflict in SCL-SUP corresponds to a minimal false clause and that the trail is always constructed in such a way that the Resolve applications per Conflict call are limited to the maximal literal in the conflict; this property is needed or the next clause that would be learned by SCL no longer coincides with the clauses learned by SUP-MO. (xiv) Describes the final state in case the input clause set is unsatisfiable.

Now that we have defined how an SCL state must look like in order to simulate a superposition state, we define SCL-SUP, the SCL strategy that eventually simulates a SUP-MO run. First, note that not all states visited by SCL-SUP satisfy the invariants of Definition 7. However, the invariants hold again after each so-called *atomic sequence* of SCL-SUP steps. Second, one atomic sequence of SCL-SUP steps may skip over several successive superposition states. The reason is that SCL can and must skip all steps of SUP-MO that occur because the maximal literal in a clause is not strictly maximal, i.e., superposition Factoring steps. SCL performs factoring implicitly in its Propagation rule so SCL never has to explicitly simulate case (4) of Definition 5. Third, definition of the SCL-SUP strategy is split in two parts and each part describes some atomic sequences of SCL-SUP steps.

Definition 8 (SCL Superposition Strategy: SCL-SUP Part 1). *Let $S_0 = (\Gamma; N^0; U; \beta; k; \top)_{(i,C,\gamma)}$ be an SCL state with additional annotations for the strategy. Let D be the next largest clause from C in the ordering $\prec_\gamma$ with respect to the ground clause set $N^0 \cup U$. Let L be the maximal literal of D. Let $[\neg A_1, \neg A_2, \dots \neg A_n]$ be all negative literals such that for all i we have $A_i \prec L$, all A_i undefined in Γ, A_i occurs in $N^0 \cup U$, and $A_i \prec A_{i+1}$. Let $D' = \gamma(D)$ be in N^i such that $\mathrm{sfac}(D) = \mathrm{sfac}(D')$. Let $j_0 + 1$ be the number of occurrences of L in D' and $j = i + j_0$. Then the* SCL Superposition Strategy *(SCL-SUP) performs the following steps to S_0 (possibly without any actual SCL rule applications, just changing the state annotation):*

- **(1)** *First decide all literals $[\neg A_1, \neg A_2, \dots \neg A_n]$ in order, i.e., $S_0 \Rightarrow^{*}_{\textit{SCL-SUP}}{}^{\textit{Decide}} S_1$, where $S_1 = (\Gamma, \neg A_1^{k+1}, \dots, \neg A_n^{k+n}; N^0; U; \beta; k+n; \top)_{(i,D,\gamma)}$.*
- **(2a)** *If the maximal literal L in D is positive (i.e., $L = B$), $\Gamma, \neg A_1^{k+1}, \dots, \neg A_n^{k+n} \not\models D$, and Conflict is not applicable to $S_2 = (\Gamma, \neg A_1^{k+1}, \dots, \neg A_n^{k+n}, B^{k+n+1}; N^0; U; \beta; k+n+1; \top)_{(j,D,\gamma')}$, then decide B, i.e., $S_1 \Rightarrow^{\textit{Decide}}_{\textit{SCL-SUP}} S_2$, where γ' is the same as γ except that $\gamma'(D) = \mathrm{sfac}(D)$.*
- **(2b)** *If the maximal literal L in D is positive (i.e., $L = B$), $\Gamma, \neg A_1^{k+1}, \dots, \neg A_n^{k+n} \not\models D$, and E is the smallest clause in $N^0 \cup U$ that is false in wrt. $\Gamma, \neg A_1^{k+1}, \dots, \neg A_n^{k+n}, B^{\mathrm{sfac}(D)}$, then propagate B and apply Conflict to E, i.e., $S_1 \Rightarrow^{\textit{Propagate}}_{\textit{SCL-SUP}} S_2' \Rightarrow^{\textit{Conflict}}_{\textit{SCL-SUP}} S_2$, where $S_2 = (\Gamma, \neg A_1^{k+1}, \dots, \neg A_n^{k+n}, B^{\mathrm{sfac}(D)}; N^0; U; \beta; k+n; E)_{(j,D,\gamma')}$ and γ' is the same as γ except that $\gamma'(D) = \mathrm{sfac}(D)$.*
- **(2c)** *Otherwise, $S_2 = S_1$ and no further rules have to be applied.*

A (potentially empty) sequence of SCL rule applications according to SCL-SUP is called an atomic sequence *of SCL-SUP steps if it starts from a state S_0 and ends in a state S_2 outlined in the cases (2a-c).*

The first part of the strategy simulates the recursive construction of the partial model used in the SUP-MO strategy (see Definition 3). It assumes that the model is already constructed up to the current annotated clause C and extends this model for the next largest clause $D \in (N^0 \cup U)$. To this end, it uses the rule Decide in step (1) to set all atoms A to false that are still undefined but can no longer be produced by any clause greater or equal to D. Next the strategy makes a case distinction. Step (2a) handles the case where D corresponds to a clause D' in the superposition state (modulo some Factoring steps skipped by SCL) that produces atom B; SCL-SUP then adds B to the trail with the rule Decide because producing/adding this atom does not falsify a clause. Step (2b) handles a similar case compared to step (2a); but in this case producing/adding the atom B to the trail results in a minimal false clause E; in order to force a resolution step between clause D and E, SCL-SUP first uses Propagate to add B to the trail and then applies conflict to E. Step (2c) handles the case where D corresponds to a clause D' that will not produce an atom B even modulo some Factoring steps; in this case no further SCL rule applications are necessary as the SUP-MO model will not change. Note that the annotated function γ is needed so the SCL state knows when the superposition state would have applied Factoring to a clause C, which also means that it is now treated as its factorized version $\gamma(C) = \mathrm{sfac}(C)$ in our inductive clause ordering.

Example 9. Let us now further demonstrate the three different cases of the first part of the SCL-SUP strategy with the help of an example. Let N^0 be our initial set of clauses:

$$N^0 = \{(C_1)\ P(a), \quad (C_2)\ \neg P(b) \vee Q(a), \quad (C_3)\ \neg P(a) \vee Q(a) \vee Q(a),$$
$$(C_4)\ P(a) \vee \neg Q(a), \quad (C_5)\ \neg P(a) \vee \neg Q(a)\}$$

We compare the run of SCL-SUP for N^0 with the run of SUP-MO for N^0 to demonstrate that both runs coincide. As superposition ordering, we choose an LPO with precedence $a \prec b \prec P \prec Q$. This means that the atoms are ordered $P(a) \prec P(b) \prec Q(a) \prec Q(b)$ and the clauses in N^0 are ordered $C_1 \prec C_2 \prec C_3 \prec C_4 \prec C_5$. The initial SUP-MO state is simply the clause set N^0 and the initial SCL-SUP state is $(\epsilon, N^0, \emptyset, \beta, 0, \top)_{(0,\bot,\gamma_0)}$, where $\gamma_0(C) = C$ for all clauses C. In the first step of SCL-SUP, SCL-SUP first selects the clause C_1 as its new decision aid because it is the next largest clause in N^0 compared to $\bot$. Then SCL-SUP continues with step (1) of Definition 3. In this step SCL-SUP does nothing because there are no atoms smaller than $P(a)$. Next, SCL-SUP detects that the maximal literal of C_1 is positive, $\epsilon \not\models C_1$, and that the trail $[P(a)^1]$ does not result in a conflict. Therefore, SCL-SUP follows step (2a) of Definition 3 and Decides $P(a)$, which results in the state $([P(a)^1], N^0, \emptyset, \beta, 1, \top)_{(0,C_1,\gamma_0)}$. Meanwhile, SUP-MO starts with constructing a model for N^0 starting with the clause C_1. The result is that C_1 is productive and $\delta_{C_1} = \{P(a)\}$ and $N^0_{C_1} = \emptyset$, which coincides with our new SCL trail.

SCL-SUP considers the clause C_2 as its new decision aid and continues with step (1) of Definition 3. This time there is an atom smaller than the maximal literal of C_2 namely $P(b)$. Therefore, SCL-SUP Decides $\neg P(b)$ in step (1) of Definition 3, which results in $([P(a)^1, \neg P(b)^2], N^0, \emptyset, \beta, 2, \top)_{(0,C_2,\gamma_0)}$. Next, SCL-SUP detects that the maximal literal of C_2 is positive but that $[P(a)^1, \neg P(b)^2] \models C_2$. Therefore, SCL-SUP follows step (2c) of Definition 3 and ends this atomic sequence immediately. SUP-MO continues the model construction for N^0 with the clause C_2. The clause C_2 is not productive because $N^0_{C_2} \models_H C_2$, where $N^0_{C_2} = \delta_{C_1} = \{P(a)\}$ and $\delta_{C_2} = \emptyset$, which again coincides with our new SCL trail as Herbrand models do not explicitly define atoms assigned to false.

SCL-SUP now considers the clause C_3 as its new decision aid and continues with step (1) of Definition 3. In this step SCL-SUP does nothing because all atoms smaller than $Q(a)$ are already assigned. Next, SCL-SUP detects that the maximal literal of C_3 is positive, $[P(a)^1, \neg P(b)^2] \not\models C_3$, and that the clause C_5 is false with respect to the trail $[P(a)^1, \neg P(b)^2, Q(a)^{\text{sfac}(C_3)}]$. Therefore, SCL-SUP follows step (2b) of Definition 3, i.e. it Propagates $P(a)$ and applies Conflict to C_5, resulting in $([P(a)^1, \neg P(b)^2, Q(a)^{\text{sfac}(C_3)}], N^0, \emptyset, \beta, 2, C_3)_{(1,C_2,\gamma_1)}$, where γ_1 is identical to γ_0 except that $\gamma_1(C_3) = \text{sfac}(C_3) = \neg P(a) \vee Q(a)$. Note that SCL-SUP must change the state annotations because the maximal literal in C_3 is not strictly maximal, so SCL-SUP skips and eventually silently performs the Factorization step performed by SUP-MO. Note also that in the changed clause ordering $\prec_{\gamma_1}$ the order of C_2 and C_3 changed, i.e., $C_3 \prec_{\gamma_1} C_2$, which corresponds to $\text{sfac}(C_3) \prec C_2$. Meanwhile, SUP-MO continues the model construction for N^0 with the clause C_3. The clause C_3 is not productive because the maximal literal is not strictly maximal so $\delta_{(3)} = \emptyset$ and $N^0_{C_3} \cup \delta_{C_3} \not\models_H C_3$ so C_3 is the minimal false clause in N^0. SUP-MO resolves this conflict by applying Factoring to C_3, which means SUP-MO infers the clause $C_6 = \text{sfac}(C_3) = \neg P(a) \vee Q(a)$. The new clause order in superposition state $N^1 = N^0 \cup \{C_6\}$ is $C_1 \prec C_6 \prec C_2 \prec C_3 \prec C_4 \prec C_5$, which matches the changed ordering $C_3 \prec_{\gamma_1} C_2$ because $C_6 = \gamma_1(C_3)$. Next, SUP-MO updates its model construction for N^1. The result is that C_1 and C_6 are productive and that $N^1_{C_6} \cup \delta_{C_6} = \{P(a), Q(a)\}$, which matches the current SCL trail. Moreover, if we continue the model construction upto C_5 then no new literals are produced and C_5 also turns into the minimal false clause for N^1.

Definition 10 (SCL Superposition Strategy: SCL-SUP Part 2). *Let $S_0 = (\Gamma, B^{\text{sfac}(C)}; N^0; U; \beta; k; E)_{(i,C,\gamma)}$ be an SCL state with $E \notin \{\top, \bot\}$ and additional annotations for the strategy. Let $L = \neg B$ be the maximal literal of E. Let Γ contain only decision literals. Let all atoms A occurring in $N^0 \cup U$ with $A \prec B$ be defined in Γ following the order $\prec$, i.e., for all A occurring in $N^0 \cup U$ with $A \prec B$ there exist Γ' and Γ'' such that $\Gamma = \Gamma', L_A, \Gamma''$, $L_A = A$ or $L_A = \neg A$ and all atoms $A' \in N^0 \cup U$ with $A' \prec A$ are defined in Γ'. Let E be contained in N^i. Let j_0 be the number of occurrences of L in E and $j = i + j_0$. Let* sfac$(C) = C_1 \vee B$ *and $E = E' \vee E''$, where E'' contains all occurrences of L in E. Then the* SCL Superposition Strategy *(SCL-SUP) performs the following steps to S_0:*

(1) *First apply Resolve to E until all occurrences of L are resolved away, i.e., $S_0 \Rightarrow^{*\ Resolve}_{SCL\text{-}SUP} S_1$, where $S_2 = (\Gamma, B^{\mathrm{sfac}(C)}; N^0; U; \beta; k; E_2)_{(j,C,\gamma)}$ and $E_2 = E' \vee C_1 \vee \ldots \vee C_1$.*

(2a) *If $E_2 = \bot$, then we apply Skip until the trail is empty and then stop the SCL run, i.e., $S_2 \Rightarrow^{*\ Skip}_{SCL\text{-}SUP} S_5$, where $S_5 = (\epsilon; N^0; U; \beta; 0; \bot)_{(j,\bot,\gamma)}$.*

(2b) *If $E_2 \neq \bot$, then E_2 has a maximal literal L_1. Next the strategy applies Skip until $\mathrm{comp}(L_1)$ is the topmost literal on the trail, i.e., $S_2 \Rightarrow^{*\ Skip}_{SCL\text{-}SUP} S_3$, where $S_3 = (\Gamma_0, L_1^{k_1}; N^0; U; \beta; k_1; E_2)_{(j,C,\gamma)}$. (Note that this step skips at least over the literal $B^{\mathrm{sfac}(C)}$).*

(3) *Next apply Backtrack to S_3, i.e., $S_3 \Rightarrow^{Backtrack}_{SCL\text{-}SUP} S_4$, where $S_4 = (\Gamma_0; N^0; U \cup \{E_2\}; \beta; k_1 - 1; \top)_{(j,C,\gamma)}$.*

(4a) *If L_1 is a negative literal, continue with the following rule applications. Let D be the smallest clause in $N^0 \cup U$ with maximum literal $\mathrm{comp}(L_1) = B_1$ and $\Gamma_0 \not\models D$. Then Propagate B_1 from D, and apply Conflict to E_2, i.e., $S_4 \Rightarrow^{Propagate}_{SCL\text{-}SUP} S_4'' \Rightarrow^{*\ Conflict}_{SCL\text{-}SUP} S_5$, where $S_5 = (\Gamma_0, B_1^{\mathrm{sfac}(D)}; N^0; U \cup \{E_2\}; \beta; k_1 - 1; E_2)_{(j,D,\gamma)}$.*

(4b) *If L_1 is a positive literal (i.e., $L_1 = B$) and Conflict is not applicable to $S_5 = (\Gamma_0, B_1^{k_1}; N^0; U \cup \{E_2\}; \beta; k_1; \top)_{(j_2,E_2,\gamma')}$, then decide B, i.e., $S_4 \Rightarrow^{Decide}_{SCL\text{-}SUP} S_5$, where $j_1 + 1$ is the number of occurrences of B_1 in E_2, $j_2 = j + j_1$, and γ' is the same as γ except that $\gamma'(E_2) = \mathrm{sfac}(E_2)$.*

(4c) *If L_1 is a positive literal (i.e., $L_1 = B$) and E_3 is the smallest clause in $N^0 \cup U$ that is false in $S_5' = (\Gamma_0, B_1^{\mathrm{sfac}(E_2)}; N^0; U; \beta; k_1 - 1; \top)_{(j,E_2)}$, then propagate B_1 and apply Conflict to E_3, i.e., $S_4 \Rightarrow^{Propagate}_{SCL\text{-}SUP} S_5' \Rightarrow^{Conflict}_{SCL\text{-}SUP} S_5$, where $S_5 = (\Gamma_0, B_1^{\mathrm{sfac}(E_2)}; N^0; U; \beta; k_1 - 1; E_3)_{(j_2,E_2,\gamma')}$, $j_1 + 1$ is the number of occurrences of B_1 in E_2, $j_2 = j + j_1$, and γ' is the same as γ except that $\gamma'(E_2) = \mathrm{sfac}(E_2)$.*

A (potentially empty) sequence of SCL rule applications according to SCL-SUP is called an atomic sequence *of SCL-SUP steps if it starts from a state S_0 and ends in a state S_5 outlined in the cases (2a) and (5a-c).*

The second part of the strategy simulates the actual inferences resulting from a minimal false clause found in step (2b) of Definition 8 or found in steps (4a) and (4c) of Definition 10. These inferences always correspond to Superposition Left steps of the SUP-MO strategy that resolve minimal false clauses E' in N^i with maximal literal $\neg B$ with the clause C' in N^i that produced B. Note however that SCL-SUP may combine several Superposition Left steps of the SUP-MO strategy into one new learned clause. This is the case whenever the maximal literal $\neg B$ in the minimal false clause E' in N^i is not strictly maximal. In this case, the next minimal false clause E'' will always correspond to the last inferred clause, the maximal literal of this clause will still be $\neg B$, the clause producing B will be again C', and therefore the next Superposition Left partner of E'' is also again C'. Moreover, all of the skipped inferences are actually redundant with respect to the final inference E_2' in this chain, which explains why SCL-SUP is still capable of simulating SUP-MO although it skips the intermediate inferences. The actual

SCL-SUP clause E_2 corresponding to final SUP-MO inference E_2' is computed in the steps (1) and (2) of Definition 10 with greedy applications of the rules Resolve and Factorize. The following steps of Definition 10 take care of the four different cases how E_2' changes the model and minimal false clause in N^j. The first case is that $E_2' = \bot$ so SUP-MO has reached a final state. This case is handled by step (2a) of Definition 10 that simply empties the trail with applications of the rule Skip so the resulting SCL state has the form of a SCL-SUP final state. The second case is that the maximal literal L_1 in E_2' is negative. In this case, the model for N^i and N^j is still the same and just the minimal false clause changes to E_2'. This case is handled by steps (2b)–(4a) of Definition 10 that Backtrack before comp(L_1) was decided, propagate it instead and apply Conflict to E_2. In the third and fourth case the maximal literal L_1 in E_2' is positive. In this case, the model for N^i and N^j actually changes because E_2' is always productive. Case (2b)–(4b) of Definition 10 handles the case where producing L_1 leads to no new minimal false clause, and case (2b)–(4c) of Definition 10 handles the case where it does. Both cases work symmetrically to steps (2a) and (2b) of Definition 8.

Example 11. We continue Example 9 to demonstrate cases (1)→(4a) and (1)→(2a) of the second part of the SCL-SUP strategy. We left the runs in the SCL state $([P(a)^1, \neg P(b)^2, Q(a)^{\mathrm{sfac}(C_3)}], N^0, \emptyset, \beta, 2, C_3)_{(1,C_2,\gamma_1)}$ that simulates the superposition state N^1, where

$$N^1 = \{(C_1) P(a), \quad (C_2)\ \neg P(b) \vee Q(a), \quad (C_3)\ \neg P(a) \vee Q(a) \vee Q(a),$$
$$(C_4)\ P(a) \vee \neg Q(a), \quad (C_5)\ \neg P(a) \vee \neg Q(a), \quad (C_6)\ \neg P(a) \vee Q(a)\}$$

and C_5 became the minimal false clause in N^1 after C_1 and C_6 produced together the partial model $\{P(a), Q(a)\}$. SUP-MO continues from the state N^1 by applying Superposition Left to C_5 and C_6. In the new state $N^2 = N^1 \cup \{(C_7)\ \neg P(a) \vee \neg P(a)\}$ the new clause order is $C_1 \prec C_7 \prec C_6 \prec C_2 \prec C_3 \prec C_4 \prec C_5$ and after constructing the model for C_1, which produces again $P(a)$, the clause C_7 becomes again the minimal false clause. SCL-SUP follows (1) of Definition 10 and applies Resolve to C_5 and $\mathrm{sfac}(C_3) = C_6$, resulting in the state $([P(a)^1, \neg P(b)^2, Q(a)^{\mathrm{sfac}(C_3)}], N^0, \emptyset, \beta, 2, C_7)_{(2,C_2,\gamma_1)}$. Then SCL-SUP continues with steps (2b) and (3) by applying Skip twice and Backtrack once to jump to the state $(\epsilon, N^0, \{C_7\}, \beta, 0, \top)_{(2,C_2,\gamma_1)}$. Next, SCL-SUP continues with step (4a) because the maximal literal of C_7 is $\neg P(a)$ and therefore negative. This means SCL-SUP will add $P(a)$ again to the trail but this time by applying Propagate to C_1 and afterwards it applies Conflict to C_7. The resulting state $([P(a)^{\mathrm{sfac}(C_1)}], N^0, \{C_7\}, \beta, 0, C_7)_{(2,C_1,\gamma_1)}$ matches again the SUP-MO state N^2.

SUP-MO continues from the state N^2 by applying Superposition Left to C_7 and C_1, resulting in $N^3 = N^2 \cup \{(C_8) \neg P(a)\}$. Since C_8 has the same maximal literal as C_7 it becomes automatically the next minimal false clause in N^3. As a result, SUP-MO applies Superposition Left to C_8 and C_1, which returns $N^5 = N^3 \cup \{(C_9)\ \bot\}$ a final state that proves the unsatisfiability of N^0. Meanwhile, SCL-SUP simulates both Superposition Left steps with one atomic SCL-SUP sequence. It starts with step (1) of Definition 10 and applies Resolve twice, resulting in the state $([P(a)^1, \neg P(b)^2, Q(a)^{\mathrm{sfac}(C_3)}], N^0, \emptyset, \beta, 2, \bot)_{(4,C_2,\gamma_1)}$. Then

it continues with step (2a) of Definition 10 and applies Skip until the trail is empty. The resulting state $(\epsilon, N^0, \emptyset, \beta, 2, \bot)_{(4,\bot,\gamma_1)}$ is a final state and proves unsatisfiability of N^0.

Example 12. The next example demonstrates the atomic sequence (1)→(4b) of the second part of the SCL-SUP strategy. Let N^0 be our initial set of clauses:

$$N^0 = \{(C_1)P(a), \quad (C_2)\ \neg P(b), \quad (C_3)\ \neg P(a) \vee Q(a), \quad (C_4)\ P(b) \vee \neg Q(a)$$

As superposition ordering, we choose an LPO with precedence $a \prec b \prec P \prec Q$. This means that the atoms are ordered $P(a) \prec P(b) \prec Q(a) \prec Q(b)$ and the clauses in N^0 are ordered $C_1 \prec C_2 \prec C_3 \prec C_4$. In order to keep the example short, we skip the initial SCL-SUP steps and continue directly with the state $S = ([P(a)^1, \neg P(b)^2, Q(a)^{\mathrm{sfac}(C_3)}], N^0, \emptyset, \beta, 2, C_4)_{(0,C_3,\gamma)}$, where $\gamma(C) = C$ for all clauses C and $\beta = Q(b)$. This state simulates the superposition state N^0 upto the model construction for C_3, where $N^0_{C_3} \cup \delta_{C_3} = \delta_{C_1} \cup \delta_{C_3} = \{P(a), Q(a)\}$ and C_4 is the minimal false clause. SUP-MO continues from the state N^0 by applying Superposition Left to C_4 and C_3. In the new state $N^1 = N^0 \cup \{(C_5)\ \neg P(a) \vee P(b)\}$ the new clause order is $C_1 \prec C_5 \prec C_2 \prec C_3 \prec C_4$ and the partial model upto C_5 is $N^0_{C_5} \cup \delta_{C_5} = \delta_{C_1} \cup \delta_{C_5} = \{P(a)\} \cup \{P(b)\}$, which turns C_2 into the next minimal false clause. SCL-SUP simulates the above steps by following the atomic sequence (1)→(4b) of Definition 10. The result is the state $([P(a)^1, P(b)^{\mathrm{sfac}(C_5)}], N^0, \{C_5\}, \beta, 1, C_2)_{(1,C_5,\gamma)}$ matching again our current superposition state and model.

Without clause C_2, SCL-SUP would apply the atomic sequence (1)→(4a) of Definition 10 to S, resulting in the state $([P(a)^1, P(b)^2], N^0 \setminus \{C_2\}, \{C_5\}, \beta, 2, \top)_{(1,C_5,\gamma)}$. This matches the state $N^1 \setminus \{C_2\}$ and its partial model upto C_5 that is still the same as for N^1 with the exception that it does not lead to a minimal false clause.

In order to actually show that every SCL-SUP run simulates a SUP-MO run, we need to prove three properties. The first property is that each state visited by an SCL-SUP run must simulate a state visited by the corresponding SUP-MO run. Note that this property does not yet say anything about the order in which SCL-SUP simulates the SUP-MO states. This property can also be seen as a soundness argument for our strategy.

Lemma 13 (Initial SCL State Simulates Initial Superposition State). *The initial SCL state $(\epsilon; N^0; \emptyset; \beta; 0; \top)_{(0,\bot,\gamma)}$ simulates the initial superposition state N^0 and the model construction upto $N^0_\bot \cup \delta_\bot$*

Lemma 14 (SCL-SUP Preserves Simulation). *Let the SCL state $S = (\Gamma; N^0; U; \beta; k; E)_{(i,C,\gamma)}$ simulate the superposition state N^i and the corresponding model construction upto $N^i_{C'} \cup \delta_{C'}$, where $C' = \gamma(C)$. Let the SCL state $S' = (\Gamma'; N^0; U'; \beta; k'; E')_{(j,D,\gamma')}$ be the result of one atomic sequence of SCL-SUP steps. Then there exists a clause $D' \in N^j$ with $\gamma'(D) = D'$ and S' simulates the superposition state N^j and the model construction upto $N^j_{D'} \cup \delta_{D'}$.*

The second property is that each atomic sequence of SCL-SUP steps always makes progress in the simulation. This means that each atomic sequence of SCL-SUP steps either advances the superposition state N^i simulated by the current SCL state $S = (\Gamma; N^0; U; \beta; k; E)_{(i,D,\gamma)}$, i.e., it increases the annotated i, or it still simulates the same superposition state N^i but advances the simulation of the model construction operator, i.e. it increases the annotated clause C and keeps i the same. Note that it can actually happen that an atomic sequence of SCL-SUP steps skips over several superposition states. This property can also be seen as a termination argument for our strategy because SUP-MO always terminates on ground clause sets.

Lemma 15 (SCL-SUP Advances the Simulation). *Let the SCL state $S = (\Gamma; N^0; U; \beta; k; E)_{(i,D,\gamma)}$ simulate the superposition state N^i and the model construction upto $N^i_D \cup \delta_D$. Let the SCL state $S' = (\Gamma'; N^0; U'; \beta; k'; E')_{(j,D',\gamma')}$ be the next state reachable by one atomic sequence of SCL-SUP steps. Then either $i < j$ or $i = j$ and $\gamma' = \gamma$ and $D \prec_\gamma D'$.*

The last missing property shows that the SCL-SUP strategy can always advance the current SCL state whenever the simulated superposition state can be advanced by the SUP-MO strategy. This means SCL-SUP is never stuck when SUP-MO can still progress. These properties hold because the simulation invariants in Definition 7 either correspond to a correct final state or they satisfy the preconditions of Definition 8 or Definition 10. This property can also be seen as a partial correctness argument for our strategy.

Lemma 16 (SCL-SUP Correctness of Final States). *Let the SCL state $S = (\Gamma; N^0; U; \beta; k; E)_{(i,D,\gamma)}$ simulate the superposition state N^i and the model construction upto $N^i_{\gamma(D)} \cup \delta_{\gamma(D)}$. Let there be no more states reachable from S following an atomic sequence of SCL-SUP steps. Then S is a* final state*, i.e., either (i) $E = \bot$, $D = \bot$, $\bot \in N^i$, and N^0 is unsatisfiable or (ii) $\Gamma \models N^0$.*

We can also show that any SCL-SUP run is also a regular run. Although this is not strictly necessary for the simulation proof, it is beneficial because it means that SCL-SUP inherits many properties that hold for SCL restricted to a regular strategy. For instance, that all learned clauses are non-redundant and that SCL-SUP always terminates.

Lemma 17 (SCL-SUP is a Regular SCL Strategy). *SCL-SUP is a regular SCL strategy if it is executed on a state $S = (\Gamma; N^0; U; \beta; k; E)_{(i,C,\gamma)}$ that simulates a superposition state N^i and the corresponding model construction upto $N^i_{\gamma(C)} \cup \delta_{\gamma(C)}$.*

4 Conclusion

We have shown that the SCL(FOL) calculus [9] can simulate model driven superposition [1] refutations deriving only non-redundant clauses. The superposition calculus cannot simulate SCL refutations due to its static a priori ordering.

In general, an SCL(FOL) learned clause is generated out of several resolution and factorization steps. From this perspective the SCL(FOL) calculus is more general and flexible than the superposition calculus. Furthermore, it only generates non-redundant clauses whereas any superposition implementation generates redundant clauses due to the syntactic application of the superposition inference rules.

Selection in superposition can also be simulated, but requires an additional branch in the SCL-SUP strategy, because selection of non-maximal, negative literals by superposition requires a different trail ordering for SCL in order to simulate a respective superposition left inference.

For future work, we plan to lift our simulation result from the ground case to the non-ground case. This lifting will require the extension of the SCL calculus by an additional rule that learns clauses that are computed as intermediate steps during the conflict analysis. This rule was left out of previous versions of SCL because we would never use it in a CDCL inspired SCL-run and because it would have complicated the termination and non-redundancy proofs for SCL. Nevertheless, we are confident that the rule can be designed in such a way that all properties of the original calculus still hold.

Considering the extension to the non-ground case, this result can be used in various directions. It can be used to develop an alternative implementation of the superposition calculus. Given a fixed ordering, the trail can be developed according to the ordering, generating only non-redundant superposition inferences. On the other hand, the concept of finite saturation can be kept this way preserving a strong mechanism for detecting satisfiability. Secondly, the result means that SCL can be used to naturally combine propagation driven reasoning with fixed ordering driven reasoning. This might overcome some of the issues of the current first-order portfolio approaches implemented in the state-of-the-art provers.

Another calculus contained in first-order reasoning portfolios is InstGen [12, 15]. It abstracts a first-order clause set to propositional logic via a grounding with a single constant. In case a CDCL sat solver proves the abstraction unsatisfiable, the first-order clause set is unsatisfiable too. For otherwise, the model found on the propositional level triggers an instantiation inference of a first-order clause. The instance rules out the before found propositional model modulo the abstraction.

The CDCL model building after grounding can be simulated via a respective SCL trail. This will then lead to a stuck state if SCL is restricted to the InstGen grounding. Now let C be the false first-order clause selected by InstGen for an instance. Then the SCL stuck state can be extended to a conflict state for C. Then SCL will not learn an instance of C, but a related clause that also rules out the previously found model on the propositional level. This way the relationship between InstGen and SCL can be investigated as well.

Acknowledgements. We thank our reviewers for their careful reading and constructive comments.

References

1. Bachmair, L., Ganzinger, H.: Rewrite-based equational theorem proving with selection and simplification. J. Logic Comput. **4**(3), 217–247 (1994). Revised version of Max-Planck-Institut für Informatik technical report, MPI-I-91-208, 1991
2. Bachmair, L., Ganzinger, H.: Resolution theorem proving. In: Robinson, A., Voronkov, A. (eds.) Handbook of Automated Reasoning, vol. I, chap. 2, pp. 19–99. Elsevier, Amsterdam (2001)
3. Bromberger, M.: A sorted datalog hammer for supervisor verification conditions modulo simple linear arithmetic. In: Fisman, D., Rosu, G. (eds.) TACAS 2022. LNCS, vol. 13243, pp. 480–501. Springer, Cham (2022). https://doi.org/10.1007/978-3-030-99524-9_27
4. Bromberger, M., Fiori, A., Weidenbach, C.: Deciding the Bernays-Schoenfinkel fragment over bounded difference constraints by simple clause learning over theories. In: Henglein, F., Shoham, S., Vizel, Y. (eds.) VMCAI 2021. LNCS, vol. 12597, pp. 511–533. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-67067-2_23
5. Bromberger, M., Gehl, T., Leutgeb, L., Weidenbach, C.: A two-watched literal scheme for first-order logic. In: Konev, B., Schon, C., Steen, A. (eds.) Proceedings of the Workshop on Practical Aspects of Automated Reasoning Co-located with the 11th International Joint Conference on Automated Reasoning (FLoC/IJCAR 2022), CEUR Workshop Proceedings, Haifa, Israel, 11–12 August 2022, vol. 3201. CEUR-WS.org (2022)
6. Bromberger, M., Jain, C., Weidenbach, C.: SCL(FOL) can simulate non-redundant superposition clause learning (2023)
7. Bromberger, M., Leutgeb, L., Weidenbach, C.: An efficient subsumption test pipeline for bs(lra) clauses. In: Blanchette, J., Kovacs, L., Pattinson, D. (eds.) Automated Reasoning - 11th International Joint Conference, IJCAR 2022, Held as Part of the Federated Logic Conference, Proceedings. LNCS, vol. 13385, pp. 147–168. Springer, Heidelberg (2022). https://doi.org/10.1007/978-3-031-10769-6_10
8. Bromberger, M., Schwarz, S., Weidenbach, C.: Exploring partial models with SCL. In: Konev, B., Schon, C., Steen, A. (eds.) Proceedings of the Workshop on Practical Aspects of Automated Reasoning Co-located with the 11th International Joint Conference on Automated Reasoning (FLoC/IJCAR 2022), CEUR Workshop Proceedings, Haifa, Israel, 11–12 August 2022, vol. 3201 (2022)
9. Bromberger, M., Schwarz, S., Weidenbach, C.: SCL(FOL) revisited (2023). https://doi.org/10.48550/ARXIV.2302.05954. https://arxiv.org/abs/2302.05954
10. Desharnais, M.: A formalization of the SCL(FOL) calculus: Simple clause learning for first-order logic. Archive of Formal Proofs (2023). https://isa-afp.org/entries/Simple_Clause_Learning.html, Formal proof development
11. Fiori, A., Weidenbach, C.: SCL clause learning from simple models. In: Fontaine, P. (ed.) CADE 2019. LNCS (LNAI), vol. 11716, pp. 233–249. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-29436-6_14
12. Ganzinger, H., Korovin, K.: New directions in instatiation-based theorem proving. In: Abramsky, S. (ed.) 18th Annual IEEE Symposium on Logic in Computer Science, LICS 2003, pp. 55–64. IEEE Computer Society (2003)
13. Knuth, D.E., Bendix, P.B.: Simple word problems in universal algebras. In: Leech, I. (ed.) Computational Problems in Abstract Algebra, pp. 263–297. Pergamon Press, Oxford (1970)

14. Korovin, K.: iProver – an instantiation-based theorem prover for first-order logic (system description). In: Armando, A., Baumgartner, P., Dowek, G. (eds.) IJCAR 2008. LNCS (LNAI), vol. 5195, pp. 292–298. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-71070-7_24
15. Korovin, K.: Inst-Gen – a modular approach to instantiation-based automated reasoning. In: Voronkov, A., Weidenbach, C. (eds.) Programming Logics. LNCS, vol. 7797, pp. 239–270. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-37651-1_10
16. Kovács, L., Voronkov, A.: First-order theorem proving and Vampire. In: Sharygina, N., Veith, H. (eds.) CAV 2013. LNCS, vol. 8044, pp. 1–35. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-39799-8_1
17. Leidinger, H., Weidenbach, C.: SCL(EQ): SCL for first-order logic with equality. In: Blanchette, J., Kovács, L., Pattinson, D. (eds.) Automated Reasoning - 11th International Joint Conference, IJCAR 2022, Haifa, Israel, 8–10 August 2022, Proceedings. Lecture Notes in Computer Science, vol. 13385, pp. 228–247. Springer, Heidelberg (2022). https://doi.org/10.1007/978-3-031-10769-6_14
18. Nieuwenhuis, R., Rubio, A.: Paramodulation-based theorem proving. In: Robinson, A., Voronkov, A. (eds.) Handbook of Automated Reasoning, vol. I, chap. 7, pp. 371–443. Elsevier (2001)
19. Schlichtkrull, A., Blanchette, J.C., Traytel, D., Waldmann, U.: Formalization of bachmair and ganzinger's ordered resolution prover. Archive of Formal Proofs (2018). https://isa-afp.org/entries/Ordered_Resolution_Prover.html, Formal proof development
20. Schulz, S., Cruanes, S., Vukmirović, P.: Faster, higher, stronger: E 2.3. In: Fontaine, P. (ed.) CADE 2019. LNCS (LNAI), vol. 11716, pp. 495–507. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-29436-6_29
21. Waldmann, U., Tourret, S., Robillard, S., Blanchette, J.: A comprehensive framework for saturation theorem proving. In: Peltier, N., Sofronie-Stokkermans, V. (eds.) Automated Reasoning - 10th International Joint Conference, IJCAR 2020, Paris, France, 1–4 July 2020, Proceedings, Part I. Lecture Notes in Computer Science, vol. 12166, pp. 316–334. Springer, Heidelberg (2020). https://doi.org/10.1007/978-3-030-51074-9_18
22. Weidenbach, C., Dimova, D., Fietzke, A., Kumar, R., Suda, M., Wischnewski, P.: SPASS version 3.5. In: Schmidt, R.A. (ed.) CADE 2009. LNCS (LNAI), vol. 5663, pp. 140–145. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-02959-2_10