An Isabelle/HOL Formalization of the SCL(FOL) Calculus

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

[1] Max Planck Institute for Informatics, Saarland Informatics Campus, Saarbrücken, Germany
{mbromber,desharnais,weidenbach}@mpi-inf.mpg.de

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

Abstract. We present an Isabelle/HOL formalization of Simple Clause Learning for first-order logic without equality: SCL(FOL). The main results are formal proofs of soundness, non-redundancy of learned clauses, termination, and refutational completeness. Compared to the unformalized version, the formalized calculus is simpler and more general, some results such as non-redundancy are stronger and some results such as non-subsumption are new. We found one bug in a previously published version of the SCL Backtrack rule. Compared to related formalizations, we introduce a new technique for showing termination based on non-redundant clause learning.

Keywords: interactive theorem proving · automated theorem proving · first-order logic · CDCL · SCL · non-redundant clause learning

1 Introduction

The SCL ("Clause Learning from Simple Models" or simply "Simple Clause Learning") family of calculi lifts a conflict-driven clause learning (CDCL) approach to first-order logic: SCL(FOL) is for first-order logic without equality [8,10], SCL(T) is for first-order logic with theories [6], SCL(EQ) is for first-order logic with equality [12], and HSCL is for exhaustive partial models exploration in first-order logic without equality [7]. In its original formulation [10], SCL(FOL) required exhaustive propagation and a precise strategy for the application of the rules in order to learn non-redundant clauses. This was improved upon by SCL(T) [6] by dropping exhaustive propagation and weakening the strategy, i.e., any run according to the strategy in [10] is also a run according to the strategy in [6]. The SCL(FOL) version presented in Bromberger et al. [8] integrates those changes and additionally refines the Backtrack rule.

We present an Isabelle/HOL formalization of the non-executable specification of SCL(FOL) based on and developed in parallel to Bromberger et al. The main results are soundness, non-redundancy of learned clauses, termination, and refutational completeness. In contrast to the goal of Bromberger et al. to guide toward an implementation, our goal is to be as simple and general as possible. For that, we (i) simplified the calculus (e.g., no more explicity tracking of decision levels), (ii) generalized the calculus (e.g., multiple acceptable positions in the Backtrack rule), (iii) strengthened existing theorems (e.g., Theorem 11 on non-redundancy), and (iv) proved new theorems (e.g., Corollary 12 on non-subsumption).

This work is part of the IsaFoL (Isabelle Formalization of Logic) effort [2], which aims at developing a library of results about logical calculi. The Isabelle theory files are available in the *Archive of Formal Proofs* (AFP) [9] and amount to more than 11 000 lines of source text. They build heavily upon many other entries of the AFP: (i) First_Order_Terms [17] for first-order terms, term substitutions, and MGU; (ii) Ordered_Resolution_Prover [14–16] for the clausal calculus, clause substitutions, Herbrand interpretation, and compactness of first-order logic; and (iii) Saturation_Framework_Extensions [5,18] for entailment of the clausal calculus. We contributed many lemmas and definitions back to both the Isabelle distribution and the aforementioned AFP entries (e.g., over 50 to First_Order_Terms). We made heavy use of the Isar language [19] to write structured proofs, the Sledgehammer tool [13] for proof automation, and locales [1]—Isabelle's parameterized module system—to structure our development and reuse existing components from the AFP entries. To ease associating the main results in this paper with their counterparts in the Isabelle development, names in `monospace` are taken verbatim from the formalization.

The formalization follows the basic ideas of the existing formalizations of the first-order resolution calculus [16] and propositional CDCL calculi [3,4]. Compared to propositional logic, first-order logic adds a number of challenges: the extra term level requires to consider variables, substitutions, groundings, and the concept of factorization. To preserve completeness, propagation of ground literals must not be exhaustive anymore, resulting in a level-wise exploration w.r.t. a bounding atom. Inside this bound, the calculus always terminates. If one level does not suffice to find a refutation, the bound can be increased and exploration can be continued. For unsatisfiable formulas, we prove the existence of a bound sufficient to derive $\bot$, which guarantees that only finitely many levels need to be explored.

The paper is now organized as follows. Section 2 recaps the SCL(FOL) calculus from Bromberger et al. as the basis of our formalization presented in Sect. 3. We first present the Isabelle formalization of the abstract rules of the SCL(FOL) calculus. Then we prove invariants preserved by the rules starting from the initial state, Lemma 1. Subsequently, we prove soundness, Theorem 7, non-redundancy of learned clauses, Theorem 11, termination with respect to a fixed bound, Theorem 18, and finally refutational completeness with respect to an appropriate bound, Theorem 20. We discuss important aspects of the formalization and proof ideas here and refer the reader to the formalization for more details. The paper ends with a short conclusion of the obtained results.

2 The SCL(FOL) Calculus

We shortly repeat basic first-order logic notions and the SCL(FOL) calculus presented in Bromberger et al. We consider an untyped, first-order logic without equality. A *term* is defined inductively as either a variable x or a function application $f(\overrightarrow{t})$ for a constant f and a (possibly-empty) list of terms $\overrightarrow{t}$. An *atom* is a predicate symbol applied to a list of term arguments. A *literal* is either a positive atom A or a negative atom $\neg A$. For literals we write L or K. The atom of a literal may be selected with $\text{atom}(A) = A$ and $\text{atom}(\neg A) = A$. The complement of a literal is defined as $\text{comp}(A) = \neg A$ and $\text{comp}(\neg A) = A$. A disjunctive *clause* is a finite multiset of literals. For clauses we write C or D. We use the syntax $L \vee C$ and $C \vee D$ synonymously with the multiset sums $\{L\} + C$ and $C + D$ respectively. We also use the syntax $\bot$ synonymously with the empty multiset $\{\}$. All variables in clauses are to be understood as universally quantified.

Substitutions are total unary functions from variables to terms. A substitution σ may be applied to a variable x, a term t, an atom A, a literal L, or a clause C, denoted $x\sigma$, $t\sigma$, $A\sigma$, $L\sigma$, or $C\sigma$ respectively. Substitution application is left-associative, i.e., $C\sigma_1\sigma_2 = (C\sigma_1)\sigma_2$. The domain of a substitution σ is defined as $\text{dom}(\sigma) = \{x \mid x\sigma \neq x\}$. The composition of two substitutions σ_1 and σ_2 is defined as the function $\sigma_1 \circ \sigma_2 = (\lambda x.\, x\sigma_1\sigma_2)$. A substitution γ is a *grounding* for a term t, an atom A, a literal L, or a clause C if $t\gamma$, $A\gamma$, $L\gamma$, or $C\gamma$ are respectively ground, i.e., if they do not contain variables. A substitution ρ is a *renaming* if it is injective and $x\rho$ is a variable for all variables x. The *inverse* of a renaming ρ is any function ρ^{-1} from terms to variables such that $\rho^{-1}\,(x\rho) = x$ for all variables x. The *restriction* of a substitution σ to a set of variables V is defined as the function $(\lambda x.\ \text{if}\ x \in V\ \text{then}\ x\sigma\ \text{else}\ x)$. A substitution σ is *idempotent* if $\sigma \circ \sigma = \sigma$. A substitution υ is a *unifier* for a set of terms T if $t_1\upsilon = t_2\upsilon$ for all terms $t_1 \in T$ and $t_2 \in T$. A substitution μ is a *most general unifier* (MGU) for a set of terms T if μ is a unifier for T and there exists a substitution σ such that $\mu \circ \sigma = \upsilon$ for all unifiers υ for T. A substitution μ is an *idempotent, most general unifier* (IMGU) for a set of terms T if μ is a unifier for T and $\mu \circ \upsilon = \upsilon$ for all unifiers υ for T; note that μ is an IMGU iff it is both idempotent and a MGU.

When formalizing logical calculi, IMGUs are preferable because they allow to apply groundings to a term both directly and after applying an IMGU, i.e., $t\gamma = t\mu\gamma$ for all terms t, groundings γ, and IMGU μ. Non-idempotent MGU do not have this property as the following counter-example shows. Consider the terms $t_1 = f(x, y, z)$ and $t_2 = f(w, y, z)$, the grounding $\gamma = \{x \mapsto a,\ y \mapsto b,\ z \mapsto c,\ w \mapsto a\}$, and the non-idempotent MGU $\mu = \{x \mapsto w,\ y \mapsto z,\ z \mapsto y\}$ where x, y, z, w are variables and a, b, c are ground constants, then we have $t_1\gamma = f(a, b, c) \neq f(a, c, b) = t_1\mu\gamma$. In published literature, an IMGU is often meant instead of an MGU; the idempotency requirement is often kept implicit because standard implementations for computing MGUs actually produce IMGUs.

The function $\text{gnd}(C) = \{C\gamma \mid C\gamma \text{ is ground}\}$ expresses the set of all groundings of a clause C. The function $\text{gnd}(N) = (\bigcup C \in N.\ \text{gnd}(C))$ expresses the set of all groundings of a set of clauses N; its subset whose clauses are restricted to atoms less than or equal to a bound β w.r.t. an order $\prec_B$ is defined as $\text{gnd}^{\preceq_B \beta}(N) = \{C \in \text{gnd}(N) \mid \forall L \in C.\ \text{atom}(L) \preceq_B \beta\}$. Note that $\text{gnd}(\text{gnd}(N)) = \text{gnd}(N)$. The strict order $\prec_B$ is total on ground literals and is such that for each β there are only finitely many literals L with $L \prec_B \beta$. An example of such an order could be KBO without zero-weight symbols. Note that LPO does not satisfy the last condition of a $\prec_B$ order although it is a well-founded and total order.

Herbrand entailment is defined as $(I \models_{\mathcal{H}} N \longleftrightarrow (\forall C \in N.\ I \models_{\mathcal{H}} C))$ for a set of clauses N, $(I \models_{\mathcal{H}} C \longleftrightarrow (\exists L \in C.\ I \models_{\mathcal{H}} L))$ for a clause C, $(I \models_{\mathcal{H}} A \longleftrightarrow A \in I)$, and $(I \models_{\mathcal{H}} \neg A \longleftrightarrow A \notin I)$ for a literal with atom A; note that the symbol $\models_{\mathcal{H}}$ is overloaded. Ground entailment is defined as $(N_1 \models_{\mathcal{G}} N_2 \longleftrightarrow (\forall I.\, I \models_{\mathcal{H}} N_1 \longrightarrow I \models_{\mathcal{H}} N_2))$. First-order entailment is defined as $(N_1 \models N_2 \longleftrightarrow \text{gnd}(N_1) \models_{\mathcal{G}} \text{gnd}(N_2))$. A set of ground clauses N is satisfiable if there exists a Herbrand interpretation I such that $I \models_{\mathcal{H}} N$; otherwise, it is unsatisfiable.

An annotated literal is the pairing of a literal with an annotation. We call it a *decision literal* when the annotation is a natural number n indicating the literal's level (i.e., that it is the nth decision) and a *propagation literal* when the annotation is a closure of the clause the literal originated from. The literal of an annotated literal $\mathcal{K}$ is denoted $\text{lit}(\mathcal{K})$ and the annotation is denotated $\text{ann}(\mathcal{K})$. The level of a clause is the maximum level of its literals. A *trail* is a finite sequence of annotated ground literals: it grows from left to right. The empty trail is written ϵ and appending a new annotated literal $\mathcal{K}$ to a trail Γ is written $\Gamma,\mathcal{K}$. The concatenation of two trails Γ_1 and Γ_2 is written Γ_2, Γ_1. A trail Γ can be converted to a set with $\text{set}(\Gamma)$.

A literal L is true under trail Γ if $L \in \{\text{lit}(\mathcal{K}) \mid \mathcal{K} \in \text{set}(\Gamma)\}$. A literal L is false under trail Γ if $\text{comp}(L) \in \{\text{lit}(\mathcal{K}) \mid \mathcal{K} \in \text{set}(\Gamma)\}$. A literal L is defined in a trail Γ if L is true or false under Γ; otherwise, it is undefined. A clause C is true under trail Γ if ($\exists L \in C.$ L is true under Γ). A clause C is false under trail Γ if ($\forall L \in C.$ L is false under Γ). A clause C is defined in a trail Γ if ($\forall L \in C.$ L is defined in Γ); otherwise, it is undefined.

The SCL(FOL) calculus is defined as a transition system operating on states $(\Gamma; N; U; \beta; k; \mathcal{C})$ where Γ is a trail, N is a finite set of initial clauses, U is a finite set of learned clauses, β is a bounding atom restricting the considered ground literals, k is a natural number counting the number of decisions taken in Γ, and $\mathcal{C}$ is either $\top$ or a clause closure $(C; \gamma)$ such that $C\gamma$ is ground and false in Γ. The initial state is $(\epsilon; N; \emptyset; \beta; 0; \top)$ for some initial clause set N and bound β.

The transition relation $\Rightarrow_{\text{SCL}}$ is a mapping between states. The rules below are from Bromberger et al. and serve as a reference for the Isabelle formalization described in Sect. 3.

Propagate $(\Gamma; N; U; \beta; k; \top) \ \Rightarrow_{\text{SCL}} \ (\Gamma, L\gamma^{((C_0 \vee L)\mu;\gamma)}; N; U; \beta; k; \top)$
if $(C \vee L) \in (N \cup U)$, $C = C_0 \vee C_1$, $C_1\gamma = L\gamma \vee \cdots \vee L\gamma$, $C_0\gamma$ does not contain $L\gamma$, μ is the IMGU of the literals in C_1 and L, $(C \vee L)\gamma$ is ground, $(C \vee L)\gamma \prec_B \{\beta\}$, $C_0\gamma$ false under Γ, and $L\gamma$ is undefined in Γ.

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

Conflict $(\Gamma; N; U; \beta; k; \top) \Rightarrow_{\mathrm{SCL}} (\Gamma; N; U; \beta; k; (C; \gamma))$
if $C \in (N \cup U)$, $C\gamma$ is false under Γ for a grounding substitution γ.

These rules construct a (partial) model via the trail Γ for $N \cup U$ until a conflict, i.e., a clause false under Γ is found. The above rules always terminate, because there are only finitely many ground literals L with $L \prec_B \beta$. It might be necessary to successively increase β for full refutational completeness.

Skip $(\Gamma, K; N; U; \beta; k; (C; \gamma)) \Rightarrow_{\mathrm{SCL}} (\Gamma; N; U; \beta; k - i; (C; \gamma))$
if comp(K) does not occur in $C\gamma$, if K is a decision literal then $i = 1$; otherwise, $i = 0$.

Factorize $(\Gamma; N; U; \beta; k; (C \lor L \lor L'; \gamma)) \Rightarrow_{\mathrm{SCL}} (\Gamma; N; U; \beta; k; ((C \lor L)\mu; \gamma))$
if $L\gamma = L'\gamma$ and $\mu = \mathrm{IMGU}(L, L')$.

Note that this rule may be used multiple times if the conflicting clause contains more than two duplicates of a given literal or if multiple distinct literals have duplicates.

Resolve $(\Gamma, K\gamma_D{}^{(D \lor K; \gamma_D)}; N; U; \beta; k; (C \lor L; \gamma_C))$
$\Rightarrow_{\mathrm{SCL}} (\Gamma, K\gamma_D{}^{(D \lor K; \gamma_D)}; N; U; \beta; k; ((C \lor D)\mu; \gamma_C \circ \gamma_D))$
if $K\gamma_D = \mathrm{comp}(L\gamma_C)$, $\mu = \mathrm{IMGU}(K, \mathrm{comp}(L))$.

The clauses $D \lor K$ and $C \lor L$ are assumed to have disjoint variables.

Backtrack $(\Gamma_0, K, \Gamma_1, \mathrm{comp}(L\gamma)^k; N; U; \beta; k; (C \lor L; \gamma))$
$\Rightarrow_{\mathrm{SCL}} (\Gamma_0; N; U \cup \{C \lor L\}; \beta; j; \top)$
if $C\gamma$ is of level $i' < k$, and Γ_0, K is the minimal trail subsequence such that there is a grounding substitution γ' with $(C \lor L)\gamma'$ is false under Γ_0, K but not in Γ_0, and Γ_0 is of level j.

The clause $C \lor L$ added by the rule Backtrack to U is called a *learned clause.* The empty clause $\bot$ can only be generated by rule Resolve or be already present in N, hence, as usual for CDCL-style calculi, the generation of $\bot$ together with the clauses in $N \cup U$ represent a resolution refutation.

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.

3 Formalization of the SCL(FOL) Calculus

The formalization introduces some new concepts absent from Sect. 2. A multiset C can be converted to a set, i.e., without duplicates, with set(C). The multiplicity of an element x in a multiset C is denoted by count(C, x). The cardinality of a multiset—the sum of the multiplicities of its elements—is denoted by $|C|$. The multiset whose only element is x with multiplicity n is denoted by repeat(n, x); note that count(repeat$(n, x), x) = n$, and set(repeat$(n, x)) = \{x\}$ if $n > 0$. The multiset extension of an order on literals extends the order to multisets containing literals; we use the Huet-Oppen specification [11], one of several equivalent alternatives for this extension. The *adaptation* of a substitution σ to a renaming ρ is a function whose domain is the renamed domain of σ and whose codomain is the same as σ; it is defined as the function $(\lambda x. \text{if } x \in \{y\rho \mid y \in \text{dom}(\sigma)\} \text{ then } (\rho^{-1}\, x)\sigma \text{ else } x)$. A substitution γ is a *merged grounding* of a grounding γ_A for a set of variables A and a grounding γ_B for a set of variables B if $(A \cap B = \{\} \longrightarrow (\forall x \in A.\ x\gamma_A \text{ is ground}) \longrightarrow (\forall x \in B.\ x\gamma_B \text{ is ground}) \longrightarrow (\forall x \in A.\ x\gamma = x\gamma_A) \wedge (\forall x \in B.\ x\gamma = x\gamma_B))$; an example of a function that fulfills this specification is $(\lambda x. \text{ if } x \in A \text{ then } x\gamma_A \text{ else } x\gamma_B)$. The length of a trail Γ is denoted by $|\Gamma|$. The nth right-most element of a trail Γ is denoted by $\Gamma[n]$; we use zero-based indexing where the right-most element is the 0th element. The Herbrand interpretation of a trail Γ is defined as $\mathcal{HI}(\Gamma) = (\bigcup \mathcal{K} \in \text{set}(\Gamma).\ \text{case lit}(\mathcal{K}) \text{ of } A \Rightarrow \{A\} \mid \neg A \Rightarrow \{\})$.

The formalization also changes some existing concepts. No distinction is made between *atoms* and terms, so first-order terms are used everywhere in place of atoms. The level annotation of a *decision literal* is not required anymore and replaced by a † marker, it is now written $K = (K; \dagger)$ for some literal K. A *propagation literal* is written $(K\gamma_D)^{(K;D;\gamma_D)} = (K\gamma_D; (D; K; \gamma_D))$ for some literal K, clause D, and grounding γ_D. Note that the propagated literal is explicitly separated from its clause in the closure annotation; this eases the formulation of the additional invariants 5 and 6 of Lemma 1., that the respective clause is always false under the respective trail. For the *trail* $\Gamma,\mathcal{K}$, the Isabelle formalization uses the constructor `List.Cons` $\mathcal{K}\,\Gamma$ which actually grows from right to left. However, we keep the well-established left-to-right convention in this paper because it significantly eases the presentation. An state is a tuple $(\Gamma; U; \mathcal{C})$ where Γ is a trail, U is a finite set of learned clauses, and $\mathcal{C}$ is an optional clause closure. The individual components can be selected with $\text{trail}((\Gamma; U; \mathcal{C})) = \Gamma$, $\text{learned}((\Gamma; U; \mathcal{C})) = U$, and $\text{conflict}((\Gamma; U; \mathcal{C})) = \mathcal{C}$. The *initial state* is $(\epsilon; \{\}; \top)$, i.e., empty trail, no learned clauses, and no conflicting closure. The finite set of initial clauses N and the bounding atom β are no longer stored in the state but are rather parameters of the transition relation; this was done to highlight the fact that they are never modified by any rule. The natural number k counting the number of decisions, used in Sect. 2 to determine an appropriate backtracking point, turned out not to be necessary and was dropped entirely. We assume the existence of a binary relation on atoms $\prec_B$ such that $(\forall \beta.\ \{t \mid t \prec_B \beta\} \text{ is finite})$ but dropped the requirement for $\prec_B$ to be a strict order total on ground terms. We also don't lift $\prec_B$ to literals and clauses, but always use it at the atom level. We define the relation $\preceq_B$ as the reflexive closure of $\prec_B$.

The transition relation $\Rightarrow_{\text{SCL}}^{N,\beta}$ is a binary predicate between states and is parameterized by the finite set N of initial clauses and the bounding atom β. It is defined as the disjunction of the following rules. Following each rule, we highlight the main differences from Sect. 2 not already covered.

Propagate $\quad (\Gamma; U; \top) \Rightarrow_{\text{Propagate}}^{N,\beta} (\Gamma, (L\mu\gamma)^{(L\mu;C_0\mu;\gamma)}; U; \top)$
if $(L \vee C) \in (N \cup U)$, γ is a grounding for $L \vee C$, $(\forall K \in (L \vee C).\ \text{atom}(K\gamma) \preceq_B \beta)$, $C_0 = \{K \in C \mid K\gamma \neq L\gamma\}$, $C_1 = \{K \in C \mid K\gamma = L\gamma\}$, $C_0\gamma$ is false under Γ, $L\gamma$ is undefined in Γ, and μ is an IMGU for all terms in $\{\text{atom}(K) \mid K \in (L \vee C_1)\}$.

Compared to Sect. 2, we express the splitting of C into C_0 and C_1 formally as set operations and replace $\prec_B$ with $\preceq_B$. This replacement has no effect on the results but allowing the bound β to be in $\text{gnd}^{\preceq_B \beta}(N)$ eases the proof of Lemma 21, where the largest element of the (finite) unsatisfiable core is directly used as new bound. There are also situations where the maximal element of a signature is required to derive a contradiction: a non-strict bound requires to artificially extend the signature while a non-strict bound does not.

Decide $(\Gamma; U; \top) \Rightarrow^{N,\beta}_{\text{Decide}} (\Gamma, (L\gamma); U; \top)$
if $(L \vee C) \in N$, γ is a grounding for L, $L\gamma$ is undefined in Γ, and $\text{atom}(L\gamma) \preceq_B \beta$.

Compared to Sect. 2, we replace $\prec_B$ with $\preceq_B$ and take the decision literal from N instead of $N \cup U$. The ground instances of literals of U are a subset of the ground instances of literals of N so it is redundant to also consider U here.

Conflict $(\Gamma; U; \top) \Rightarrow^{N,\beta}_{\text{Conflict}} (\Gamma; U; (C; \gamma))$
if $C \in (N \cup U)$, γ is a grounding for C, and $C\gamma$ is false under Γ.

Skip $(\Gamma, \mathcal{K}; U; (C; \gamma)) \Rightarrow^{N,\beta}_{\text{Skip}} (\Gamma; U; (C; \gamma))$
if $\text{comp}(\text{lit}(\mathcal{K})) \notin C\gamma$.

Factorize $(\Gamma; U; (L' \vee L \vee C; \gamma)) \Rightarrow^{N,\beta}_{\text{Factorize}} (\Gamma; U; ((L \vee C)\mu; \gamma))$
if $L\gamma = L'\gamma$ and μ is the IMGU for the terms $\text{atom}(L)$ and $\text{atom}(L')$.

Resolve $(\Gamma; U; (L \vee C; \gamma_C)) \Rightarrow^{N,\beta}_{\text{Resolve}} (\Gamma; U; ((C\rho_C \vee D\rho_D)\mu; \gamma))$
if $\Gamma = \Gamma', (K\gamma_D)^{(K;D;\gamma_D)}$, and $K\gamma_D = \text{comp}(L\gamma_C)$, ρ_C and ρ_D are renamings such that the variables of $(L \vee C)\rho_C$ and $(K \vee D)\rho_D$ are disjoint, μ is the IMGU for the terms $\text{atom}(L)\rho_C$ and $\text{atom}(K)\rho_D$, γ'_C and γ'_D are adaptations of γ_C and γ_D to the renamings ρ_C and ρ_D respectively, and γ is a merged grounding of γ'_C for the variables of $(L \vee C)\rho_C$ and γ'_D for the variables of $(K \vee D)\rho_D$.

Note that the definition of merged grounding implies the following equalities: $\mu \circ \gamma = \gamma$, $L\rho_C\gamma = L\gamma_C$, $C\rho_C\gamma = C\gamma_C$, $K\rho_D\gamma = K\gamma_D$, and $D\rho_D\gamma = D\gamma_D$.

Compared to Sect. 2, we explicitly rename the merged clauses to avoid variable-name clashes instead of assuming disjoint variables, and use an abstract specification for the merged grounding instead of forcing substitution composition. The latter makes our rule more general by allowing more freedom to an implementation.

Backtrack $(\Gamma, \Gamma', K; U; (L \vee C; \gamma)) \Rightarrow^{N,\beta}_{\text{Backtrack}} (\Gamma; \{L \vee C\} \cup U; \top)$
if $K = \text{comp}(L\gamma)$ and $(\nexists \gamma'.\ (L \vee C)\gamma'$ is ground and false under $\Gamma)$.

Compared to Sect. 2, we allow backtracking to any non-conflicting trail instead of specifying the position. This makes our rule more general by, again, allowing more freedom to an implementation. The minimally backtracking strategy introduced in Definition 4 brings back equivalence to the Backtrack rule of Sect. 2.

Isabelle Technicalities. We define the SCL rules in the `scl_fol_calculus` locale. It fixes an abstract binary relation $\prec_B$ as a locale parameter and assumes that it bounds a finite number of atoms. It also fixes an abstract function to generate variable renamings as a locale parameter and assumes its correctness; this function is not required for the specification of the calculus but is required in multiple proofs. Most of the following definitions and theorems are in the context of this locale. Each SCL rule is defined separately as an inductive predicate. Having separate definitions allows to refer to the rules individually in subsequent definitions and theorems. Using inductive predicates, as opposed to plain definitions, is convenient because Isabelle automatically generates some useful introduction and elimination lemmas, and configures structured Isar syntax for case analysis.

From the SCL rules, we can prove a number of invariants about states. Most of them are intuitive while few are technicalities of the Isabelle formalization. We will use the invariants as hypotheses for many of the main lemmas and theorems.

Lemma 1 (`scl_state_invariants`). *Let $(\Gamma; U; \mathcal{C})$ be an state w.r.t. $\Rightarrow_{SCL}^{N,\beta}$. The following invariants hold for the initial state $(\epsilon; \{\}; \top)$ and are each individually preserved by the SCL rules.*

1. *All annotated literals in Γ are ground.*
   – $\forall K \in \{\mathit{lit}(\mathcal{K}) \,|\, \mathcal{K} \in \mathit{set}(\Gamma)\}.\ K$ *is a ground literal*
2. *The atoms of all annotated literals in Γ are $\preceq_B \beta$.*
   – $\forall K \in \{\mathit{lit}(\mathcal{K}) \,|\, \mathcal{K} \in \mathit{set}(\Gamma)\}.\ \mathit{atom}(K) \preceq_B \beta$
3. *All annotated literals in Γ are undefined in their respective subtrail of Γ.*
   – $\forall \Gamma'\, \mathcal{K}\, \Gamma''.\ \Gamma = \Gamma', \mathcal{K}, \Gamma'' \longrightarrow \mathit{lit}(\mathcal{K})$ *is undefined in* Γ'
4. *All closures in Γ and $\mathcal{C}$ are ground.*
   – $\forall \mathcal{K} \in \mathit{set}(\Gamma).\ \forall D\, K\, \gamma.\ \mathcal{K} = (K\gamma)^{(K;D;\gamma)} \longrightarrow D\gamma$ *is ground*
   – $\forall C\, \gamma.\ \mathcal{C} = (C;\gamma) \longrightarrow C\gamma$ *is ground*
5. *All closures in Γ and $\mathcal{C}$ are false under their respective subtrail of Γ.*
   – *invariant 4. holds*
   – $\forall D\, K\, \gamma\, \Gamma'\, \Gamma''.\ \Gamma = \Gamma', (K\gamma)^{(K;D;\gamma)}, \Gamma'' \longrightarrow D\gamma$ *is false under* Γ'
   – $\forall C\, \gamma.\ \mathcal{C} = (C;\gamma) \longrightarrow C\gamma$ *is false under* Γ
6. *All propagated literals in Γ are the grounding of the non-ground literal in their closure annotations.*
   – $\forall \mathcal{K} \in \mathit{set}(\Gamma).\ \forall D\, K\, \gamma.\ \mathit{ann}(\mathcal{K}) = (D; K; \gamma) \longrightarrow \mathit{lit}(\mathcal{K}) = K\gamma$
7. *The complements of all propagated literals in Γ are absent from their closure annotation.*
   – $\forall \mathcal{K} \in \mathit{set}(\Gamma).\ \forall D\, K\, \gamma.\ \mathcal{K} = (K\gamma)^{(K;D;\gamma)} \longrightarrow \mathit{comp}(K\gamma) \notin D\gamma$
8. *All literals of the clauses in Γ's propagating clauses, U, and $\mathcal{C}$ have a corresponding, more general literal in N.*
   – $\forall D \in \{D \,|\, (K\gamma)^{(K;D;\gamma)} \in \mathit{set}(\Gamma)\} \cup U \cup (\mathit{if}\ \mathcal{C} = (C;\gamma)\ \mathit{then}\ \{C\}\ \mathit{else}\ \{\}).$
   $\forall K \in D.\ \exists D' \in N.\ \exists K' \in D'.\ \exists \sigma.\ K'\sigma = K$
9. *All annotated literals in Γ have a corresponding more general literal either in N or in U.*
   – $\forall \mathcal{K} \in \mathit{set}(\Gamma).\ \exists L \in N \cup U.\ \exists \sigma.\ L\sigma = \mathit{lit}(\mathcal{K})$
10. *All clauses in Γ, U, and $\mathcal{C}$ are entailed by N.*
    – $\forall \mathcal{K} \in \mathit{set}(\Gamma).\ \forall D\, K\, \gamma.\ \mathcal{K} = (K\gamma)^{(K;D;\gamma)} \longrightarrow N \models \{K \vee D\}$
    – $N \models U$
    – $\forall C\, \gamma.\ \mathcal{C} = (C;\gamma) \longrightarrow N \models \{C\}$

The SCL calculus is defined as a transition system where many decisions are deferred to strategies. A *strategy* specifies a transition system whose transitions are a subset of those from an existing transition system. We say that a strategy $\mathcal{S}$ *restricts* a transition system $\mathcal{T}$ (or symmetrically that $\mathcal{T}$ is *restricted* by $\mathcal{S}$) if $(\forall x\, y.\ \mathcal{S}\, x\, y \longrightarrow \mathcal{T}\, x\, y)$. Note that strategies can be chained to iteratively apply more restrictions.

We define the reasonable and regular strategies restricting the $\Rightarrow^{N,\beta}_{\mathrm{SCL}}$ relation in order to prove the main results of this paper.

Definition 2. *The reasonable strategy $\Rightarrow^{N,\beta}_{\mathit{Rea\text{-}SCL}}$ restricts the SCL calculus by preventing decisions that immediately lead to a conflict. Such situations could be replaced by a propagation. Formally:*

$$S \Rightarrow^{N,\beta}_{\mathit{Rea\text{-}SCL}} S' \;\longleftrightarrow\; S \Rightarrow^{N,\beta}_{\mathit{SCL}} S' \wedge (S \Rightarrow^{N,\beta}_{\mathit{Decide}} S' \longrightarrow (\nexists S''. S' \Rightarrow^{N,\beta}_{\mathit{Conflict}} S''))$$

Definition 3. *The regular strategy $\Rightarrow^{N,\beta}_{\mathit{Reg\text{-}SCL}}$ restricts the reasonable strategy by prioritizing the conflict rule to any other. Formally:*

$$S \Rightarrow^{N,\beta}_{\mathit{Reg\text{-}SCL}} S' \;\longleftrightarrow\; S \Rightarrow^{N,\beta}_{\mathit{Rea\text{-}SCL}} S' \wedge ((\exists S''.\, S \Rightarrow^{N,\beta}_{\mathit{Conflict}} S'') \longrightarrow S \Rightarrow^{N,\beta}_{\mathit{Conflict}} S')$$

While not required for the coming results, we also define the minimally backtracking strategy to express the constraint on the backtracking position found in Sect. 2.

Definition 4. *The minimally backtracking strategy $\Rightarrow^{N,\beta}_{\mathit{Min\text{-}Bac\text{-}SCL}}$ restricts the regular strategy by requiring that backtracking removes the shortest possible suffix of the trail. Formally:*

$$\begin{aligned} S \Rightarrow^{N,\beta}_{\mathit{Min\text{-}Bac\text{-}SCL}} S' \;\longleftrightarrow\; & S \Rightarrow^{N,\beta}_{\mathit{Reg\text{-}SCL}} S' \wedge (S \Rightarrow^{N,\beta}_{\mathit{Backtrack}} S' \longrightarrow \\ & \mathit{trail}(S') \text{ is the longest prefix of } \mathit{trail}(S) \\ & \text{not in conflict with the learned clause}) \end{aligned}$$

All three strategies build on one-another and ultimately restrict the SCL relation. We can express this formally as implications, of which the first can be used to show that coming results (e.g., Corollaries 13 and 19) also hold for the minimally backtracking strategy.

Lemma 5 (`strategy_restrictions`). *The minimally backtracking strategy restricts the regular strategy, which restricts the reasonable strategy, which restricts the SCL calculus. Formally:*

- $\forall N\, \beta\, S\, S'.\ S \Rightarrow^{N,\beta}_{\mathit{Min\text{-}Bac\text{-}SCL}} S' \longrightarrow S \Rightarrow^{N,\beta}_{\mathit{Reg\text{-}SCL}} S'$
- $\forall N\, \beta\, S\, S'.\ S \Rightarrow^{N,\beta}_{\mathit{Reg\text{-}SCL}} S' \longrightarrow S \Rightarrow^{N,\beta}_{\mathit{Rea\text{-}SCL}} S'$
- $\forall N\, \beta\, S\, S'.\ S \Rightarrow^{N,\beta}_{\mathit{Rea\text{-}SCL}} S' \longrightarrow S \Rightarrow^{N,\beta}_{\mathit{SCL}} S'$

The bounding atom β restricts the calculus to only consider the finitely many ground atoms less than or equal to β w.r.t. $\prec_B$; this will play an important role in the termination proof. When SCL terminates, it either derived a contradiction, or it found a model for the bounded groundings of the initial clauses. Because β is usually chosen heuristically, the model might be unsatisfactory for the considered use case and one may want to continue execution with a bigger bound. This is allowed if the new bound properly extends the previous bound β w.r.t. $\preceq_B$.

Theorem 6 (`monotonicity_wrt_bound`). *If the ground atoms bound by β are a subset of the ground atoms bound by β', formally if $(\forall A.\ A$ is ground $\longrightarrow A \preceq_B \beta \longrightarrow A \preceq_B \beta')$, then the SCL, reasonable SCL, regular SCL, and minimally backtracking transitions w.r.t. β are also transitions w.r.t. β', formally*

- $\forall N\, S\, S'.\ S \Rightarrow^{N,\beta}_{\mathit{SCL}} S' \longrightarrow S \Rightarrow^{N,\beta'}_{\mathit{SCL}} S'$,
- $\forall N\, S\, S'.\ S \Rightarrow^{N,\beta}_{\mathit{Rea\text{-}SCL}} S' \longrightarrow S \Rightarrow^{N,\beta'}_{\mathit{Rea\text{-}SCL}} S'$,
- $\forall N\, S\, S'.\ S \Rightarrow^{N,\beta}_{\mathit{Reg\text{-}SCL}} S' \longrightarrow S \Rightarrow^{N,\beta'}_{\mathit{Reg\text{-}SCL}} S'$, *and*
- $\forall N\, S\, S'.\ S \Rightarrow^{N,\beta}_{\mathit{Min\text{-}Bac\text{-}SCL}} S' \longrightarrow S \Rightarrow^{N,\beta'}_{\mathit{Min\text{-}Bac\text{-}SCL}} S'$.

Theorem 6 implies that all properties w.r.t. a bound β also hold w.r.t. a compatible bound β'. Its hypothesis is fulfilled if $\preceq_B$ is transitive on ground atoms, β and β' are ground atoms, and $\beta \preceq_B \beta'$. The bounding atom could even be increased at any point in an SCL run, not just when the calculus terminated.

The different rules and strategies considered so far express a single step of computation for the SCL calculus; they offer a good level of granularity to both understand and mechanize the details of the calculus. But many results of the following sections ought to express properties of the calculus as a whole. We express such results in terms of a run from the initial state. A *run* is the reflexive, transitive closure of a rule or strategy, e.g. $S\ (\Rightarrow^{N,\beta}_{\mathrm{SCL}})^*\ S'$ is an SCL run from the state S to the state S'.

The soundness of the individual SCL rules is shown by invariant 10. We now consider the soundness of terminating runs of the SCL calculus as a whole.

Theorem 7 (`correct_termination`). *Let $S = (\Gamma; U; \mathcal{C})$ be a state w.r.t. $\Rightarrow^{N,\beta}_{\mathit{SCL}}$. If invariants 2, 3, 5, 6 and 10 hold for S, and if S is a stuck state with some restrictions, formally if*

- $\nexists S'.\ S \Rightarrow^{N,\beta}_{\mathit{Propagate}} S'$,
- $\nexists S'.\ S \Rightarrow^{N,\beta}_{\mathit{Decide}} S' \wedge (\nexists S''.\ S' \Rightarrow^{N,\beta}_{\mathit{Conflict}} S'')$,
- $\nexists S'.\ S \Rightarrow^{N,\beta}_{\mathit{Conflict}} S'$,
- $\nexists S'.\ S \Rightarrow^{N,\beta}_{\mathit{Skip}} S'$,
- $\nexists S'.\ S \Rightarrow^{N,\beta}_{\mathit{Resolve}} S'$,
- $\nexists S'.\ S \Rightarrow^{N,\beta}_{\mathit{Backtrack}} S'$ *and the backtracking is minimal,*

then either the conflicting clause $\bot$ has been derived and the groundings $gnd(N)$ of the initial clauses N are unsatisfiable, or there is no conflicting clause and the groundings $gnd^{\preceq_B \beta}(N)$ of the initial clauses N are satisfiable by the trail, formally either

- $(\exists \gamma.\ \mathcal{C} = (\bot; \gamma)) \wedge (\nexists I.\ I \models_{\mathcal{H}} gnd(N))$, *or*

– $\mathcal{C} = \top \wedge \mathcal{HI}(\Gamma) \models_{\mathcal{H}} gnd^{\preceq_B \beta}(N)$.

Note that no hypothesis restricts the usage of the Factorize rule because it is an optional step of conflict resolution that has no impact on satisfiability.

Theorem 7 holds for a family of strategies, in contrast to Theorem 5 from Bromberget et al., which was only shown for what is here called the minimally backtracking strategy. This family of strategies contains any strategy that preserves the required invariants and is restricted by the minimally backtracking strategy. From Lemma 5 we know that these two requirements are fulfilled by the SCL relation but also by the reasonable, regular, and minimally backtracking strategies. This leads to a more intuitive corollary based on runs.

Corollary 8 (`correct_termination_strategies`). *If an SCL, reasonable SCL, regular SCL, or minimally backtracking SCL run starting from the initial state* $(\epsilon; \{\}; \top)$ *terminates in a state* $S = (\Gamma; U; \mathcal{C})$*, formally any of*

– $(\epsilon; \{\}; \top)\,(\Rightarrow^{N,\beta}_{SCL})^*\, S \;\wedge\; (\nexists S'.\, S \Rightarrow^{N,\beta}_{SCL} S')$,
– $(\epsilon; \{\}; \top)\,(\Rightarrow^{N,\beta}_{Rea\text{-}SCL})^*\, S \;\wedge\; (\nexists S'.\, S \Rightarrow^{N,\beta}_{Rea\text{-}SCL} S')$,
– $(\epsilon; \{\}; \top)\,(\Rightarrow^{N,\beta}_{Reg\text{-}SCL})^*\, S \;\wedge\; (\nexists S'.\, S \Rightarrow^{N,\beta}_{Reg\text{-}SCL} S')$, *or*
– $(\epsilon; \{\}; \top)\,(\Rightarrow^{N,\beta}_{Min\text{-}Bac\text{-}SCL})^*\, S \;\wedge\; (\nexists S'.\, S \Rightarrow^{N,\beta}_{Min\text{-}Bac\text{-}SCL} S')$,

then the conclusion of Theorem 7 holds.

Note that each strategy is used with positive polarity in the "run" hypothesis and negative polarity in the "no-more-step" hypothesis. For this reason, it is impossible to provide a corollary with a single requirement to restrict or be restricted by any known strategy.

Traditional saturation-based calculi for first-order logic, e.g. Resolution and Superposition, can learn redundant clauses and thus their implementations require costly checks for non-redundancy. SCL(FOL) learns only non-redundant clauses. Thus, an implementation would not need to check for (forward) non-redundancy. We first repeat the definition of *standard redundancy* as found in [18].

Definition 9. *A clause* C *is redundant w.r.t. a set of clauses* N *and a strict order on clauses* $\prec$ *if* $(\forall C' \in gnd(C).\ \{D' \in gnd(N) \mid D' \prec C'\} \models_{\mathcal{G}} C')$.

We first prove non-redundancy w.r.t. a trail-induced dynamic order and then lift this result to non-redundancy w.r.t. a static order.

Definition 10. *A trail* Γ *induces a well-founded, strict partial order* $\prec^{\Gamma}$*, total on all atoms in* Γ*'s literals. Assuming* Γ *has the form* $L_n^*, \ldots, L_2^*, L_1^*, L_0^*$ *for all* $* \in \{\dagger, (D, \gamma_D)$ *for some* D *and* $\gamma_D\}$*, we have the following ordering.*

$$atom(L_n) \prec^{\Gamma} \cdots \prec^{\Gamma} atom(L_2) \prec^{\Gamma} atom(L_1) \prec^{\Gamma} atom(L_0)$$

In other words, "older" elements on the left are smaller than "newer" elements on the right. Formally:

$$t_1 \prec^{\Gamma} t_2 \longleftrightarrow (\exists i < |\Gamma|.\ \exists j < i.\ t_1 = atom(lit(\Gamma[i])) \wedge t_2 = atom(lit(\Gamma[j])))$$

Compared to Bromberger et al., the trail-induced order is defined on atoms instead of literals and non-redundancy is proven for any lifting to literals.

Theorem 11 (`dynamic_non_redundancy_regular_scl`). *Following conflict resolution in a regular run, formally if*

- $(\epsilon; \{\}; \top)\,(\Rightarrow^{N,\beta}_{\mathit{Reg\text{-}SCL}})^*\,(\Gamma; U; \top)$,
- $(\Gamma; U; \top) \Rightarrow^{N,\beta}_{\mathit{Conflict}} S_1$,
- $S_1\,(\Rightarrow^{N,\beta}_{\mathit{Skip,Factorize,Resolve}})^+\,S_n$, *and*
- $S_n \Rightarrow^{N,\beta}_{\mathit{Backtrack}} S_{1+n}$,

then neither is the learned clause $C = \mathit{conflict}(S_n)$ generalized by any initial or learned clause, formally $(\nexists D \in N \cup U.\, \exists \sigma.\, D\sigma = C)$, nor is it redundant w.r.t. $N \cup U$ and the order we get by first lifting the trail-induced order $\prec^{\Gamma}$ from atoms to literals and then taking its multiset extension.

Dynamic non-redundancy with respect to the trail-induced order does not by itself release an implementation from performing backward non-redundancy checks, but it is a strong guarantee on the quality of learned clauses. For backward redundancy checks an order needs to be used that encompasses all dynamic trail-induced orders. An order based on a strict multiset relation has this property. So for backward redundancy we can, e.g., delete subsumed clauses.

Corollary 12 (`static_non_subsumption_regular_scl`). *If a regular run starting from the initial state $(\epsilon; \{\}; \top)$ learns a clause C, formally if*

- $(\epsilon; \{\}; \top)\,(\Rightarrow^{N,\beta}_{\mathit{Reg\text{-}SCL}})^*\,(\Gamma; U; (C; \gamma))$ *and*
- $(\Gamma; U; (C; \gamma)) \Rightarrow^{N,\beta}_{\mathit{Backtrack}} S$,

then C is not subsumed by any of the initial or learned clauses, formally $\nexists D \in N \cup U.\, \exists \sigma.\, D\sigma \subseteq C$.

All non-redundancy results can be generalized to an arbitrary strategy restricting the regular strategy. We only show one example here and refer the reader to the formalization for the others.

Corollary 13 (`dynamic_non_redundancy_strategy`). *Following conflict resolution in the run of a strategy restricting regular SCL, formally if*

- $(\epsilon; \{\}; \top)\,(\Rightarrow^{N,\beta}_{\mathit{Strategy}})^*\,(\Gamma; U; \top)$,
- $(\Gamma; U; \top) \Rightarrow^{N,\beta}_{\mathit{Conflict}} S_1$,
- $S_1\,(\Rightarrow^{N,\beta}_{\mathit{Skip,Factorize,Resolve}})^+\,S_n$,
- $S_n \Rightarrow^{N,\beta}_{\mathit{Backtrack}} S_{1+n}$, *and*
- $\forall S\, S'.\, S \Rightarrow^{N,\beta}_{\mathit{Strategy}} S' \longrightarrow S \Rightarrow^{N,\beta}_{\mathit{Reg\text{-}SCL}} S'$,

then neither is the learned clause generalized by any initial or learned clause, formally $(\nexists D \in N \cup U.\, \exists \sigma.\, D\sigma = C)$, nor is it redundant w.r.t. $N \cup U$ and the order we get by first lifting the trail-induced order $\prec^{\Gamma}$ from atoms to literals and then taking its multiset extension.

During the development of this formalization, we discovered that the original Backtrack rule found in [6] allows to learn a duplicate of the last learned clause, which violates the stated non-redundancy of learned clauses. The original Backtrack rule ensures that the conflict closure is not false under the new trail, but the learned clause could still be in conflict w.r.t. another grounding. Following this conflict, the Backtrack rules would be immediately applicable and would learn the same clause again. This could only happen a finite number of times as backtracking reduces the length of the (finite) trail. As an example, consider the set of clauses $N = \{P(x), Q(y), \neg Q(z) \vee R(z), \neg R(w) \vee S(w), \neg P(v) \vee \neg S(v)\}$, and a big enough β. The following SCL run was valid with the original Backtrack rule. Note that the notation for the trail was shortened to save space.

$$
\begin{array}{ll}
 & (\epsilon; \{\}; \top) \\
(\Rightarrow^{N,\beta}_{\text{Decide}})^* & (P(a), Q(a), P(b), Q(b); \{\}; \top) \\
(\Rightarrow^{N,\beta}_{\text{Propagate}})^* & (P(a), Q(a), P(b), Q(b), R(b)^{(R(z);\neg Q(z);z\mapsto b)}, S(b)^{(S(w);\neg R(w);w\mapsto b)}; \{\}; \top) \\
\Rightarrow^{N,\beta}_{\text{Conflict}} & (P(a), Q(a), P(b), Q(b), R(b)^{(R(z);\neg Q(z);z\mapsto b)}, S(b)^{(S(w);\neg R(w);w\mapsto b)}; \{\}; (\neg P(v) \vee \neg S(v); v\mapsto b)) \\
\Rightarrow^{N,\beta}_{\text{Resolve+Skip}} & (P(a), Q(a), P(b), Q(b), R(b)^{(R(z);\neg Q(z);z\mapsto b)}; \{\}; (\neg P(v) \vee \neg R(v); v\mapsto b)) \\
\Rightarrow^{N,\beta}_{\text{Resolve+Skip}} & (P(a), Q(a), P(b), Q(b); \{\}; (\neg P(v) \vee \neg Q(v); v \mapsto b)) \\
\Rightarrow^{N,\beta}_{\text{Backtrack}} & (P(a), Q(a), P(b); \{\neg P(v) \vee \neg Q(v)\}; \top) \\
\Rightarrow^{N,\beta}_{\text{Conflict+Skip}} & (P(a), Q(a); \{\neg P(v) \vee \neg Q(v)\}; (\neg P(v) \vee \neg Q(v); v\mapsto a)) \\
\Rightarrow^{N,\beta}_{\text{Backtrack}} & (P(a); \{\neg P(v) \vee \neg Q(v)\}; \top)
\end{array}
$$

This counterexample was only discovered when we failed to prove Theorem 11 in Isabelle. Note that this formalization is based on and was developed simultaneously to Bromberger et al., which originally inherited the Backtrack rule from [10]. The solution, which was promptly integrated into this formalization and Bromberger et al., is for the Backtrack rule to find a position without conflict w.r.t. the learned clause. Note that the original Backtrack rule reaches such a state after having learned the same clause finitely often, which has no effect on the set of learned clauses because sets ignore duplicates. Thus, the original Backtrack rule did not invalidate the other properties of the SCL calculus. This discovery is strong evidence of the usefulness of mechanized formalization for both published work and ongoing research: the Isabelle formalization lead to the discovery of a previously unknown bug and it guided the development of the refinement.

A calculus expressed as a state machine terminates if the transition relation starting from the initial state is well-founded following the arrow direction. We prove well-foundedness of regular SCL in three steps: (1) we first prove well-foundedness of SCL without backtracking, denoted $\Rightarrow^{N,\beta}_{\text{SCL-no-Back}}$; (2) we then prove that a regular run can only learn finitely many clauses; and (3) from these two results we finally prove well-foundedness of regular SCL. Step 1 is novel to the formalization. Prior work in Bromberger et al. focuses exclusively on the Backtrack rule (step 2) in order to prove termination of regular SCL (step 3). Also novel to the formalization are decreasing measuring functions for steps 1 and 2.

Definition 14. *The measuring function $\mathcal{M}_3(N, \beta, S)$ for SCL without backtracking maps a set of initial clauses N, a bounding atom β, and a state S to a 4-tuple. The tuple elements are (1) a boolean identifying whether the state is conflict-free, (2) a (finite) set overapproximating the literals that could be added to the trail, (3) a (finite) list overapproximating the numbers of resolution steps that could be performed at each position in the trail, and (4) the (finite) cardinality of the conflicting clause. Formally:*

$$\mathcal{M}_1(\beta, \Gamma) \;=\; \{L \mid \mathit{atom}(L) \preceq_B \beta\} - \{\mathit{lit}(\mathcal{K}) \mid \mathcal{K} \in \mathit{set}(\Gamma)\}$$

$$\begin{aligned}
\mathcal{M}_2(\epsilon, C) &= \epsilon \\
\mathcal{M}_2((\Gamma, K), C) &= \mathcal{M}_2(\Gamma, C), 0 \\
\mathcal{M}_2((\Gamma, (K\gamma)^{(K;D;\gamma)}), C) &= \mathit{let}\ n = \mathit{count}(C, \mathit{comp}(K\gamma))\ \mathit{in} \\
&\ \mathcal{M}_2(\Gamma, C \vee \mathit{repeat}(n, D\gamma)), n
\end{aligned}$$

$$\begin{aligned}
\mathcal{M}_3(N, \beta, (\Gamma; U; \top)) &= (\mathit{True};\ \mathcal{M}_1(\beta, \Gamma);\ \epsilon;\ 0) \\
\mathcal{M}_3(N, \beta, (\Gamma; U; (C; \gamma))) &= (\mathit{False};\ \{\};\ \mathcal{M}_2(\Gamma, C);\ |C|)
\end{aligned}$$

With this, we can prove termination of SCL without backtracking (step 1).

Theorem 15 (`termination_scl_without_back`). *SCL without backtracking is well-founded on all states reachable by an SCL-without-backtracking run starting from the initial state, formally on $\{S \mid (\epsilon; \{\}; \top)\ (\Rightarrow^{N,\beta}_{\mathit{SCL\text{-}no\text{-}Back}})^*\ S\}$.*

We now turn to proving termination of regular SCL with backtracking by first defining an appropriate measuring function.

Definition 16. *The measuring function $\mathcal{M}_4(\beta, S)$ for the rule Backtrack maps a bounding atom β and a state S to a finite set of clauses without duplicates. It computes an over-approximation of the set of clauses that could still be learned modulo duplicates. Formally:*

$$\mathcal{M}_4(\beta, S) \;=\; 2^{\{L \mid \mathit{atom}(L) \preceq_B \beta\}} - \{\mathit{set}(C) \mid C \in \mathit{gnd}(\mathit{learned}(S))\}$$

We then prove that it decreases every time we learn a new clause (step 2).

Lemma 17 ($\mathcal{M}$`_back_after_regular_backtrack`). *Following conflict resolution in a regular run, formally if*

- $(\epsilon; \{\}; \top)\ (\Rightarrow^{N,\beta}_{\mathit{Reg\text{-}SCL}})^*\ (\Gamma; U; \top)$,
- $(\Gamma; U; \top) \Rightarrow^{N,\beta}_{\mathit{Conflict}} S_1$,
- $S_1\ (\Rightarrow^{N,\beta}_{\mathit{Skip,Factorize,Resolve}})^+\ S_n$, *and*
- $S_n \Rightarrow^{N,\beta}_{\mathit{Backtrack}} S_{1+n}$, *then*

1. *the ground conflict is distinct from all groundings of initial and learned clauses modulo duplicates, formally ($\exists C\, \gamma.\ \mathit{conflict}(S_n) = (C; \gamma) \wedge \mathit{set}(C\gamma) \notin \{\mathit{set}(D) \mid D \in \mathit{gnd}(N \cup U)\}$), and*

2. *the set of clauses that could potentially be learned strictly diminishes, formally* $\mathcal{M}_4(\beta, S_{1+n}) \subset \mathcal{M}_4(\beta, S_n)$.

Lemma 17 is novel to the formalization. Together with Theorem 15 it allows us to prove termination of regular SCL with backtracking (step 3).

Theorem 18 (`termination_regular_scl`). *Regular SCL is well-founded on all states reachable by a regular-SCL run starting from the initial state, formally on* $\{S \mid (\epsilon; \{\}; \top)\, (\Rightarrow^{N,\beta}_{\mathit{Reg\text{-}SCL}})^*\, S\}$.

All termination results can be generalized to an arbitrary strategy restricting the regular strategy. We only show one example here and refer the reader to the formalization for the others.

Corollary 19 (`termination_strategy`). *If a strategy restricts regular SCL, formally if* $(\forall S\, S'.\ S \Rightarrow^{N,\beta}_{\mathit{Strategy}} S' \longrightarrow S \Rightarrow^{N,\beta}_{\mathit{Reg\text{-}SCL}} S')$, *then it is well-founded on all states reachable by a run using this strategy and starting from the initial state, formally on* $\{S \mid (\epsilon; \{\}; \top)\, (\Rightarrow^{N,\beta}_{\mathit{Strategy}})^*\, S\}$.

All theorems until now were first expressed and proven using invariants and then the versions expressed using runs were derived. However, Theorem 18 posed an interesting problem because its proof requires the backtracking step to have knowledge of the trail when a conflict last occurred. But this information is lost in the SCL state due to the Skip rule shrinking the trail. We did define an invariant that expresses the historical form of the trail and its properties derived from the regular strategy, but it is complex and the added value compared to working directly on a regular run is questionable. For simplicity, we chose not to present this invariant in this paper.

Together, soundness and termination allow us to prove refutational completeness of the regular SCL calculus w.r.t. a fixed bound.

Theorem 20 (`completeness_wrt_bound`). *If the groundings* $\mathit{gnd}^{\preceq_B \beta}(N)$ *of the initial clauses* N *are unsatisfiable, then all regular SCL runs starting from the initial state terminate and derive the conflicting clause* $\bot$, *formally*

1. *there is no infinite regular run starting from the initial state, and*
2. $(\forall S.\ (\epsilon; \{\}; \top)\, (\Rightarrow^{N,\beta}_{\mathit{Reg\text{-}SCL}})^*\, S \wedge (\nexists S'.\ S \Rightarrow^{N,\beta}_{\mathit{Reg\text{-}SCL}} S') \longrightarrow (\exists \gamma.\ \mathit{conflict}(S) = (\bot; \gamma)))$.

Theorem 20 is only defined w.r.t. a bound, but fortunately we can prove that there must always exist an appropriate bound.

Lemma 21 (`ex_bound_if_unsat`). *If the relation* $\prec_B$ *is a well-founded, strict order, total on ground atoms and the groundings* $\mathit{gnd}(N)$ *of the initial clauses* N *are unsatisfiable, then there exists a bound* β *such that the groundings* $\mathit{gnd}^{\preceq_B \beta}(N)$ *are unsatisfiable.*

Note that while Lemma 21 proves the existence of an appropriate bound, it provides no constructive way of finding one. What one can do is follow along Theorem 6 and iteratively increase a heuristically chosen bound until an appropriate one is found; if the set of initial clauses is unsatisfiable, this will terminate.

Isabelle Technicalities. Lemma 21's hypothesis that $\prec_B$ is a well-founded, total, strict order cannot be expressed as a theorem-local hypothesis. The reason is that the compactness theorem for clausal first-order logic requires terms to be an instance of the `wellorder` type class, which is not the case in the `scl_fol_calculus` locale, where the assumptions on the $\prec_B$ relation are kept minimal. Because Isabelle does not allow to instantiate a type class with a concrete type inside a locale or theorem, we define a new locale that extends `scl_fol_calculus` and adds a type class requirement on the first-order term constants. This enables the type-class system to automatically instantiate the `wellorder` type class for terms using the previously registered Knuth-Bendix order. We then instantiate the $\prec_B$ relation of `scl_fol_calculus` with the Knuth-Bendix order. This type class and locale gymnastic could be avoided if the formalization of the compactness theorem was refactored to offer a predicate-based version alongside the existing type-class-based version.

4 Conclusion

We generalized and formalized the SCL(FOL) calculus in Isabelle/HOL. The main results are formal proofs of soundness, non-redundancy of learned clauses, termination, and refutational completeness. Because the formalization was performed simultaneously to Bromberger et al., they could benefit from each other. A mechanized formalization must consider low-level details, but it is also the opportunity to identify the most import aspects of the theory and abstract over details needed in the context of an actual implementation. For example, we abstracted from the level of a state to define the Backtracking rule and replaced it with an abstract specification of the result. A level was used in all pen-an-paper presentations of the calculus in order to have a constructive way of going back to the maximal trail where the learned clause propagates. The abstraction supports investigation of several Backtrack rule versions and to base the soundness result on a version with a minimal requirement, i.e., the learned clause is no longer false with respect to the trail.

The formalization did uncover a small bug in the calculus, but also showed that its effect was very localized and naturally lead to a solution. Another benefit of the formalization is how much it supports refactoring and exploratory experimentation. When making a change to a definition or a conjecture, Isabelle immediately and exhaustively points to the parts that need to be adapted. Very often, proofs can automatically be adapted using proof automation tools such as Sledgehammer. This was invaluable to quickly try out ideas or change subtle parts of the calculus. One such example is in the Resolve rule, where the formalization first used substitution composition as found in the original calculus and latter replaced it by an abstract specification of merged grounding. This idea came from a private discussion sketching an eventual C implementation where it became clear that substitution composition would be a costly operation. We then introduced the abstract specification of merged grounding and fixed the formalization by following the mistakes reported by Isabelle.

Acknowledgments. We thank Sophie Tourret and Jasmin Blanchette for many fruitful discussions. We thank the anonymous reviewers for their constructive feedback on this paper.

References

1. Balarin, C.: Locales: a module system for mathematical theories. J. Autom. Reason. **52**(2), 123–153 (2014). https://doi.org/10.1007/s10817-013-9284-7
2. Blanchette, J.C.: Formalizing the metatheory of logical calculi and automatic provers in Isabelle/HOL (invited talk). In: Mahboubi, A., Myreen, M.O. (eds.) Proceedings of the 8th ACM SIGPLAN International Conference on Certified Programs and Proofs, CPP 2019, Cascais, Portugal, 14–15 January 2019, pp. 1–13. ACM (2019). https://doi.org/10.1145/3293880.3294087
3. Blanchette, J.C., Fleury, M., Lammich, P., Weidenbach, C.: A verified SAT solver framework with learn, forget, restart, and incrementality. J. Autom. Reason. **61**(1–4), 333–365 (2018). https://doi.org/10.1007/s10817-018-9455-7
4. Blanchette, J.C., Fleury, M., Weidenbach, C.: A verified SAT solver framework with learn, forget, restart, and incrementality. In: Olivetti, N., Tiwari, A. (eds.) IJCAR 2016. LNCS (LNAI), vol. 9706, pp. 25–44. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-40229-1_4
5. Blanchette, J.C., Tourret, S.: Extensions to the comprehensive framework for saturation theorem proving. Archive of Formal Proofs (2020). https://isa-afp.org/entries/Saturation_Framework_Extensions.html. Formal proof development
6. 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
7. 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), Haifa, Israel, 11–12, August 2022. CEUR Workshop Proceedings, vol. 3201. CEUR-WS.org (2022). http://ceur-ws.org/Vol-3201/paper5.pdf
8. Bromberger, M., Schwarz, S., Weidenbach, C.: SCL(FOL) revisited (2023). https://doi.org/10.48550/ARXIV.2302.05954
9. 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
10. 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
11. Huet, G., Oppen, D.C.: Equations and rewrite rules: a survey. Formal Language Theory, pp. 349–405 (1980)
12. Leidinger, H., Weidenbach, C.: SCL(EQ): SCL for first-order logic with equality. In: Blanchette, J., Kovács, L., Pattinson, D. (eds.) IJCAR 2022. LNCS, vol. 13385, pp. 228–247. Springer, Cham (2022). https://doi.org/10.1007/978-3-031-10769-6_14
13. Paulson, L.C., Blanchette, J.C.: Three years of experience with sledgehammer, a practical link between automatic and interactive theorem provers. In: Sutcliffe, G.,

Schulz, S., Ternovska, E. (eds.) The 8th International Workshop on the Implementation of Logics, IWIL 2010, Yogyakarta, Indonesia, 9 October 2011. EPiC Series in Computing, vol. 2, pp. 1–11. EasyChair (2010). https://doi.org/10.29007/36dt
14. 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
15. Schlichtkrull, A., Blanchette, J.C., Traytel, D., Waldmann, U.: Formalizing Bachmair and Ganzinger's ordered resolution prover. In: Galmiche, D., Schulz, S., Sebastiani, R. (eds.) IJCAR 2018. LNCS (LNAI), vol. 10900, pp. 89–107. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-94205-6_7
16. Schlichtkrull, A., Blanchette, J.C., Traytel, D., Waldmann, U.: Formalizing Bachmair and Ganzinger's ordered resolution prover. J. Autom. Reason. **64**(7), 1169–1195 (2020). https://doi.org/10.1007/s10817-020-09561-0
17. Sternagel, C., Thiemann, R.: First-order terms. Archive of Formal Proofs (2018). https://isa-afp.org/entries/First_Order_Terms.html. Formal proof development
18. Waldmann, U., Tourret, S., Robillard, S., Blanchette, J.C.: A comprehensive framework for saturation theorem proving. J. Autom. Reason. **66**(4), 499–539 (2022). https://doi.org/10.1007/s10817-022-09621-7
19. Wenzel, M.: Isabelle/Isar–a generic framework for human-readable proof documents. In: Matuszewski, R., Zalewska, A. (eds.) From Insight to Proof: Festschrift in Honour of Andrzej Trybulec, Studies in Logic, Grammar, and Rhetoric, vol. 10, no. 23. University of Białystok (2007)