SCL
Clause Learning from Simple Models

Alberto Fiori[1,2] and Christoph Weidenbach[1](✉)

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

Abstract. Several decision procedures for the Bernays-Schoenfinkel (BS) fragment of first-order logic rely on explicit model assumptions. In particular, the procedures differ in their respective model representation formalisms. We introduce a new decision procedure SCL deciding the BS fragment. SCL stands for clause learning from simple models. Simple models are solely built on ground literals. Nevertheless, we show that SCL can learn exactly the clauses other procedures learn with respect to more complex model representation formalisms. Therefore, the overhead of complex model representation formalisms is not always needed. SCL is sound and complete for full first-order logic without equality.

1 Introduction

There has been intensive research into the development of decision procedures for the Bernays-Schoenfinkel (BS) first-order clause fragment without equality [1,3,4,7,9,14,17]. Even classical tableau can be turned into a decision procedure for BS [2]. The procedures follow three different paradigms. They either employ an explicit CDCL-style [12] partial model assumption [1,3,4,14], or they implement an abstraction-refinement approach [9,17], or merely rely on syntactic restrictions on inferences [7] yielding finite saturations.

The BS fragment is a natural generalization of propositional logic but still enjoys the finite-model property. Furthermore, any finite BS clause set can be transformed into a satisfiability equivalent SAT problem by finite instantiation at the price of a, worst case, exponentially larger clause set. For example, this relationship is used as a reasoning principle in Answer Set Programming (ASP) [8]. The exponential "overhead" is, in the worst case, unavoidable for any decision procedure, because BS satisfiability is NEXPTIME-complete [11,15]. This means, worst case, that an explicit model representation gets exponentially large, or satisfiability testing with respect to the model representation cannot be done in polynomial time. This justifies and motivates the research for procedures with different model representation formalisms as well as alternative approaches through abstraction or saturation. Actually, the leading systems at recent CASCs [16] have implemented a portfolio containing procedures from all of the aforementioned paradigms.

P. Fontaine (Ed.): CADE 2019, LNAI 11716, pp. 233–249, 2019.
https://doi.org/10.1007/978-3-030-29436-6_14

One contribution of this paper is a CDCL-style calculus deciding the BS fragment and being sound and complete for full first-order logic without equality. The model representation is simple: it consists of a sequence of ground literals. It is therefore properly contained in known model representation formalisms [1, 3,4,14]. However, we show that this model representation formalism together with the respective inference rules is sufficient to learn the very same clauses as in NRCL [1]. NRCL has one of the most expressive model representation formalisms. We call the procedure *SCL* for *clause learning from simple models*. The most important computations with respect to a model representation are the consistent extension of the current trail and the detection of a propagating literal or a false clause. The currently available procedures can be further divided into procedures where these computations can be done in polynomial time [3,6] and procedures where such computations are worst case NP-complete [1,14].[1] The advantages of the latter two formalisms are exponentially more compact model representations where the model representation language of the NRCL calculus [1] is more general than [3,6,14]. One contribution of this paper is that for model-driven clause learning, sophisticated model representations are not needed (Theorem 24). More precisely, we prove that any clause learned by the NRCL calculus can also be learned by our new SCL calculus, where the model representation consists of ground literals only. One of the simplest but also most efficient model representations known with respect to computations. This result holds for full first-order logic without equality. Model representations for BS clause sets with respect to ground literals can become exponentially larger compared to the above-mentioned more sophisticated model representations. The model size is exponential in the maximal arity of a predicate, which we will discuss in detail in Sect. 5. The implication of our result is that SCL can be efficiently used on problems where the ground literal model representation does not become “too large”, further discussed in Sect. 6.

Another contribution in addition to SCL being sound, complete, and a decision procedure for the BS fragment is the fact that it only learns non-redundant clauses with respect to so-called *reasonable* strategies, see Sect. 3. A clause is *redundant* with respect to a clause set, if it is implied by smaller clauses, see Sect. 2. Non-redundancy is a powerful property: in the BS context, we prove it NEXPTIME-complete, Theorem 14. Practically, this implies that a clause generated by SCL with a reasonable strategy does not need to be tested for forward redundancy, e.g., forward Subsumption. Saturation-based theorem provers spent a substantial share of their run time on testing forward redundancy.

A third contribution concerning SCL is its ability to simulate resolution, Sect. 4, Theorem 20. Arbitrary resolution steps may generate redundant clauses, hence giving up a reasonable strategy is a prerequisite for the simulation. In this context we also discuss the performance of SCL with respect to proof length, Sect. 4, following [13].

Finally, we investigate a so called *weakly-reasonable* strategy, where propagations need not to be exhaustive, specifically unit clauses need not to be

[1] For [4] no complexity result has been published so far.

propagated. Although propagating unit clauses is typically a good strategy for SAT, for the BS fragment this depends already on the actual problem, because one unit clause may cause exponentially many subsequent propagation steps. In summary, the weakly-reasonable strategy generates non-redundant clauses with the exception of unit instances, Theorem 12, and allows for exponentially shorter proofs compared to a reasonable strategy, which exhausts propagation, Example 9. We end the paper with a short summary and discussion of the obtained results, Sect. 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. The complement of a literal is denoted by the function comp. 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 $\operatorname{dom}(\sigma) := \{x \mid x\sigma \neq x\}$ is finite and $\operatorname{codom}(\sigma) := \{t \mid x\sigma = t, x \in \operatorname{dom}(\sigma)\}$. 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 $\operatorname{codom}(\sigma)$ 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. A *closure* is denoted as $C \cdot \sigma$ and is a pair of a clause C and a ground substitution σ. The function gnd computes the set of all ground instances of a literal, clause, or clause set. Note that for BS this set is always finite, whereas for first-order logic it is infinite, in general. 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.

In addition, we assume a well-founded, total, strict ordering $\prec$ 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 $C' \in \operatorname{gnd}(C)$ C' is redundant with respect to $\cup_{D \in N} \operatorname{gnd}(D)$.*

3 SCL Rules and Properties

The inference rules of SCL are represented by an abstract rewrite system. They operate on a problem state, a five-tuple $(\Gamma; N; U; k; u)$ where Γ is a sequence of annotated ground literals, the *trail*; N and U are the sets of *initial* and *learned* clauses; k counts the number of decisions; and u is a status that is either true $\top$, false $\bot$, or a closure $C \cdot \sigma$. Literals in Γ are either annotated with a number, 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 $(\Gamma; N; U; k; u)$ if L or $\operatorname{comp}(L)$ occurs in Γ and the first decision literal left from L ($\operatorname{comp}(L)$) 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 $(\Gamma; N; U; k; u)$ if i is the maximal level of a literal in D; the level of the empty clause $\bot$ is 0. Recall u is a non-empty closure or $\top$ or $\bot$.

A literal L is *undefined* in Γ if neither L nor $\operatorname{comp}(L)$ occur in Γ. The initial state for a first-order clause set N is $(\epsilon, N, \emptyset, 0, \top)$. The rules for conflict search are

Propagate $(\Gamma; N; U; k; \top) \Rightarrow_{\text{SCL}} (\Gamma, L\sigma^{(C \vee L)\cdot\sigma}; N; U; k; \top)$
provided $C \vee L \in (N \cup U)$, $C\sigma$ is ground and false under Γ, $L\sigma$ is undefined in Γ

Decide $(\Gamma; N; U; k; \top) \Rightarrow_{\text{SCL}} (\Gamma, L^{k+1}; N; U; k+1; \top)$
provided L is a ground literal undefined in Γ

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

These rules construct a (partial) model via the trail Γ for $N \cup U$ until a conflict, i.e., a false clause with respect to Γ is found. The above rules always terminate with respect to the BS fragment, but not for first-order logic, in general. In the special case of a unit clause L, the rule Propagate actually annotates the literal L with a closure of itself. So the propagated literals on the trail are annotated with the respective propagating clause and the decision literals with the respective level. If a conflict is found, it is resolved by the rules below. Before any Resolve step, we assume that the respective clauses are renamed such that they do not share any variables and that the grounding substitutions of closures are adjusted accordingly.

Skip $(\Gamma, L\delta^{(C \vee L)\cdot\delta}; N; U; k; D \cdot \sigma) \Rightarrow_{\text{SCL}} (\Gamma; N; U; k; D \cdot \sigma)$
provided $\operatorname{comp}(L\delta)$ does not occur in $D\sigma$

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

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

Backtrack $(\Gamma, K^{i+1}, \Gamma'; N; U; k; (D \lor L) \cdot \sigma) \ \Rightarrow_{\text{SCL}} \ (\Gamma; N; U \cup \{D \lor L\}; i; \top)$ provided $L\sigma$ is of level k and $D\sigma$ is of level i.

The clause $D \lor L$ 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. The rules for SCL are applied in a don't-care style, hence, the calculus offers freedom with respect to factorization. Literals in the conflict clause can, but do not have to be factorized. In particular, the Factorize rule may remove duplicate literals. The rule Resolve does not remove the literal resolved upon from the trail. Actually, Resolve is applied as long as the rightmost propagated trail literal occurs in the conflict clause. This literal is eventually removed by rule Skip from the trail.

For example, consider the clause set $N = \{D = Q \lor R(a, y) \lor R(x, b), C = Q \lor S(x, y) \lor P(x) \lor P(y) \lor \neg R(x, y)\}$ and a problem state:

$$(\neg P(a)^1, \neg P(b)^2, \neg S(a, b)^3, \neg Q^4, \neg R(a, b)^{C \cdot \{x \mapsto a, y \mapsto b\}}, N, \emptyset, 4, \top)$$

derived by SCL. The rule Conflict is applicable and yields the conflict state

$$(\neg P(a)^1, \neg P(b)^2, \neg S(a, b)^3, \neg Q^4, R(a, b)^{C \cdot \{x \mapsto a, y \mapsto b\}}; N; \emptyset; 4; D \cdot \{x \mapsto a, y \mapsto b\})$$

from which we can either learn the clause

$$C_1 = Q \lor S(x, b) \lor P(x) \lor P(b) \lor S(a, y) \lor P(a) \lor P(y)$$

or the clause

$$C_2 = Q \lor S(a, b) \lor P(a) \lor P(b)$$

depending on whether we first resolve or factorize. Note that C_2 does not subsume C_1. Both clauses are non-redundant. In order to learn C_1 we need to resolve twice with $R(a, b)^{C \cdot \{x \mapsto a, y \mapsto b\}}$.

The first property we prove about SCL is soundness. We prove it via the notion of a sound state.

Definition 2 (Sound States). *A state* $(\Gamma; N; U; k; u)$ *is* sound *if the following conditions hold*

1. *Γ is a consistent sequence of annotated ground literals,*
2. *for each decomposition $\Gamma = \Gamma_1, L\sigma^{C \lor L \cdot \sigma}, \Gamma_2$ we have that $C\sigma$ is false under Γ_1 and $L\sigma$ is undefined under Γ_1, $C \lor L \in (N \cup U)$,*
3. *for each decomposition $\Gamma = \Gamma_1, L^k, \Gamma_2$ we have that L is undefined in Γ_1,*
4. *$N \models U$,*
5. *if $u = C \cdot \sigma$ then $C\sigma$ is false under Γ and $N \models C$.*

Note that an initial state $(\epsilon, N, \emptyset, 0, \top)$ is sound. A rule is *sound* if it maps sound states to sound states.

Theorem 3 (Soundness of SCL). *The rules of SCL are sound, hence SCL starting with an initial state is sound.*

Proof. (Idea) By induction on the length of an SCL derivation and a case analysis for the different rules preserving soundness of states. □

Next we introduce regular and weakly-regular runs. Regular runs always generate non-redundant clauses, but require exhaustive propagation. Weakly-regular runs do not require exhaustive propagation and almost always generate non-redundant clauses except for instances of unit clauses. However, although exhaustive propagation is typically done in CDCL style SAT, already for the BS fragment it should not always be preferred, because unit clauses can have already exponentially many ground instances.

Definition 4 (Regular States). *A state $(\Gamma; N; U; k; u)$ is* regular *if and only if the following hold:*

1. *for every decomposition $\Gamma = \Gamma_1, L^k, \Gamma_2$ there is no clause in $N \cup U$ that could propagate from Γ_1,*
2. *for each decomposition $\Gamma = \Gamma_1, L, \Gamma_2$ where L may be either propagated or decided, there is no clause from $\operatorname{gnd}(N \cup U)$ false under Γ_1.*

Definition 5 (Weakly-Regular States). *A state $(\Gamma; N; U; k; u)$ is* weakly-regular *if and only if the following hold:*

1. *for every decomposition $\Gamma = \Gamma_1, L^k, \Gamma_2$ there is no non-unit clause in $N \cup U$ that could propagate from Γ_1,*
2. *for each decomposition $\Gamma = \Gamma_1, L, \Gamma_2$ where L may be either propagated or decided, there is no clause from $\operatorname{gnd}(N \cup U)$ false under Γ_1.*

Some of the below results hold both for regular and weakly-regular states or runs. In this case we write "(weakly-) regular" meaning both cases.

Theorem 6 (Correct Termination). *If no rules are applicable to a (weakly-) regular state $(\Gamma; N; U; k; u)$ then either $u = \bot$ and N is unsatisfiable or N is satisfiable and $\Gamma \models N$.*

Proof. For a state $(\Gamma; N; U; k; u)$ where $u \notin \{\top, \bot\}$, one of the rules Resolve, Skip, Factorize or Backtrack is applicable. If the top level literal is a propagated literal then either Resolve or Skip are applicable. If the top level literal is a decision then one of the rules Backtrack or Factorize is applicable. If $u = \top$ and Propagate, Decide, and Conflict are not applicable it means that there are no undefined ground literals in Γ, so $\Gamma \models N$. □

Definition 7 (Regular Runs). *A derivation of regular states is* regular *or a* regular run *if the rules Conflict and Propagate are always applied before all other rules in decreasing order of priority.*

Definition 8 (Weakly-Regular Runs). *A derivation of regular states is* weakly-regular *or a* weakly-regular run *if the following conditions hold:*

1. Conflict has higher priority than all other rules,
2. if Conflict is not applicable and we can apply Propagate to a non-unit clause then Propagate has higher priority than any other rule,
3. Decide never adds a literal L to the trail if $\text{comp}(L)$ *is a unit clause in* $\text{gnd}(N \cup U)$*,*
4. Resolve has higher priority than Backtrack if the current conflict clause is subsumed in N by a unit clause.

Example 9 (Comparing Proof Length of Regular and Weakly-Regular Runs). Proofs generated by weakly-regular runs can be exponentially shorter than proofs generated by regular runs. Consider the simple BS clause set

$$N = \{R(x_1, \ldots, x_n, a, b), P \lor Q, P \lor \neg Q, \neg P \lor Q, \neg P \lor \neg Q\}.$$

A weakly-regular run can ignore generating the 2^n different ground instances of $R(x_1, \ldots, x_n, a, b)$ and directly proceed in refuting the propositional part of N in the usual CDCL style by starting with a decision on P or Q. For the example it is obvious that the instances of $R(x_1, \ldots, x_n, a, b)$ can be ignored, but in general it is not. This phenomenon already occurs for NP-complete problems: when deciding linear integer arithmetic in a CDCL style, exhaustive propagation is not required by respective calculi for the very same reason [5].

Definition 10 (State Induced Ordering). *Let $(L_1, L_2, \ldots, L_n; N; U; k; u)$ be a sound state of SCL where the annotations of the L_i are ignored. The trail induces a total well-founded strict order on the defined literals by*

$$L_1 \prec_\Gamma \text{comp}(L_1) \prec_\Gamma L_2 \prec_\Gamma \text{comp}(L_2) \prec_\Gamma \cdots \prec_\Gamma L_n \prec_\Gamma \text{comp}(L_n).$$

We extend $\prec_\Gamma$ to a strict total order on all literals where all undefined literals are larger than $\text{comp}(L_n)$*. We also extend $\prec_\Gamma$ to a strict total order on ground clauses by multiset extension and also on multisets of ground clauses and overload $\prec_\Gamma$ for all these cases. With $\preceq_\Gamma$ we denote the reflexive closure of $\prec_\Gamma$.*

Theorem 11 (Learned Clauses in Regular Runs). *Let $(\Gamma; N; U; k; C_0 \cdot \sigma_0)$ be the state resulting from the application of Conflict in a regular run and let C be the clause learned at the end of the conflict resolution, then C is not redundant with respect to $N \cup U$ and $\prec_\Gamma$.*

Proof. Consider the following fragment of a derivation learning a clause:

$$\Rightarrow^{\text{Conflict}}_{\text{SCL}} (\Gamma; N; U; k; C_0 \cdot \sigma_0) \Rightarrow^{\{\text{Skip, Fact., Res.}\}^*}_{\text{SCL}} (\Gamma'; N; U; k; C \cdot \sigma) \Rightarrow^{\text{Backtrack}}_{\text{SCL}} .$$

By soundness $N \cup \models C$ and $C\sigma$ is false under both Γ and Γ'. We prove that $C\sigma$ is non-redundant.

Assume there is an $S \subseteq \text{gnd}(N \cup U)^{\preceq_\Gamma C\sigma}$ s.t. $S \models C\sigma$. There is a clause $D \in S$ false under Γ, $S \preceq_\Gamma \{C\sigma\}$ and $C\sigma \notin S$. All clauses in S have a defined truth value (as all undefined literals are greater than all defined literals) and if $\Gamma \models S$ then $\Gamma \models C\sigma$, a contradiction.

We distinguish whether the two trails Γ and Γ' are equal or Γ' is a strict prefix of Γ.

If $\Gamma \neq \Gamma'$ then at least one Skip application was performed during conflict resolution, so $C\sigma$ does not contain the rightmost literal of Γ and since $D \prec_\Gamma C\sigma$ neither does D. So at a previous point in the derivation there must be a conflict search state such that D was false under the current trail but was not chosen as conflict instance, a contradiction to the exhaustive application of Conflict.

If $\Gamma = \Gamma'$ we distinguish two sub-cases according to whether the rightmost literal in Γ is the result of a Decision or a Propagation.

If the rightmost literal of $\Gamma = \Gamma'', L^k$ is a decision literal, then D is either true in Γ'' or has at least two literals undefined or of level k. Since D must be false under Γ, D must have two or more occurrences of literals undefined under Γ or of $\text{comp}(L)$. Since $C\sigma$ has no undefined literal and exactly one occurrence of $\text{comp}(L)$ we have a contradiction with $D \prec_\Gamma C\sigma$.

If $\Gamma = \Gamma'', L^{C'\cdot\delta}$ then at most one literal in $C\sigma$ is of level k and all other literals, if any, are of level at most $k-1$. Moreover, D is also either true in Γ'' or has at least two literals undefined under Γ'' or of level k or $k=0$. Backtrack requires the presence of at least one decision literal on the trail and so $k>0$ so D must have at least two literals of level k. If both of those literal are different from $\text{comp}(L)$ then by regularity, we would have applied Conflict instead of Propagate on the trail Γ''. So at least one of the literals of level k in D must be $\text{comp}(L)$. Simple case analysis shows that under these conditions $C\sigma \prec_\Gamma D$, a contradiction. □

Theorem 12 (Learned Clauses in Weakly-Regular Runs). *Let the state $(\Gamma; N; U; k; C_0 \cdot \sigma_0)$ the result of a Conflict application in a weakly-regular run and let C be the clause learned at the end of the conflict resolution, then C is not redundant w.r.t. $N \cup U$ and $\prec_\Gamma$ or C is an instantiation of a unit clause in $N \cup U$.*

Proof. Consider the following fragment of a derivation learning a clause:

$$\Rightarrow_{\text{SCL}}^{\text{Conflict}} (\Gamma; N; U; k; C_0 \cdot \sigma_0) \Rightarrow_{\text{SCL}}^{\{\text{Skip, Fact., Res.}\}^*} (\Gamma'; N; U; k; C \cdot \sigma) \Rightarrow_{\text{SCL}}^{\text{Backtrack}} .$$

By soundness $N \cup U \models C$ and $C\sigma$ is false under both Γ and Γ'. We need to prove that there exists a ground instantiation of C that is non-redundant or that C is unit; we assume that C is not a unit and prove by contradiction that $C\sigma$ is non-redundant.

Assume there is an $S \subseteq \text{gnd}(N \cup U)^{\preceq_\Gamma C\sigma}$ s.t. $S \models C$. There is a clause $D \in S$ false under Γ; indeed since $S \prec_\Gamma \{C\sigma\}$ all clauses in S have a defined truth value and if all clauses in S were to be true under Γ we would also have that $C\sigma$ would be true under Γ by transitivity of entailment.

We distinguish whether the two trails Γ and Γ' are equal or Γ' is a strict prefix of Γ.

If $\Gamma \neq \Gamma'$ then rule Skip was applied at least once during conflict resolution, so $C\sigma$ does not contain the rightmost literal of Γ and since $D \prec_\Gamma C\sigma$ neither does D. So at a previous point in the derivation there must be a conflict search state such that D was false under the current trail but was not chosen as conflict instance, a contradiction to the exhaustive application of Conflict.

If $\Gamma = \Gamma'$ we distinguish two sub-cases according to whether the rightmost literal in Γ is the result of a Decision or a Propagation.

If the rightmost literal of $\Gamma = \Gamma'', L^k$ is a decision literal then D is either true in Γ'', a unit clause, or has at least two literals undefined or of level k. If D is a unit clause then it must be true or undefined in Γ'', and since decisions are restricted to never falsify unit clauses D cannot be false in Γ''. Otherwise, D must have two or more occurrences of literals undefined under Γ'' or of $\text{comp}(L)$. Since $C\sigma$ has no undefined literal and exactly one occurrence of $\text{comp}(L)$, we have a contradiction with $D \prec_\Gamma C\sigma$.

If $\Gamma = \Gamma'', L^{C'\cdot\delta}$ then at most one literal in $C\sigma$ is of level k and all other literals, if any, are of level at most $k-1$, moreover D is either true in Γ'', a unit clause, or has at least two literals undefined or of level k. Under these conditions $D \prec_\Gamma C\sigma$ and $\Gamma \models \neg D$ imply $D = \neg L$ and $C\sigma = C'' \lor \neg L$, but then the next rule cannot be a Backtrack as by weak regularity Resolve would have higher priority. □

Proposition 13. *All regular runs are weakly-regular runs.*

Proof. There are four conditions that a weakly-regular run needs to satisfy. A regular run clearly respects the first two conditions, so we prove that regular runs respect the last two. In a regular run Propagate has always higher priority than Decide, so whenever we could decide a literal L s.t. $\text{comp}(L) \in \text{gnd}(N \cup U)$ the literal $\text{comp}(L)$ must have already been propagated and so L is not undefined. By Theorem 11 whenever we can apply Backtrack the conflict clause is not redundant in N and so not subsumed in N. □

Theorem 14 (BS Non-redundancy is NEXPTIME-Complete). *Deciding non-redundancy of a BS clause C with respect to a finite BS clause set $N^{\preceq C}$ is NEXPTIME-Complete.*

Proof. We only show hardness, because containment of the problem in NEXPTIME is obvious. To this end, let $N = \{C_1, \dots, C_n\}$ be an arbitrary, finite BS clause set. We consider an LPO ordering $\prec_{\text{LPO}}$. Next we add two fresh predicates of arity zero, P, Q with $P \prec_{\text{LPO}} Q$, where P, Q are larger in the LPO precedence than any other symbol from N. Then obviously N is satisfiable iff the finite BS clause set $N' = \{P, Q, C_1 \lor \neg Q, C_2 \lor \neg P, C_3, \dots, C_n\}$ is satisfiable. Furthermore, the clause $\neg P \lor \neg Q$ is $\prec_{\text{LPO}}$ larger than any clause in $N' \setminus \{P, Q\}$. The clause $\neg P \lor \neg Q$ is non-redundant with respect to $N' \setminus \{P, Q\}$ iff N' is satisfiable. □

Theorem 15 (Termination). *If N is a clause set and $\text{gnd}(N)$ is finite then any (weakly-) regular run of SCL terminates.*

Proof. Any infinite run learns infinitely many clauses. Firstly, for a regular run, by Theorem 11, all learned clauses are non-redundant. The number of different ground clauses and literals is finite. So there is no infinite regular run. Secondly, for weakly-regular runs, the learned clause is either non-redundant, or an instance of a unit clause from N or from the set of learned clauses. However, there are also only finitely many instances of unit clauses from N. So there is no infinite weakly-regular run. □

Theorem 16 (SCL Refutational Completeness). *If N is unsatisfiable, then there is a (weakly-) regular run of SCL deriving $\bot$.*

Proof. If N is unsatisfiable, then as a consequence of Herbrand's Theorem there is a finite set of ground instances N' from N that is unsatisfiable. Now restrict the rules Decide and Propagate to ground literals from N'. By Theorems 15 and 6 any (weakly-) regular run on this restriction derives $\bot$. □

Theorem 17 (SCL decides the BS fragment). *SCL restricted to weakly-regular runs decides satisfiability of a BS clause set.*

Proof. There are only finitely many ground instances of a BS clause set. Following the proof of Theorem 15, any SCL (weakly-) regular run will terminate on a BS clause set. □

4 Simulating Resolution by SCL

It is well-known that resolution inferences may generate redundant clauses. Therefore, by Theorems 11 and 12 (weakly-) regular runs cannot simulate resolution. However, the SCL calculus is still flexible enough to simulate arbitrary resolution inferences that do not result in tautologies by a non-regular strategy.

Lemma 18 (SCL Simulates Resolution). *Let N be a clause set, $C = C' \lor H$ and $D = D' \lor H'$ be clauses in N such that C' and D' are non-empty, $\eta = \mathrm{mgu}(H, \mathrm{comp}(H'))$ exists, the literals H and H' are not duplicated in either C or D and the three clauses $C\eta$, $D\eta$ and $(C' \lor D')\eta$ are not tautologies. Then, there is an SCL run starting from the state $(\epsilon; N; \emptyset; 0; \top)$ where the first learned clause is $(C' \lor D')\eta$.*

Proof. Let θ be a grounding substitution on the variables of $C\eta$ and $D\eta$ such that distinct literals are mapped to distinct ground literals, also let $\{L_1, \ldots, L_n\} = (C' \lor D')\eta\theta$.

We can start our derivation with n Decisions where at the i-th step we decide the literal $\mathrm{comp}(L_i)$ to obtain the state $(K_1^1, \ldots, K_n^n; N; U; n; \top)$, $K_i = \mathrm{comp}(L_i)$. This is possible as $(C' \lor D')$ is not a tautology.

$H\eta\theta$ is still undefined but every other literal in $C'\eta\theta$ is falsified under the current trail; so in the next step we can propagate $H\eta\theta$ on the trail.

$$(K_1^1, \ldots, K_n^n; N; \emptyset; n; \top) \Rightarrow_{\mathrm{SCL}}^{\mathrm{Propagate}} (K_1^1, \ldots, K_n^n, H\eta\theta^{C' \lor H \cdot \eta\theta}; N; \emptyset; n; \top)$$

Now the rule Conflict is applicable to $(D' \lor H')\eta\theta$ resulting in

$$(K_1^1, \ldots, K_n^n, H\eta\theta^{C' \lor H \cdot \eta\theta}; N; \emptyset; n; D' \lor H' \cdot \eta\theta).$$

We apply Resolve once and we reach the state

$$(K_1^1, \ldots, K_n^n; N; \emptyset; n; (C' \vee D')\eta \cdot \theta).$$

From this state we can backtrack to

$$(K_1^1, \ldots, K_k^k; N; \{(C' \vee D')\eta\}; k; \top).$$

□

Lemma 19 (SCL Simulates Factoring). *Let N be a clause set, $C = C' \vee H \vee H'$ a non-tautological clause with unifiable literals H and H'. There is a run starting from the state $(\epsilon; N; \emptyset; 0; \top)$ where the first learned clause is $(C' \vee H)\eta$ where $\eta = \operatorname{mgu}(H, K)$.*

Proof. Let $\{L_1, \ldots, L_k\} = C\eta$ and let ρ be a grounding of C injective on the literals of $C\eta$. By applying Decide k times we can reach a state $(K_1^1\rho, \ldots, K_k^k\rho; N; \emptyset; k; \top)$, $K_i = \operatorname{comp}(L_i)$, from which we can apply rule Conflict to $C\eta\rho$ resulting in $(K_1^1\rho, \ldots, K_k^k\rho; N; \emptyset; k; C \cdot \eta\rho)$. Now we can factorize H and H', deriving $\Rightarrow_{\mathrm{SCL}}^{\mathrm{Factorize}}$ $(K_1^1\rho, \ldots, K_k^k\rho; N; \emptyset; k; (C' \vee H)\eta \cdot \rho)$ and, finally, backtrack and learn $(C' \vee H)\eta$. □

In order to simulate an overall refutation of the resolution calculus, an additional Restart rule is needed. Note that tautologies can be ignored in refutations by the resolution calculus.

Restart $(\Gamma; N; U; k; \top) \Rightarrow_{\mathrm{SCL}} (\epsilon; N; U; 0; \top)$
provided $\Gamma \not\models N$

Theorem 20 (SCL Simulates the Resolution Calculus). *SCL together with the Restart rule can simulate a resolution refutation by a non-regular strategy.*

Proof. By Lemmas 18 and 19. □

Consider another example, taken from [13], where exhaustive propagation leads to exponentially longer proofs compared to the shortest resolution proof. Let i be a positive integer and consider the clause set N^i with one predicate P of arity i consisting of the following clauses, where we write $\bar{x}, \bar{0}$ and $\bar{1}$ to denote sequences of the appropriate length of variables and constants to meet the arity of P:

$$P(\bar{0}) \qquad \neg P(\bar{1})$$

and i clauses of the form

$$\neg P(\bar{x}, 0, \bar{1}) \vee P(\bar{x}, 1, \bar{0})$$

where the length of $\bar{1}$ varies between 0 and $i - 1$. The example encodes an i-bit counter. A regular run of SCL (NRCL) on this clause set would find a conflict after $O(2^i)$ propagations without any application of Decide. For example, for $i = 4$ we get the clauses of N^4:

$$\begin{array}{l}
1 : P(0,0,0,0) \\
2 : \neg P(x_1,x_2,x_3,0) \lor P(x_1,x_2,x_3,1) \\
3 : \neg P(x_1,x_2,0,1) \lor P(x_1,x_2,1,0) \\
4 : \neg P(x_1,0,1,1) \lor P(x_1,1,0,0) \\
5 : \neg P(0,1,1,1) \lor P(1,0,0,0) \\
6 : \neg P(1,1,1,1)
\end{array}$$

For this clause set a regular SCL (NRCL) generates all unit clauses from $P(0,0,0,0)$ to $P(1,1,1,1)$ via 2^4 applications of Propagate, then finds a conflict with clause 6 and then uses 2^4 times Resolve to end up in $\bot$.

Instead a short resolution refutation can be obtained by

$$\begin{array}{ll}
\text{2.2 Res 3.1} & 7 : \neg P(x_1,x_2,0,0) \lor P(x_1,x_2,1,0) \\
\text{7.2 Res 2.1} & 8 : \neg P(x_1,x_2,0,0) \lor P(x_1,x_2,1,1) \\
\text{8.2 Res 4.1} & 9 : \neg P(x_1,0,0,0) \lor P(x_1,1,0,0) \\
\text{9.2 Res 8.1} & 10 : \neg P(x_1,0,0,0) \lor P(x_1,1,1,1) \\
\text{10.2 Res 5.1} & 11 : \neg P(0,0,0,0) \lor P(1,0,0,0) \\
\text{11.2 Res 10.1} & 12 : \neg P(0,0,0,0) \lor P(1,1,1,1) \\
\text{12.1 Res 6.1} & 13 : \bot
\end{array}$$

In general, $O(2i)$ many resolution steps are sufficient to refute N^i. The above resolution proof cannot be simulated by an SCL weakly-regular run. As soon as we decide a ground literal $[\neg]P(\ldots)$ propagation using the two literal clauses yields a conflict with either $\neg P(1,1,1,1)$ or $P(0,0,0,0)$. The above resolution proof can only be simulated by a non-regular SCL run, Theorem 20. It is an open problem to find a notion of regularity that both guarantees non-redundant clause learning and can simulate resolution proofs of the above type.

5 Simulating NRCL by SCL

In this section we show that even under the restriction of a ground trail, SCL can generate any clause learned by NRCL. In the worst case, the trail generated by SCL may be exponentially longer than the NRCL trail, see Example 25. We use $\Gamma, \Gamma_1, \Gamma_2$ to denote SCL trails and $\Gamma', \Gamma'_1, \Gamma'_2$ to denote NRCL trails.

The ordering $\prec_{\Gamma'}$ is defined as in [1] and concerning NRCL we exactly stick to the notions from [1]. Nevertheless, the most important notions from NRCL are recalled below.

Definition 21 (Constrained Clauses [1]). *A constrained clause $(C \cdot \sigma; \pi)$ is a pair of a closure $C \cdot \sigma$ and a constraint π of the form $\bigwedge_{i=0}^{n} \vec{s}_i \neq \vec{t}_i$ where $\vec{s}_i$ and $\vec{t}_i$ are tuples of terms of equal length. The set of ground instances of $(C \cdot \sigma; \pi)$ denoted as* $\text{gnd}(C \cdot \sigma; \pi)$ *is* $\{C\sigma\delta \mid C\sigma\delta \in \text{gnd}(C\sigma)$ *and* $\pi\delta$ *is true*$\}$. *A constraint $\pi = \bigwedge_{i=0}^{n} \vec{s}_i \neq \vec{t}_i$ is true if for all $0 \leq i \leq n$ $\vec{s}_i$ and $\vec{t}_i$ are not unifiable. A ground* *clause C' is covered by a constrained clause $(C \cdot \sigma; \pi)$ if $C' \in \operatorname{gnd}(C \cdot \sigma; \pi)$. We similarly define constrained literals $(L \cdot \sigma; \pi)$ and say that a ground literal L' is defined true by $(L \cdot \sigma; \pi)$ if it is covered by $(L \cdot \sigma; \pi)$ and defined false if it covered by $(\operatorname{comp}(L) \cdot \sigma; \pi)$.*

Definition 22 (State induced ordering in NRCL [1]). *A ground literal L is defined under a trail Γ' if L or $\operatorname{comp}(L)$ is covered by some constrained literals L' in Γ'. Such a literal is necessarily unique and is denoted by $\operatorname{def}(L)$. Given two defined ground literals L_1, L_2 we say $L_1 \prec_{\Gamma'} L_2$ if $\Gamma' = \Gamma'_0, \operatorname{def}(L_1), \Gamma'_1, \operatorname{def}(L_2), \Gamma'_2$.*

The proof of the simulation is done in two steps. Firstly, we show that we can generate via SCL a suitable ground instance of any NRCL trail, Lemma 23. Then we show that on this basis, we can actually learn exactly the clause NRCL learns, Theorem 24.

Lemma 23 (Simulating the NRCL Trail). *Let $(\Gamma'; N; U; k; \top)$ be a regular state in an NRCL run and let $\prec_0$ be a total order on ground literals compatible with $\prec_{\Gamma'}$. Then there is an SCL derivation starting from $(\epsilon; N; U; 0; \top)$ which produces a trail Γ such that for any ground literal L, L is true, false, undefined under Γ if and only if it is so under Γ', and for all ground literals $L_1, L_2 \in \operatorname{gnd}(\Gamma')$: $L_1 \prec_\Gamma L_2$ if and only if $L_1 \prec_0 L_2$. We call Γ a* grounding *of Γ'.*

Proof. By induction on the length of Γ'. If $\Gamma' = \epsilon$ then we choose $\Gamma = \epsilon$ satisfying the conjecture. If $\Gamma' = \Gamma'_1, (L; \sigma; \pi)^k$ then all ground literals $\operatorname{gnd}(L; \sigma; \pi)$ are undefined in Γ_1 so we can simply decide every literal in $\operatorname{gnd}(L; \sigma; \pi)$ in increasing order according to $\prec_{\Gamma'}$ to obtain a trail Γ_2 that clearly satisfies the conjecture. If the rightmost literal of Γ' is a propagation then we apply Propagate instead of Decide on the SCL trail. □

A consequence of Lemma 23 is that SCL can simulate the derivation of $\bot$ from a state without decisions. So this needs not to be considered anymore. The most sophisticated rule of NRCL to be considered for the simulation is Backjump, because its side conditions are substantially different from the side conditions of the SCL Backtrack rule.

Backjump $(\Gamma, K^{k'}, \Gamma'; N; U; k; (C \cdot \sigma; \pi)) \;\Rightarrow_{\text{NRCL}}\; (\Gamma, K^{k'}; N; U \cup \{C\}; k'; \top)$

provided one of the following conditions hold (i) $k = 0$ and $C = \bot$, (ii) $k > 0$, all ground clauses covered by $(C\sigma; \pi)$ have exactly one literal of level k and $(C\sigma; \pi)$ has no false instances under Γ (iii) $k > 0$, the right-most element of Γ' is a decision, some ground clauses covered by $(C\sigma; \pi)$ have two or more literals of level k, $(C\sigma; \pi)$ has no false instances under Γ and Factorize cannot be applied.

Theorem 24 (SCL Simulates NRCL). *If from an NRCL conflict state we can learn a clause $C \neq \bot$, then we can learn the same clause C from a grounding of that state by SCL.*

Proof. Let $w'_1 = (\Gamma'_1; N; U; k; (C_1; \sigma_1; \pi_1))$ be an NRCL conflict state from which we learn the clause C. We prove by induction on the length of conflict resolution that there exists an SCL state $w_1 = (\Gamma_1; N; U; k; C_1 \cdot \delta_1)$ obtained by grounding w'_1 from which we can learn C.

As a base case, we prove that if we can apply Backjump to an NRCL state $(\Gamma'_2; N; U; k'; (C_2; \sigma_2; \pi_2))$ then we can also apply Backtrack to a grounding $(\Gamma_2; N; U; k; C_2 \cdot \delta_2)$. In particular, we choose $\delta_2 = \sigma_2\rho$ where ρ is an injective mapping from the variables of C_2 to a set of fresh constants. If we can apply Backjump in NRCL then one of the three cases of the rule applies. We consider them separately. The first case cannot apply as we have assumed a clause different from $\bot$. The second case implies that we can backtrack to the very same level via SCL. For the third case, we note that in SCL there is no equivalent of the concept of blocking decisions or blocking clauses as they can only arise when a decision defines multiple ground literals. In SCL all literals defined by decisions in a conflict clause have different levels and thus we can always apply Backtrack from any grounding of the conflict clause if the third case of Backjump applies.

For the inductive step, consider a rule application $(\Gamma'_1; N; U; k'; (C_1; \sigma_1; \pi_1)) \Rightarrow_{\text{NRCL}} (\Gamma'_2; N; U; k'; (C_2; \sigma_2; \pi_2))$ in a conflict resolution. For any grounding $(\Gamma_2; N; U; k; C_2 \cdot \delta_2)$ of $(\Gamma'_2; N; U; k'; (C_2; \sigma_2; \pi_2))$ we build a grounding $(\Gamma_1; N; U; k;\ C_1 \cdot \delta_1)$ of $(\Gamma'_1; N; U; k'; (C_1; \sigma_1; \pi_1))$ from which we can still learn the clause C. In particular we will need to define Γ_1 and δ_1 in terms of $(\Gamma_2; N; U; k; C_2 \cdot \delta_2)$

Case Resolve: we consider the NRCL rule application

$$\begin{aligned}&(\Gamma'_1, (L_0; \sigma_0; \pi_0)^{C_0 \vee L_0}; N; U; k'; (C_1 \vee L_1; \sigma_1; \pi_1))\\ &\qquad \Rightarrow^{\text{Resolve}}_{\text{NRCL}} (\Gamma'_2, (L_0; \sigma_0; \pi_0)^{C_0 \vee L_0}; N; U; k'; (C_2; \sigma_2; \pi_2))\end{aligned}$$

by the conditions of Resolve in NRCL we have

1. there exists $\eta_0 = \text{mgu}(\text{comp}(L_0), L_1)$
2. there exists $\eta = \text{mgu}(\text{comp}(L_0)\sigma_0, L_1\sigma_1)$
3. $\eta_0\sigma_2 = \sigma_1\sigma_0\eta$
4. $C_2 = (C_1 \vee C_0)\eta_0$

and from the grounding we have $\delta_i = \sigma_i\delta'_i$ for some δ'_i, $i = 0, 1, 2$. It is clear that any grounding substitution δ_2 on the variables of $(C_1 \vee C_0)\eta_0$ can be induced by choosing opportune grounding substitutions δ_0 and δ_1. In particular, we can define δ_i for $i = 0, 1$ as the restriction of $\eta_0\delta_2\rho$ to the variables of $C_i \vee L_i$ where ρ is a grounding on $\text{var}(L_0\eta_0, L_1\eta_0) \setminus \text{var}(C_0\eta_0, C_1\eta_0)$. If there is a grounding ρ such that $L_0\eta_0\delta_2\rho$ is undefined in Γ_2 then we define Γ_1 as $\Gamma_2, L_0\eta_0\delta_2\rho^{C_0 \vee L_0 \cdot \eta_0\delta_2\rho}$. If such a substitution ρ does not exist then the literal $L_0\rho'^{C_0 \vee L_0 \cdot \rho'}$ is already defined in Γ_2 obtained from grounding the same literal $(L_0; \sigma_0; \pi_0)^{C_0 \vee L_0}$; we can then define Γ_1 equal to Γ_2 and resolve once more with the literal $L_0\rho'^{C_0 \vee L_0 \cdot \rho'}$.

Case Factorize: we consider the NRCL rule application

$$\begin{aligned}&(\Gamma'_1, (L_0; \sigma_0; \pi_0)^{C_0 \vee L_0}; N; U; k'; (C_1 \vee L_1 \vee L'_1; \sigma_1; \pi_1))\\ &\qquad \Rightarrow^{\text{Factorize}}_{\text{NRCL}} (\Gamma'_2, (L_0; \sigma_0; \pi_0)^{C_0 \vee L_0}; N; U; k'; (C_2 \vee L_2; \sigma_2; \pi_2))\end{aligned}$$

by the conditions on Factorize in NRCL we have

1. there exists $\eta_0 = \text{mgu}(L_1, L'_1)$
2. $C_2 = C_1\eta_0$ and $L_2 = L_1\eta_0$
3. there exists $\eta = \text{mgu}(\text{comp}(L_0)\sigma_0, L_1\sigma_1, L'_1\sigma_1)$
4. $\eta_0\sigma_2 = \sigma_1\eta$

and from the grounding we have $\delta_i = \sigma_i\delta'_i$. We can define $\delta_1 = \eta_0\delta_2$ which produces an acceptable grounding of $(C_1 \vee L_1 \vee L'_1; \sigma_1; \pi_1)$ as $\eta_0\delta_2 = \eta_2\sigma_2$ and so $\delta_1 = \sigma_1\delta'_1$ for $\delta'_1 = \eta\delta'_2$. We can, moreover, define $\Gamma_1 = \Gamma_2$ as in SCL Factorize is not restricted to the rightmost literal.

Case Skip: we consider the NRCL rule application

$$(\Gamma'_1, (L_0; \sigma_0; \pi_0)^{C_0 \vee L_0}; N; U; k'; (C_1; \sigma_1; \pi_1))$$
$$\Rightarrow^{\text{Skip}}_{\text{NRCL}} (\Gamma'_2; N; U; k'; (C_2 \vee L_2; \sigma_2; \pi_2)).$$

We can simply define $(\Gamma_1; N; U; k; C_1 \cdot \delta_1) = (\Gamma_2; N; U; k; C_2 \cdot \delta_2)$ where Factorize is independent from the trail in SCL and the induction step for Resolve can add any needed literal. □

Example 25 (SCL Trails may be Exponentially Longer). Consider an unsatisfiable clause set

$$N^n = \begin{cases} Q_n(x_1, \dots, x_n) \\ \neg Q_{i+1}(x_1, \dots, x_i, 0) \vee \neg Q_{i+1}(x_1, \dots, x_i, 1) \vee Q_i(x_1, \dots, x_i) & \text{if } 0 \le i < n \\ \neg Q_0 \end{cases}$$

NRCL can refute the clause set N^n at level $k = 0$ in linear time by building through rule Propagate the trail $\neg Q_0, Q_n(x_1, \dots, x_n), Q_{n-1}(x_1, \dots, x_{n-1}), \dots,$ $Q_1(x_1)$, where we do not show the clauses annotated to the literals, resulting in a conflict with $\neg Q_1(1) \vee \neg Q_1(0) \vee Q_0$. For SCL all ground instances of the $Q_i(x_1, \dots, x_i)$ have to be enumerated and there are 2^i many such ground instances for each Q_i.

Another relevant aspect is whether the SCL run constructed in Theorem 24 is (weakly-) regular. The example below shows it is not, in general, however at least for the example below, it is the case that SCL can actually learn a more general clause by a regular run.

Example 26 (Simulating SCL Runs are not (Weakly-) Regular). Consider the clauses

$$Q(x) \vee \neg P(x) \qquad P(x) \vee P(a) \vee P(b) \vee P(c) \qquad P(a) \vee \neg P(b)$$
$$P(b) \vee \neg P(c) \qquad P(c) \vee \neg P(a)$$

In NRCL we can decide the literal $\neg Q(x)$, propagate $\neg P(x)$ and result in a conflict with $P(x) \vee P(a) \vee P(b) \vee P(c)$. After factorization and resolution with $Q(x) \vee \neg P(x)$, NRCL learns the clause $Q(a) \vee Q(b) \vee Q(c)$. This clause cannot be learned with a (weakly-) regular run in SCL. After deciding any ground instance of $\neg Q(x)$, immediately all ground literals $\neg P(a)$, $\neg P(b)$, $\neg P(c)$ are propagated through the two literal clauses. Now the conflict does not only rely on the first two clauses but also involves the two literal clauses. After resolution and factoring steps, if SCL started with $\neg Q(a)$ it eventually learns the clause $Q(a)$ which makes the NRCL learned clause $Q(a) \vee Q(b) \vee Q(c)$ redundant.

6 Conclusion

The contributions of this paper are: (i) a sound, complete, SCL calculus for full first-order logic learning non-redundant clauses with respect to regular runs, (ii) weakly-regular runs do not exhaustively propagate unit clauses but still learn non-redundant clauses except for unit instances, (iii) the used notion of non-redundancy is NEXPTIME-complete for the BS fragment, (iv) SCL simulates resolution by non-regular runs, (v) SCL simulates NRCL by non-regular runs, (vi) exhaustive propagation is not always a good strategy for the BS fragment and beyond, and (vii) SCL is a decision procedure for the BS fragment.

The price for the simple SCL models is that trails can be exponentially longer compared to trails of calculi with more expressive model representation languages. For a BS clause set N the overall trail size is bound by mr^k where k is the maximal arity of a predicate in N, m the number of predicates, and r the number of constant symbols in N. Exploiting the actual recursive structure of N this bound can be further refined for a specific problem [10]. So in a simple preprocessing step, it can be checked whether an SCL trail potentially becomes "too large". Then a procedure can either start with a more expressive trail language [1,3,4,7,9,14,17] or dynamically decide to switch the trail model representation formalism. In practice, there are many interesting problems where the maximal predicate arity is not larger than three and there are not "too many" constants. Recall that all our examples inducing an exponentially growing trail or an exponentially growing proof length include the encoding of some type of binary counter.

Although SCL with a regular strategy and also all other calculi with exhaustive propagation cannot simulate resolution, the resolution calculus has also drawbacks. Worst case, the resolution calculus may generate more clauses, even in a terminating setting [7], than there are potential ground model assumptions as they are explored by SCL. Still one open question is whether the advantages of resolution and SCL can be combined: learning only non redundant clauses via partial model assumptions and being able to simulate non-redundant resolution inferences, in general. Such a result would unify both paradigms.

Acknowledgments. This work was funded by DFG grant 389792660 as part of TRR 248.

References

1. Alagi, G., Weidenbach, C.: NRCL - a model building approach to the Bernays-Schönfinkel fragment. In: Lutz, C., Ranise, S. (eds.) FroCoS 2015. LNCS (LNAI), vol. 9322, pp. 69–84. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-24246-0_5
2. Baumgartner, P.: Hyper tableau—the next generation. In: de Swart, H. (ed.) TABLEAUX 1998. LNCS (LNAI), vol. 1397, pp. 60–76. Springer, Heidelberg (1998). https://doi.org/10.1007/3-540-69778-0_14

3. Baumgartner, P., Fuchs, A., Tinelli, C.: Lemma learning in the model evolution calculus. In: Hermann, M., Voronkov, A. (eds.) LPAR 2006. LNCS (LNAI), vol. 4246, pp. 572–586. Springer, Heidelberg (2006). https://doi.org/10.1007/11916277_39
4. Bonacina, M.P., Plaisted, D.A.: Semantically-guided goal-sensitive reasoning: model representation. J. Autom. Reason. **56**(2), 113–141 (2016)
5. Bromberger, M., Sturm, T., Weidenbach, C.: Linear integer arithmetic revisited. In: Felty, A.P., Middeldorp, A. (eds.) CADE 2015. LNCS (LNAI), vol. 9195, pp. 623–637. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-21401-6_42
6. Fermüller, C.G., Pichler, R.: Model representation over finite and infinite signatures. J. Log. Comput. **17**(3), 453–477 (2007)
7. Hillenbrand, T., Weidenbach, C.: Superposition for bounded domains. In: Bonacina, M.P., Stickel, M.E. (eds.) Automated Reasoning and Mathematics. LNCS (LNAI), vol. 7788, pp. 68–100. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-36675-8_4
8. Kaufmann, B., Leone, N., Perri, S., Schaub, T.: Grounding and solving in answer set programming. AI Mag. **37**(3), 25–32 (2016)
9. 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
10. Korovin, K.: Non-cyclic sorts for first-order satisfiability. In: Fontaine, P., Ringeissen, C., Schmidt, R.A. (eds.) FroCoS 2013. LNCS (LNAI), vol. 8152, pp. 214–228. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-40885-4_15
11. Lewis, H.R.: Complexity results for classes of quantificational formulas. J. Comput. Syst. Sci. **21**(3), 317–353 (1980)
12. Nieuwenhuis, R., Oliveras, A., Tinelli, C.: Solving sat and sat modulo theories: from an abstract Davis-Putnam-Logemann-Loveland procedure to DPLL (T). J. ACM **53**, 937–977 (2006)
13. Navarro, J.A., Voronkov, A.: Proof systems for effectively propositional logic. In: Armando, A., Baumgartner, P., Dowek, G. (eds.) IJCAR 2008. LNCS (LNAI), vol. 5195, pp. 426–440. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-71070-7_36
14. Piskac, R., de Moura, L.M., Bjørner, N.: Deciding effectively propositional logic using DPLL and substitution sets. J. Autom. Reason. **44**(4), 401–424 (2010)
15. Plaisted, D.A.: Complete problems in the first-order predicate calculus. J. Comput. Syst. Sci. **29**, 8–35 (1984)
16. Sutcliffe, G.: The CADE ATP system competition - CASC. AI Mag. **37**(2), 99–101 (2016)
17. Teucke, A., Weidenbach, C.: First-order logic theorem proving and model building via approximation and instantiation. In: Lutz, C., Ranise, S. (eds.) FroCoS 2015. LNCS (LNAI), vol. 9322, pp. 85–100. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-24246-0_6