First-Order Automatic Literal Model Generation

Martin Bromberger, Florent Krasnopol, Sibylle Möhle, Christoph Weidenbach

First-Order Automatic Literal Model Generation

Martin Bromberger[1], Florent Krasnopol[1,2], Sibylle Möhle[1], and Christoph Weidenbach[1(✉)]

[1] Max Planck Institute for Informatics, Saarbrucken, Germany
{mbromber,fkrasnopol,smoehle,weidenbach}@mpi-inf.mpg.de
[2] École Normale Supérieure, Paris-Saclay, France

Abstract. Given a finite consistent set of ground literals, we present an algorithm that generates a complete first-order logic interpretation, i.e., an interpretation for all ground literals over the signature and not just those in the input set, that is also a model for the input set. The interpretation is represented by first-order linear literals. It can be effectively used to evaluate clauses. A particular application are SCL stuck states. The SCL (Simple Clause Learning) calculus always computes with respect to a finite number of ground literals. It then finds either a contradiction or a stuck state being a model with respect to the considered ground literals. Our algorithm builds a complete literal interpretation out of such a stuck state model that can then be used to evaluate the clause set. If all clauses are satisfied an overall model has been found. If it does not satisfy some clause, this information can be effectively explored to extend the scope of ground literals considered by SCL.

1 Introduction

Explicit and effective model representations as well as model building out of a set of first-order clauses have a long tradition [3,10,12,16,20,24,32,41,48,50,51]. In addition, they naturally arise out of decision procedures for decidable first-order clause set fragments [1–3,11,14,15,18,19,22,25–31,33,36,38,39,43,44,46,52–55]. The problem we are studying here is to the best of our knowledge new: given a finite set of consistent ground literals, find a finite representation of an overall, typically infinite Herbrand style interpretation, satisfying those ground literals. Of course, there are trivial solutions to this problem, e.g., by assigning any missing ground literal to true or false. Projecting the results of [23] to first-order logic results in such a trivial solution. However such a solution will not fit our motivating application: the family of SCL calculi [6,7,9,21,37] where we here concentrate on the case of first-order logic without equality. Similar to CDCL [40], SCL computes resolution inferences with respect to a partial ground model, i.e., a consistent sequence of first-order ground literals. The number of ground literals considered by SCL is finite at any point in time, thanks to an upper bound ground literal β with respect to a well-founded (quasi)-ordering. For the purpose of this paper we simply consider the number of symbols in a

C. Benzmüller et al. (Eds.): IJCAR 2024, LNAI 14739, pp. 133–153, 2024.
https://doi.org/10.1007/978-3-031-63498-7_9

literal with respect to $\leq$. In this context SCL either produces the empty clause with respect to β or a partial model satisfying all first-order clause instances smaller than β. In case of such a partial model we want to extend it to an overall interpretation for the clause set and then check whether this interpretation is a model for the first-order clause set considered, or, if not, find a suitable extension to β that then covers false clauses with respect to the generated interpretation. So all considered ground literals are instances of existing literals from some clause set. Therefore, we look for a solution that respects the term structure of the ground literals. Our approach starts with a universal relation and then refines it according to the term structure of the considered ground literals until it fits all ground literals,

For illustration, consider the following very simple example. For the three first-order clauses

$$\begin{array}{ll} \neg P(x) \vee P(g(g(x)) & P(a) \\ \neg P(g(g(g(g(a))))) & \end{array}$$

an SCL run with $\beta = P(g(g(g(a))))$, i.e., exclusively atoms smaller or equal $P(g(g(g(a))))$ are dealt with, where for the ordering we simply count symbols, a partial model generated by SCL could be

$$[P(a), P(g(g(a))), P(g(a))^1, P(g(g(g(a))))]$$

where the third literal is decided and the others are propagated from the above clauses. It is a model for all ground instances with literals smaller or equal $P(g(g(g(a))))$, hence, excluding $\neg P(g(g(g(g(a))))))$. Our model building calculus would start with state

$$(\{P(a), P(g(g(a))), P(g(a)), P(g(g(g(a))))\}; \emptyset; \{P(x)\}; \emptyset)$$

meaning that the initial model assumption for P is the universal relation, i.e., P holds on all ground terms. Processed ground literals are moved from the first to the second component of the state and final literal interpretation literals from the third to the fourth component by the algorithm. Thus finally, all processed ground literals are moved to the second component and the final literal model is contained in the fourth component while the other two are empty. One application of rule Solve, see page 7, immediately establishes the model $P(x)$, because it satisfies all ground literals. Of course, this interpretation does not satisfy the three clauses without the restriction to instances bounded by β. Still, we can use our interpretation to find the smallest clause instance falsified by it, in our example $\neg P(g(g(g(g(a)))))$, and use the maximal literal in that clause as our new $\beta = P(g(g(g(g(a)))))$. Running SCL with the new β will immediately yield the contradiction. Now consider a small modification of the three clauses where we replace the final unit clause by a disjunction.

$$\begin{array}{ll} \neg P(x) \vee P(g(g(x)) & P(a) \\ \neg P(g(g(g(g(a))))) \vee \neg P(g(g(g(a)))) & \end{array}$$

Running SCL on this clause set with $\beta = P(g(g(g(a))))$ may yield the same partial model as before and hence the same overall interpretation $P(x)$. Again

the final clause is falsified by the interpretation yielding a new minimal $\beta = P(g(g(g(g(a)))))$. Running SCL again with this β yields a final partial model

$$[P(a), P(g(g(a))), P(g(g(g(g(a))))), \neg P(g(g(g(a)))), \neg P(g(a))^1]$$

and now starting with this ground model

$$(\{P(a), P(g(g(a))), P(g(g(g(g(a))))), \neg P(g(g(g(a)))), \neg P(g(a))\}; \emptyset; \{P(x)\}; \emptyset)$$

the initial candidate interpretation $P(x)$ needs to be refined, because it has positive and negative instances among the set of ground literals. Refining means, we exhaustively instantiate $P(x)$ until no model candidate atom has both positive and negative instances by rule Refine, see page 6. This eventually yields

$$(\emptyset; \{P(a), P(g(g(a))), P(g(g(g(g(a))))), \neg P(g(g(g(a)))), \neg P(g(a))\};$$
$$\emptyset; \{P(a), \neg P(g(a)), P(g(g(a))), \neg P(g(g(g(a)))), P(g(g(g(g(x))))))\})$$

which in fact covers all ground literals and constitutes a model for the three clauses.

The paper is now organized as follows: after fixing some notions and notation in Sect. 2, and a short introduction to SCL, Sect. 3, our contributions are contained in Sect. 4. Important results are: (i) out of a set of ground literals we can generate in polynomial time an overall interpretation, Lemma 4, Lemma 8 and Lemma 5; (ii) our literal model representation satisfies the well-known requirements for explicit model representations [10], in particular supports effective clause evaluation, see page 13; (iii) the literal model representation can effectively be used to find a new minimal β in case it does not satisfy the clause set, Lemma 13. The paper ends with a discussion of the obtained results and further research directions, Sect. 5.

2 Preliminaries

We assume a first-order language without equality over a signature $\Sigma = (\Omega, \Pi)$ of operator symbols and predicates, respectively. All signature symbols come with a fixed arity. Terms, atoms, literals, clauses and clause sets are defined as usual, where in particular clauses are identified both as disjunctions and multisets of literals. Then 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. We write $f/1$ or $R/2$ for a function symbol of arity 1 or predicate symbol of arity 2, respectively. The complement of a literal is denoted by the function comp. The atom of a literal by the function atom, i.e., $\text{atom}(\neg A) = A$ and $\text{atom}(A) = A$. 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 $\text{dom}(\sigma) := \{x \mid x\sigma \neq x\}$ is finite and $\text{codom}(\sigma) := \{t \mid x\sigma = t, x \in \text{dom}(\sigma)\}$. Their application is extended to literals, clauses, and sets of such objects in the

usual way. A term, atom, clause, or a set of these objects is *ground* if it does not contain any variable. A substitution σ is *ground* if $\mathrm{codom}(\sigma)$ is ground. A substitution σ is *grounding* for a term t, literal L, clause C if $t\sigma$, $L\sigma$, $C\sigma$ is ground, respectively. A literal L is an *atom instance* of a literal K if $\mathrm{atom}(K)\sigma = \mathrm{atom}(L)$ for some σ. A term, literal is called *linear* if any variable occurs at most once in the term, literal. The function mgu denotes the *most general unifier* of two terms, atoms, literals. We assume that any mgu of two terms or literals does not introduce any fresh variables and is idempotent.

A *position* is a word over the naturals with empty word ϵ. The set of positions of a term, atom is inductively defined by: $\mathrm{pos}(x) = \{\epsilon\}$ if x is a variable, $\mathrm{pos}(f(t_1, \ldots, t_n)) = \{\epsilon\} \cup \bigcup_{i=1}^{n} \{ip \mid p \in pos(t_i)\}$ for terms, and $\mathrm{pos}(P(t_1, \ldots, t_n)) = \{\epsilon\} \cup \bigcup_{i=1}^{n} \{ip \mid p \in pos(t_i)\}$ for atoms. For a position $p \in \mathrm{pos}(t)$ we define $t|_p = t$ if $p = \epsilon$ and $f(t_1, \ldots, t_n)|_p = t_i|_{p'}$ if $p = ip'$. Moreover, we define by $t[s]_p$ the term, atom one receives by replacing the subterm $t|_p$ at position p of t with the term s. The size of a term t, atom A is defined by $\mathrm{size}(t) = |\mathrm{pos}(t)|$ or $\mathrm{size}(A) = |\mathrm{pos}(A)|$. The size of a substitution σ is defined by $\mathrm{size}(\sigma) = \sum_{x \in \mathrm{dom}(\sigma)} \mathrm{size}(x\sigma)$. The size of a set of terms, atoms, substitution is the sum of the size of its members. A position $p \in \mathrm{pos}(t)$ is *maximal* in t if for any other position $q \in \mathrm{pos}(t)$ we have $|q| \leq |p|$. The *depth* of a position p is 0 if $p = \epsilon$ and $|p|$ otherwise. The *depth* of a term t, atom A is the maximal depth of any position in t, A, i.e., $\mathrm{depth}(t) = \max\{|p| \mid p \in \mathrm{pos}(t)\}$ and $\mathrm{depth}(A) = \max\{|p| \mid p \in \mathrm{pos}(A)\}$, respectively. The depth of a term s in a term t is the depth of a maximal position p such that $t|_p = s$.

Two literals are *inconsistent* if they have different sign and their atoms are unifiable. A set of literals is *consistent* if it does not contain a pair of inconsistent literals. A *literal interpretation* M is a finite set of consistent literals. A literal interpretation $\mathcal{I}$ is *complete* with respect to a signature Σ if for any Σ ground atom A there is a literal $K \in \mathcal{I}$ such that $\mathrm{atom}(K)\sigma = A$ for some σ. A literal interpretation $\mathcal{I}$ satisfies a ground literal K, $\mathcal{I} \models K$ if there is an $L \in \mathcal{I}$ such that $L\sigma = K$ for some σ. It satisfies a non-ground literal K if it satisfies all groundings of K.

We overload notation for sets where "," is overloaded for disjoint union, and disjoint addition, e.g., "Γ_1, Γ_2" stands for $\Gamma_1 \cup \Gamma_2$ and Γ_1, L stands for the set $\Gamma_1 \cup \{L\}$.

3 SCL: Clause Learning from Simple Models

The family of SCL calculi (short "Simple Clause Learning") [6,9,21,37] lifts CDCL (Conflict-Driven Clause Learning) from propositional logic [34,42,49] to variants of first-order logic. The idea is to have superposition-style resolutions on non-ground, first-order clauses but instead of the usual static order that guides them, SCL uses as its guide ground partial models Γ, i.e., sequences of ground literals, also called *trails*. A trail for a clause set N is constructed/extended by guessing literals via so called Decisions and by propagating literals based on the current trail and the current clauses in N [9]. This construction continues until

we determine that Γ falsifies a ground instance $C\sigma$ of a clause $C \in N$. The conflict between Γ and C is then resolved by applying Resolution to C and the clauses used for propagation during the construction of Γ. At the end of these resolutions, SCL learns a new clause D and a prefix Γ' of Γ from which D can be propagated to start the construction of the next trail, which is guaranteed to never encounter the same conflict due to D. Furthermore D is not redundant, in particular, not subsumed by any clause.

The maximal length of the trail is always finitely bounded by all literals being smaller than a fixed ground literal β. In case all ground literals have been explored and not clause is falsified this constitutes a so called *stuck state*. In a stuck state the trail is model for all ground clause instantiations smaller β, but not in general.

In its first, original version [21], the focus of the SCL calculus is on deciding the Bernays-Schoenfinkel class without equality. Moreover, the original version is already a sound and refutationally complete semi-decision procedure for general first-order logic without equality that guarantees non-redundant clause learning. Subsequently, SCL has been extended to handle theories [6] and first-order logic with equality [37].

In the meantime, there exists a refined version [9] unifying and extending the previous versions [6,37] for first-order logic called SCL(FOL). In particular, this version introduces a refined Backtrack rule and a refined reasonable strategy criterion. In parallel we proved correctness and soundness of SCL(FOL) in Isabelle [5]. The Isabelle SCL(FOL) version relaxes some of the original requirements. SCL computations are performed with respect to a quasi-ordering $\preceq$ on ground atoms where the strict part is well-founded. We adopt this setting also in this paper by instantiating $\preceq$ with symbol counting and $\leq$. SCL(FOL) is only allowed to add literals L to the trail Γ with $\mathrm{atom}(L) \preceq \beta$ for some atom β. Note that the bounding atom β may grow, but only if we reach a stuck state, where $\Gamma \models \mathrm{gnd}^{\preceq\beta}(N)$ and where the function $\mathrm{gnd}^{\preceq\beta}$ computes the set of all ground instances of a clause set where the grounding is restricted to produce literals L with $\mathrm{atom}(L) \preceq \beta$. This guarantees that SCL(FOL) (with a reasonable strategy) will always find a refutation if the input clause set is unsatisfiable. Moreover, for a fixed β, SCL(FOL) turns into a decision procedure for $\mathrm{gnd}^{\preceq\beta}(N)$. And even if we allow β to grow, SCL(FOL) regularly visits partial models Γ that at least satisfy $\mathrm{gnd}^{\preceq\beta}(N)$, that may even be extendable to full models for N, or at least guide the selection for our next bounding literal β' [8].

4 Generating Models

A motivation for our model generating algorithm is the extension of SCL ground trail Γ out of a stuck state to a complete literal interpretations. Such an interpretation either satisfies the considered clause set, or it falsifies some clause. The latter information can then be used to extend the SCL search for a model or a contradiction. Our extension from Γ is not trivial, e.g., by assigning all atoms beyond Γ to true. Instead, it respects the literal structure in Γ and naturally extends it to a complete literal interpretation.

The starting point is simply a set of ground literals and the finite signature used to build the set. The algorithm is presented by three abstract rewrite rules operating on a state in a non-deterministic way. The state is a tuple $(\Gamma; \Delta; \mathcal{I}; M)$ where Γ, Δ are consistent sets of ground literals, M is a set of linear literals that defines a partial interpretation such that $M \models L$ for each $L \in \Delta$ and M does not have any conflict with Γ; $\mathcal{I}$ is a set of linear atoms such that $\mathcal{I} \cup M$ represents a complete literal interpretation; initially M is empty and $\mathcal{I}$ the set $\{P(x_1, \ldots, x_n)\}$ for some predicate P and linear atom $P(x_1, \ldots, x_n)$, denoting the universal relation for P. Processed literals/atoms are moved by the rewrite rules from Γ to Δ and $\mathcal{I}$ to M, respectively. The rewrite calculus then builds an overall interpretation of P according to Γ where we assume that Γ only contains P literals. So given a set of ground literals for each occurring predicate a separate run starting with the respective literals is needed.

The start state is $(\Gamma; \emptyset; \{P(x_1, \ldots, x_n)\}; \emptyset)$ for a finite consistent set of ground literals Γ over P and linear atom $P(x_1, \ldots, x_n)$ and a final state is $(\emptyset; \Delta; \emptyset; M)$ where we will show $M \models \Gamma$. We assume a finite signature Σ.

The first rule Refine covers the situation where some atom A in $\mathcal{I}$ has both positive and negative instances in Γ. Since Γ is consistent, the atom A can be split into instances A_i of itself and each of the resulting instances is guaranteed to eventually have only positive or negative instances in Γ. Note that this may require repeated applications of the rule Refine.

Refine $(\Gamma; \Delta; \mathcal{I}, P(t_1, \ldots, t_n); M)$
$\Rightarrow_{\text{mod}}$ $(\Gamma; \Delta; \mathcal{I}, \cup_{f_i/k \in \Omega}\{P(t_1, \ldots, t_n)\{x \mapsto f_i(y_{i_1}, \ldots, y_{k_i})\}\}; M)$
provided $P(t_1, \ldots, t_n)|_p = x$, f_i are all function symbols (including constants) from Ω and k_i denotes their respective arity, all variables in $f_i(y_{i_1}, \ldots, y_{k_i})$ are fresh, different variables, and p is a minimal variable position in $P(t_1, \ldots, t_n)$ for which there exist literals $P(l_1, \ldots, l_n)$ and $\neg P(r_1, \ldots, r_n)$ in Γ such that both are atom instances of $P(t_1, \ldots, t_n)$, and the atoms $P(l_1, \ldots, l_n)|_p$ and $P(r_1, \ldots, r_n)|_p$ are not unifiable

Due to refinement and the construction of the final complete literal interpretation it may happen that certain atoms in $\mathcal{I}$ do not have any instances in Γ. They are then moved to the final representation of the interpretation by rule Clean.

Clean $(\Gamma; \Delta; \mathcal{I}, P(t_1, \ldots, t_n); M)$
$\Rightarrow_{\text{mod}}$ $(\Gamma; \Delta; \mathcal{I}; M, P(t_1, \ldots, t_n))$
provided $P(t_1, \ldots, t_n)$ has no atom instance in Γ

Actually, the atom $P(t_1, \ldots, t_n)$ could be added positively or negatively to the final literal interpretation. In favor of Theorem 12 we stick here to adding all literals without instances positively. If all instances of some atom in $\mathcal{I}$ from Γ have an identical sign, they are solved and both the atom and the instances can be removed from $\mathcal{I}$ and Γ by rule Solve, respectively.

Solve $(\Gamma; \Gamma'; \Delta; \mathcal{I}, P(t_1, \ldots, t_n); M)$
$\Rightarrow_{\text{mod}}$ $(\Gamma; \Delta, \Gamma'; \mathcal{I}; M, \#P(t_1, \ldots, t_n))$

provided Γ', $\Gamma' \neq \emptyset$, consists only of positive literal instances of $P(t_1, \ldots, t_n)$ or only of negative literal instances; Γ contains no literal instances of $P(t_1, \ldots, t_n)$; $\# = \neg$ if the literals in Γ' are negative and $\#$ is empty otherwise

Example 1. Let $\Sigma = (\{a/0, f/1, g/1\}, \{P/3\})$ be a signature. Now consider $\Gamma = \{K, L\}$ where $K = P(a, f(a), a)$, $L = \neg P(a, g(a), a)$ over the signature Σ. An execution trace of $\Rightarrow_{\text{mod}}$ is as follows:

$$
\begin{array}{ll}
1: & (\{K, L\}; \emptyset; \{P(x_1, x_2, x_3)\}; \emptyset) \\
2: \Rightarrow_{\text{mod}}^{\text{Refine}} & (\{K, L\}; \emptyset; \{P(x_1, a, x_3), P(x_1, f(y_1), x_3), P(x_1, g(y_2), x_3)\}; \emptyset) \\
3: \Rightarrow_{\text{mod}}^{\text{Clean}} & (\{K, L\}; \emptyset; \{P(x_1, f(y_1), x_3), P(x_1, g(y_2), x_3)\}; \{P(x_1, a, x_3)\}) \\
4: \Rightarrow_{\text{mod}}^{\text{Solve}} & (\{L\}; \{K\}; \{P(x_1, g(y_2), x_3)\}; \{P(x_1, a, x_3), P(x_1, f(y_1), x_3)\}) \\
5: \Rightarrow_{\text{mod}}^{\text{Solve}} & (\emptyset; \{K, L\}; \emptyset; \{P(x_1, a, x_3), P(x_1, f(y_1), x_3), \neg P(x_1, g(y_2), x_3)\})
\end{array}
$$

Step 1: The initial state $(\Gamma; \Delta; \mathcal{I}; M)$ consists of the set $\Gamma = \{K, L\}$, the empty set Δ, the set $\mathcal{I}$ containing only $P(x_1, x_2, x_3)$ which generalizes all literals over Σ with predicate P, and the empty set of literals M.

Step 2: Both K and L are atom instances of $P(x_1, x_2, x_3)$, but with opposite signs. Moreover, the terms $K|_2 = f(a)$ and $L|_2 = g(a)$ are not unifiable, and the position $p = 2$ is the minimal variable position in $P(x_1, x_2, x_3)$ for which this is the case, and the preconditions of rule Refine are met. A refinement of $P(x_1, x_2, x_3)$ in position 2 takes place and $P(x_1, x_2, x_3)$ is replaced by literals differing from it only in position 2 by replacing x_2 by every constant and function symbol occurring in Σ. The resulting atoms are $P(x_1, a, x_3)$, $P(x_1, f(y_1), x_3)$, and $P(x_1, g(y_2), x_3)$ and they cover again all P ground instances.

Step 3: The literal $P(x_1, a, x_3) \in \mathcal{I}$ has no atom instance on Γ and is moved to M by means of rule Clean.

Step 4: The positive literal K is the only instance of $P(x_1, f(y_1), x_3)$ on Γ, and the preconditions of rule Solve are met. The literal K is moved from Γ to Δ, and $P(x_1, f(y_1), x_3)$ is moved from $\mathcal{I}$ to M.

Step 5: The negative literal L is the only atom instance of $P(x_1, g(y_2), x_3)$ with negative sign, and the preconditions of rule Solve are met. The literal L is moved from Γ to Δ, whereas $P(x_1, g(y_2), x_3)$ is removed from $\mathcal{I}$ and added to M with negative sign. Now both Γ and $\mathcal{I}$ are empty, and the execution stops with the linear complete literal interpretation $M = \{P(x_1, a, x_3), P(x_1, f(y_1), x_3), \neg P(x_1, g(y_2), x_3)\}$.

Next we prove that $\Rightarrow_{\text{mod}}$ always computes an overall interpretation and model of the initial Γ. The basis for these results is the notion of a sound state below. We then show by induction on the length of a $\Rightarrow_{\text{mod}}$ derivation that the initial state is sound and any follower state is sound assuming its start state is sound.

Definition 2 (Sound State). *A state* $(\Gamma; \Delta; \mathcal{I}; M)$ *is* sound*, if the following invariants hold:*

1. *All literals in* $\mathcal{I} \cup M$ *are linear.*
2. *The atoms of any two different literals from* $\mathcal{I} \cup M$ *are not unifiable.*
3. *For any ground atom* A *over* P *there is an atom* B *in* $\mathcal{I} \cup M$ *such that* $A = B\sigma$ *for some* σ*.*
4. *Any literal* $L \in \Delta$ *is an instance of a literal* $K \in M$*.*
5. *Any literal* $L \in \Gamma$ *is an atom instance of a literal* $K \in \mathcal{I}$*.*
6. *The maximal depth of an atom in* $\mathcal{I} \cup M$ *is at most one larger than the maximal depth of a ground atom in* $\Gamma \cup \Delta$*.*

Lemma 3 (Soundness of Initial State). *The initial state* $(\Gamma; \emptyset; \{P(x_1, \ldots, x_n)\}; \emptyset)$ *is sound.*

Proof. Invariants 1 to 6 given in Definition 2 hold in the initial state:
1. Only $P(x_1, \ldots, x_n) \in \mathcal{I} \cup M$ which is linear by definition.
2. Holds trivially, because $\mathcal{I} \cup M$ contains only $P(x_1, \ldots, x_n)$.
3. The atom $P(x_1, \ldots, x_n) \in \mathcal{I} \cup M$ generalizes all ground atoms over Σ with predicate P.
4. Holds trivially, because $\Delta = \emptyset$.
5. The atom B of $P(x_1, \ldots, x_n) \in \mathcal{I}$ generalizes all ground atoms over Σ with predicate P, and any atom$(L) \in \Gamma$ is one of those.
6. Holds trivially, because the maximal depth of $P(x_1, \ldots, x_n) \in \mathcal{I}$ is equal to one. □

Lemma 4. (Soundness of $\Rightarrow_{\mathbf{mod}}$ Rules). *The rules of* $\Rightarrow_{mod}$ *preserve state soundness.*

Proof. The proof is carried out by induction over the number of rule applications. By the induction hypothesis, we assume Invariants 1 to 6 given in Definition 2 hold in a state $(\Gamma'; \Delta'; \mathcal{I}'; M')$ and show that after the application of any rule they are still met in the resulting state $(\Gamma; \Delta; \mathcal{I}; M)$.

Rule Refine. The literal $L = P(t_1, \ldots, t_n) \in \mathcal{I}'$ is replaced by literals L_i differing from L solely in the position p, which now contains either a constant symbol or a function symbol whose arguments are fresh different variables.
1. The literal L is linear by the induction hypothesis. The variables introduced in the literals L_i are fresh and different, hence all literals L_i are linear, too. The literals in M' are linear by the induction hypothesis and M' remains unaffected. Therefore, all literals in $\mathcal{I} \cup M$ are linear as well.
2. By the induction hypothesis, no two atoms in $\mathcal{I}' \cup M'$ are unifiable. This holds in particular for the atom of L and any other atom in $\mathcal{I}' \cup M'$. The terms $L_i|_p$ are not unifiable by the definition of rule Refine, hence their atoms are not unifiable. Since the literals L_i are instances of L, their atoms are not unifiable with any atom in $(\mathcal{I}' \cup M') \setminus \{L\}$ and thus of $\mathcal{I} \cup M$. Moreover, the atoms $\mathcal{I}' \cup M' \setminus \{L\}$ are by induction hypothesis not unifiable with each other.
3. Let A be any ground atom. By the induction hypothesis, there exists a literal K in $\mathcal{I}' \cup M'$ such that A is an instance of atom(K), i.e., there exists a substitution σ such that $A =$ atom$(K)\sigma$. If K is not the literal L that is refined, then K is

still in $\mathcal{I} \cup M$ and so the property holds for atom A. If K is the literal L that is refined on position p with $x = L|_p$, then we know that atom A has a term $A|_p = f_i(s_1, \ldots, s_{k_i})$ at position p. This means A is also an instance of the literal $L_i = P(t_1, \ldots, t_n)\{x \mapsto f_i(y_{i_1}, \ldots, y_{k_i})\}$ that was newly added to $\mathcal{I}$ such that $A = \text{atom}(L_i)\sigma_i$, where $\sigma_i = \sigma \cup \{y_{i_1} \mapsto s_1, \ldots, y_{k_i} \mapsto s_{k_i}\}$. Hence the property holds for all ground atoms.
4. Both Δ' and M' remain unaffected, and Invariant 4 still holds.
5. The set Γ' remains unaffected by rule Refine, and its atoms are ground atoms over P. By the induction hypothesis, for any atom A in Γ' there exist a literal K in $\mathcal{I}'$ with $\text{atom}(K) = B'$ and a σ such that $B'\sigma = A$. If K is not the literal L that is refined, then K is still in $\mathcal{I}$ and so the property holds for atom A. If K is the literal L that is refined at position p with $x = L|_p$, then we know that atom A has a term $A|_p = f_i(s_1, \ldots, s_{k_i})$ at position p. This means A is also an instance of the literal $L_i = P(t_1, \ldots, t_n)\{x \mapsto f_i(y_{i_1}, \ldots, y_{k_i})\}$ that was newly added to $\mathcal{I}$ such that $A = \text{atom}(L_i)\sigma_i$ where $\sigma_i = \sigma \cup \{y_{i_1} \mapsto s_1, \ldots, y_{k_i} \mapsto s_{k_i}\}$. Hence the property holds for all atoms A in Γ.
6. By the definition of rule Refine, the depth of any literal L_i added to $\mathcal{I}'$ can be at most the depth of $P(t_1, \ldots, t_n)$ plus one, due to the introduction of a function symbol at position p. Therefore, the maximal depth of the literals in $\mathcal{I}'$ may increase at most by one. However, the depth of any atom in Γ' which is an instance of $P(t_1, \ldots, t_n)$ is at least equal to the one of $P(t_1, \ldots, t_n)$. Furthermore, Γ', Δ', and M' remain unaltered, and therefore Invariant 6 still holds after the application of rule Refine.

Rule Clean.
1–3,5,6. The sets Γ' and Δ' remain unaltered. The literal $P(t_1, \ldots, t_n)$ in $\mathcal{I}'$ is moved from $\mathcal{I}'$ to $\mathcal{I}$. It remains unaffected, just as any other literal in $\mathcal{I}' \cup M'$, and by the induction hypothesis, Invariants 1–3, 5, and 6 hold.
4. The ground set Δ' remains unchanged, and no literal is removed from M', therefore Invariant 4 still holds after executing rule Clean.

Rule Solve.
1–3,6. The literal $P(t_1, \ldots, t_n)$ is moved from $\mathcal{I}'$ to M, either with positive or negative sign. Its atom remains unaffected, just as any other literal in $\mathcal{I}' \cup M'$, and by the induction hypothesis, Invariants 1–3, and 6 hold.
4. The literals added to Δ are ground instances of $\#P(t_1, \ldots, t_n)$ added to M, and Invariant 4 is met after the application of rule Solve.
5. The literals removed from Γ' are ground instances of $\#P(t_1, \ldots, t_n)$ and $P(t_1, \ldots, t_n)$ is removed from $\mathcal{I}'$. Therefore, Invariant 5 still holds after applying rule Solve.
6. The removal of $P(t_1, \ldots, t_n)$ from $\mathcal{I}'$ and the addition of $\#P(t_1, \ldots, t_n)$ to M do not affect the maximal depth of the atoms in $\mathcal{I}' \cup M'$, and $\Gamma' \cup \Delta'$ remains unaffected. Therefore, Invariant 6 still holds after applying rule Solve. □

Next we show termination and that $\Rightarrow_{\text{mod}}$ does not get stuck and always ends in a final state. From now on we only consider sound states.

Lemma 5 (Termination and Runtime). *$\Rightarrow_{mod}$ terminates in polynomial time $O(\text{size}(\Gamma)^2)$ with respect to the size of Γ.*

Proof. For a state $(\{L_1, \ldots, L_n\}; \Delta; \mathcal{I}; M)$ let $\mathcal{I}'$ be a multiset of literals $\{K_1, \ldots, K_n\}$ out of literals from $\mathcal{I}$ such that $\text{atom}(L_i) = \text{atom}(K_i)\sigma_i$ for some σ_i. Note that for a given $\{L_1, \ldots, L_n\}$ and $\mathcal{I}$ the multiset $\mathcal{I}'$ is unique. This is a result of a sound state and Invariant 2.2. Let

$$\delta(\{L_1, \ldots, L_n\}, \{K_1, \ldots, K_n\}) = \sum_{1 \leq i \leq n} \text{size}(\sigma_i).$$

Now the measure $(\delta(\{L_1, \ldots, L_n\}, \{K_1, \ldots, K_n\}), |\mathcal{I}|)$ with $>_{\text{lex}}$ strictly decreases with each rule application: The rules Clean and Solve strictly decrease the number of L_i and/or $|\mathcal{I}|$. For the rule Refine the atom $P(t_1, \ldots, t_n)$ has at least two different instances among the L_i and after application of the rule the respective σ_i for all those instances decrease in size by one.

There are at most $\text{size}(\Gamma)$ many applications of Refine possible and for each of these applications at most $\text{size}(\Gamma)$ many applications of Clean or Solve are possible, resulting in the above upper bound. Please recall that the number of symbols in Ω is also bound by $\text{size}(\Gamma)$. □

Lemma 6 (No Stuck States). *If for a state $(\Gamma; \Delta; \mathcal{I}; M)$ we have $\Gamma \neq \emptyset$ or $\mathcal{I} \neq \emptyset$ then at least one $\Rightarrow_{mod}$ rule is applicable.*

Proof. Suppose $\Gamma \neq \emptyset$ and $L \in \Gamma$. By soundness, Definition 2.5, there exists a literal $K \in \mathcal{I}$ such that $\text{atom}(L)$ is an instance of K. If in addition $\mathcal{I}$ contains a literal H of sign opposite to the one of L where $\text{atom}(H)$ is an instance of K and a minimal variable position p in K such that $\text{atom}(L)|_p$ and $\text{atom}(H)|_p$ are not unifiable, the preconditions of rule Refine are met. If instead all literals $H \in \Gamma$, whose atoms are an instance of K, have the same sign as L, rule Solve can be applied. By Definition 2.5, it can not happen that $\Gamma \neq \emptyset$ and $\mathcal{I} = \emptyset$. Now assume $\Gamma = \emptyset$ and $\mathcal{I} \neq \emptyset$ and let L be a literal in $\mathcal{I}$. No atom instance of L is contained in Γ, and the preconditions of rule Clean are met. □

A consequence of Lemma 6 is that $\Rightarrow_{\text{mod}}$ always makes progress, i.e., in any non-terminal state, a rule is applicable. Finally, we prove that $\Rightarrow_{\text{mod}}$ in fact produces an overall interpretation satisfying the literals from the initial state.

Lemma 7 (All Literals are Considered). *Let $(\Gamma_0; \emptyset; \{P(x_1, \ldots, x_n)\}; \emptyset)$ be an initial state. Then for any (possibly non-final) state $(\Gamma; \Delta; \mathcal{I}; M)$ obtained during the execution of $\Rightarrow_{mod}$ on the initial state, it holds that $\Gamma \cup \Delta = \Gamma_0$.*

Proof. In the initial state $(\Gamma_0; \Delta_0; \{P(x_1, \ldots, x_n)\}; \emptyset)$, this is obviously the case since $\Delta_0 = \emptyset$. For proving that this property holds throughout the execution of $\Rightarrow_{\text{mod}}$, we assume that it holds in a state $(\Gamma'; \Delta'; \mathcal{I}'; M')$ and show that after applying one rule, it is still met in the resulting state $(\Gamma; \Delta; \mathcal{I}; M)$.

Refine, Clean. Both Γ' and Δ' remain unaltered, hence $\Gamma \cup \Delta = \Gamma_0$.

Solve. Literals are moved from Γ_0 to Δ, hence $\Gamma \cup \Delta = \Gamma_0 \cup \Delta_0 = \Gamma_0$. □

Lemma 8 (Complete Linear Literal Model). *Let* $(\Gamma_0; \emptyset; \{P(x_1, \ldots, x_n)\}; \emptyset)$ *be an initial state and* $(\emptyset; \Delta; \emptyset; M)$ *a final state generated by executing* $\Rightarrow_{mod}$ *on it. Then* M *is a complete linear literal model of* Γ_0.

Proof. M is a complete linear literal interpretation by Definition 2.1-3, Lemma 4. By Lemma 7, we have $\Delta = \Gamma_0$. By Definition 2.4, the literals in M generalize all literals in Δ and hence in Γ_0. This proves that M is a model of Γ_0. □

Our rules are not deterministic, and several factors affect the model obtained by running $\Rightarrow_{\text{mod}}$ with the same initial state $(\Gamma_0; \emptyset; P(x_1, \ldots, x_n); \emptyset)$. If the preconditions of multiple rules are met in a non-final state $(\Gamma; \Delta; \mathcal{I}; M)$, we are free to choose the order in which we execute them. If there are literals $L, K \in \mathcal{I}$ meeting the preconditions of Refine with respect to the same minimal variable position p, either may be chosen. Thus applying $\Rightarrow_{\text{mod}}$ to the same trail twice might give us two literal interpretations of different size as shown by an example.

Example 9 (Model Size). Consider the signature $\Sigma = (\{a/0, g/1\}, \{R/2\})$ and $\Gamma_0 = \{L_1, L_2, L_3, L_4, L_5, L_6\}$ where $L_1 = \neg R(a, a)$, $L_2 = R(a, g(a))$, $L_3 = R(g(a), g(g(a)))$, $L_4 = R(a, g(g(a)))$, $L_5 = \neg R(g(a), g(a))$, and $L_6 = \neg R(g(a), a)\}$. A possible run is shown below. The variables or literals we refine in the next step or apply Solve or Clean to, respectively, are underlined.

$$
\begin{array}{ll}
0: & (\Gamma_0; \emptyset; \{R(\underline{x}, y)\}; \emptyset) \\
1: \Rightarrow_{\text{mod}}^{\text{Refine}} & (\Gamma_0; \emptyset; \{R(a, \underline{y}), R(g(z), y)\}; \emptyset) \\
2: \Rightarrow_{\text{mod}}^{\text{Refine}} & (\Gamma_0; \emptyset; \{\underline{R(a, a)}, \underline{R(a, g(u))}, R(g(z), y)\}; \emptyset) \\
3: \Rightarrow_{\text{mod}}^{\text{Solve*}} & (\Gamma_1; \Delta_1; \{R(g(z), \underline{y})\}; M_1) \\
4: \Rightarrow_{\text{mod}}^{\text{Refine}} & (\Gamma_1; \Delta_1; \{\underline{R(g(z), a)}, R(g(z), g(v))\}; M_1) \\
5: \Rightarrow_{\text{mod}}^{\text{Solve}} & (\Gamma_2; \Delta_2; \{R(g(z), g(\underline{v}))\}; M_2) \\
6: \Rightarrow_{\text{mod}}^{\text{Refine}} & (\Gamma_2; \Delta_2; \{\underline{R(g(z), g(a))}, \underline{R(g(z), g(g(w)))}\}; M_2) \\
7: \Rightarrow_{\text{mod}}^{\text{Solve*}} & (\Gamma_3; \Delta_3; \emptyset; M_3)
\end{array}
$$

The initial state is given by $(\Gamma_0; \emptyset; \{R(x, y)\}; \emptyset)$ (step 0). We choose to refine x at position $p = 1$ in $R(x, y)$, since L_1 and L_3 are instances of its atom with differing signs and $L_1|_1$ and $L_3|_1$ are not unifiable. Rule Refine replaces $R(x, y)$ by $R(a, y)$ and $R(g(z), y)$ (step 1). Similarly, we refine the variable y at position $p = 2$ in $R(a, y)$, since L_1 and L_2 are instances of it having different sign and $L_1|_2$ and $L_2|_2$ are not unifiable (step 2). Then rule Solve can be applied twice, namely to $R(a, a)$ and its negative instance $L_1 \in \Gamma_0$, and to $R(a, g(u))$ and its positive instances $L_2, L_4 \in \Gamma_0$. We obtain $\Gamma_1 = \Gamma_0 \setminus \{L_1, L_2, L_4\}$, $\Delta_1 = \{L_1, L_2, L_4\}$, and $M_1 = \{\neg R(a, a), R(a, g(u))\}$ (step 3). Next, the variable y at position $p = 2$ in $R(g(z), y)$ is refined, since L_3 and L_5 are instances of it having opposite sign and their subterms at position 2 are not unifiable. The literal $R(g(z), y)$ is replaced by $R(g(z), a)$ and $R(g(z), g(v))$ (step 4). Since Γ_1 contains only a positive instance of $R(g(z), a)$, namely L_6, rule Solve is applied resulting in $\Gamma_2 = \Gamma_1 \setminus \{L_6\}$, $\Delta_2 = \Delta_1 \cup \{L_6\}$, and $M_2 = M_1 \cup \{\neg R(g(z), a)\}$ (step 5). The trail Γ_2 contains instances L_3 and L_5 of $R(g(z), g(v))$ with different sign. Variable v at position 2 1 is chosen for refinement, since $L_3|_{2\,1}$ and $L_5|_{2\,1}$ are not unifiable,

and $R(g(z), g(v))$ is replaced by $R(g(z), g(a))$ and $R(g(z), g(g(w)))$ (step 6). Now Γ_2 contains only a positive instance of $R(g(z), g(a))$ and a negative one of $R(g(z), g(g(w)))$, and rule Solve is applicable. This gives us $\Gamma_3 = \Gamma_2 \setminus \{L_3, L_5\} = \emptyset$, $\Delta_3 = \Delta_2 \cup \{L_3, L_5\} = \Gamma_0$, and $M_3 = M_2 \cup \{\neg R(g(z), g(a)), R(g(z), g(g(w)))\}$ with 5 literals (step 7).

The choice of the variable to be refined in step 1 is not deterministic, and the following steps might lead to a different model. A different run for $\Gamma'_0 = \Gamma_0$ could be as follows:

$$
\begin{aligned}
0: &\quad (\Gamma'_0; \emptyset; \{R(x, \underline{y})\}; \emptyset) \\
1: &\Rightarrow^{\text{Refine}}_{\text{mod}} (\Gamma'_0; \emptyset; \{\underline{R(x, a)}, R(x, g(z))\}; \emptyset) \\
2: &\Rightarrow^{\text{Solve}}_{\text{mod}} (\Gamma'_1; \Delta'_1; \{\overline{R(\underline{x}, g(z))}\}; M'_1) \\
3: &\Rightarrow^{\text{Refine}}_{\text{mod}} (\Gamma'_1; \Delta'_1; \{\underline{R(a, g(z))}, R(g(u), g(z))\}; M'_1) \\
4: &\Rightarrow^{\text{Solve}}_{\text{mod}} (\Gamma'_2; \Delta'_2; \{\overline{R(g(u), g(\underline{z}))}\}; M'_2) \\
5: &\Rightarrow^{\text{Refine}}_{\text{mod}} (\Gamma'_2; \Delta'_2; \{\underline{R(g(u), g(a))}, \underline{R(g(u), g(g(v)))}\}; M'_2) \\
6: &\Rightarrow^{\text{Solve}^*}_{\text{mod}} (\Gamma'_3; \Delta'_3; \emptyset; M'_3)
\end{aligned}
$$

The first refinement step involves $p = 2$, $y = R(x, y)|_2$ motivated by L_1 and L_2 (step 1). Now we can execute Solve on $R(x, y)$ since it has only negative instances on Γ'_0, which are L_1 and L_6 obtaining $\Gamma'_1 = \Gamma_0 \setminus \{L_1, L_6\}$, $\Delta'_1 = \Delta_0 \cup \{L_1, L_6\}$, and $M'_1 = \{\neg R(x, a)\}$ (step 2). The variable x at position 1 of $R(x, g(z))$ is refined, since L_2 and L_5 are a positive and negative instance, respectively, of $R(g(u), g(z))$ and their subterms at position 1 are not unifiable, by replacing $R(x, g(z))$ by $R(a, g(z))$ and $R(g(u), g(z))$ (step 3). Now rule Solve can be executed on $R(a, g(z))$, which generalizes L_2 and L_4, which are both positive. This results in $\Gamma'_2 = \Gamma'_1 \setminus \{L_2, L_4\}$, $\Delta'_2 = \Delta'_1 \cup \{L_2, L_4\}$, and $M'_2 = M'_1 \cup \{R(a, g(z))\}$ (step 4). Next, a Refine step on z at position 2 1 in $R(g(u), g(z))$ is executed due to L_3 and L_5, which are instances of $R(g(u), g(z))$ of opposite sign and whose subterms at position 2 1 are not unifiable. The literal $R(g(u), g(z))$ is replaced by $R(g(u), g(a))$ and $R(g(u), g(g(v)))$ (step 5), which have one instance each in Γ'_2. Rule Solve is applied twice, resulting in $\Gamma'_3 = \Gamma'_2 \setminus \{L_3, L_5\} = \emptyset$, $\Delta'_3 = \Delta'_2 \cup \{L_3, L_5\} = \Gamma'_0$, and $M'_3 = M'_2 \cup \{\neg R(g(u), g(a)), R(g(u), g(g(v)))\}$ with 4 literals.

So M_3 and M'_3 not only differ syntactically, but also contain a different number of literals. Refining x before y led to $\neg R(a, a), \neg R(g(z), a) \in M_3$, whereas refining y before x resulted in $\neg R(x, a) \in M'_3$, which generalizes both $\neg R(a, a)$ and $\neg R(g(z), a)$.

In summary, $\Rightarrow_{\text{mod}}$ computes an overall interpretation out of the initial finite set of consistent ground literals in polynomial time. We shortly compare our model representation formalism with the long standing literature, in particular [10,17]. They suggested four postulates which should ideally be met by any model representation formalism:

- **Uniqueness.** Each model representation M specifies a single interpretation over Σ.

- **Atom Test.** There exists a fast procedure to evaluate arbitrary ground atoms over the signature Σ in M.
- **Formula Evaluation.** There exists an algorithm deciding the truth value of an arbitrary formula over Σ in M.
- **Equivalence Test.** There exists an algorithm deciding whether two representations M and M' over Σ describe the same interpretation.

The model M obtained by $\Rightarrow_{\text{mod}}$ is a complete linear literal interpretation. Our representation formalism is therefore a special case of an atomic representation (ARM) [10] if we leave out negative literals which are implicit for ARMs. The validity of the four model building postulates has been shown for ARMs [10]. So the models computed by $\Rightarrow_{\text{mod}}$ satisfy the four model building postulates. Clause evaluation for our linear literal models M is straightforward: a clause C is valid iff there is no substitution σ such that for each $L \in C$ there is a literal $K \in M$ such that $L\sigma$ and $K\sigma$ are complementary. Recall that this is a consequence of the fact that our literal interpretations are explicit and complete: for any ground atom A over Σ there is a literal K in M such that A is a literal instance of K. The respective procedure for ARMs is more involved [45], whereas in our case established techniques for hyper-resolution apply [35,47,56].

Finally, we show consequences out of our model building procedure for non-ground literals and the SCL calculus: if the computed interpretation does not satisfy all clauses, then it can be used to effectively compute a minimal extension to the ground literal restriction of the SCL calculus.

Theorem 10 (Non-ground Guarantees). *Let Γ be a set of consistent ground literals. Let M be a model generated by $\Rightarrow_{mod}$ from Γ. Let L be a (potentially non-ground,) linear literal with* $\text{depth}(\text{atom}(L)) = d$. *Let $\epsilon = 1$ if L has a position p of depth d (i.e., $|p| = d$) such that $L|_p$ is a constant. Otherwise, $\epsilon = 0$. Let Γ contain all ground instances $L\sigma$ of L (i.e., $L\sigma \in \Gamma$) with* $\text{depth}(\text{atom}(L\sigma)) \leq d + \epsilon$. *Let Γ contain no ground instance of* $\text{comp}(L)$, *i.e., for all $K \in \Gamma$ it holds that K is not unifiable with* $\text{comp}(L)$. *Then $M \models L$.*

Proof. Proof by contradiction. We assume that all our assumptions hold, but that $M \not\models L$. By Definition 2.3 and Lemmas 3 and 4, $M \not\models L$ if there exists a $K \in M$ that is unifiable with $\text{comp}(L)$. Moreover, Γ contains no ground instances of $\text{comp}(L)$ by assumption. We will now prove by induction that we can only reach states $(\Gamma'; \Delta; \mathcal{I}; M)$ where $A \in \mathcal{I}$ is unifiable with $\text{atom}(L)$ if A has depth $\leq d + \epsilon$ and Γ' contains all ground instances $L\sigma$ of L such that $\text{atom}(L\sigma)$ is also a ground instance of A and $\text{depth}(\text{atom}(L\sigma)) \leq d + \epsilon$. (Note that there always exists at least one such ground instance because A has depth $\leq d + \epsilon$.) This property guarantees that Clean can never be applied to an atom $A \in \mathcal{I}$ that is unifiable with $\text{atom}(L)$ and that Solve is only applicable to an atom $A \in \mathcal{I}$ that is unifiable with $\text{atom}(L)$ if there is also an instance $\text{atom}(L\sigma)$ of A in Γ' that ensures that we assign A with the correct polarity. The induction base holds trivially because in the state $(\Gamma; \emptyset; \{P(x_1, \ldots, x_n)\}; \emptyset)$ the only atom in $\mathcal{I}$ is $P(x_1, \ldots, x_n)$ and it has the minimal depth 1 and Γ contains by assumption all ground instances of L with depth $\leq d + \epsilon$. For the induction step, we assume

that $(\Gamma'; \Delta; \mathcal{I}; M)$ is a sound state that satisfies our property and prove that any direct successor state $(\Gamma''; \Delta'; \mathcal{I}'; M')$ must again satisfy our property. We prove this by case distinction:

1) Clean and Solve only remove elements A from $\mathcal{I}$ and all positive and negative instances of A from Γ''. This together with Definition 2.2 guarantees that the literals removed from Γ'' do not match with any of the remaining elements of $\mathcal{I}'$. Therefore, the property still holds.

2) Refine on A, but A is not unifiable with $\text{atom}(L)$. Trivial, because any of the new elements in $\mathcal{I}'$ will also not be unifiable with L.

3) Refine on A, A is unifiable with $\text{atom}(L)$, and $\text{depth}(A) \leq d + \epsilon$, and the position p of the refined variable has depth $|p| < d + \epsilon$. This means by induction that $\Gamma' = \Gamma''$ contains all ground instances $L\sigma$ of L such that $\text{atom}(L\sigma)$ is also a ground instance of A and $\text{depth}(\text{atom}(L\sigma)) \leq d + \epsilon$. Moreover, any new atom $A' \in \mathcal{I}' \setminus \mathcal{I}$ has at most depth $|p| + 1$ so still $\leq d + \epsilon$. And lastly, since A' is an instance of A, Γ' contains all ground instances $L\sigma$ of L such that $\text{atom}(L\sigma)$ is also a ground instance of A' and $\text{depth}(\text{atom}(L\sigma)) \leq d + \epsilon$.

4) Refine on A, A is unifiable with $\text{atom}(L)$, $\text{depth}(A) = d + \epsilon$, and the position p of the refined variable has depth $|p| = d+\epsilon$. Let $A^* = \text{mgu}(A, \text{atom}(L))$ as well as $L^* = A^*$ and $L' = A$ if L is positive or else $L^* = \neg A^*$ and $L' = \neg A$. This means by induction that $\Gamma' = \Gamma''$ contains all ground instances $L^*\sigma$ with $\text{depth}(\text{atom}(L^*\sigma)) \leq d + \epsilon$ and that they all have the same polarity as L. Moreover, any variable in A has either a position q with depth $|q| = d + \epsilon$ or there exist no $(A\tau), (\neg A\tau') \in \Gamma'$ such that $(A\tau)|_q$ and $(A\tau')|_q$ are not unifiable. However, this means we also know that any ground instance $(L^*[x_q]_q)\sigma$ of $(L^*[x_q]_q)$ must be in $\Gamma' = \Gamma''$ if q is the variable position of x_q in A with $|q| < d+\epsilon$. Note that due to linearity of L and A (and assuming disjoint variables) $A^*|_q \neq A|_q$ if and only if there exist q', q'' such that $q = q'q''$, $A|_{q'}$ is the position of a variable $x_{q'}$, $|q'| < d+\epsilon$, $L|_{q'}$ is defined and not a variable, and $A^*|_{q'} = L|_{q'}$. This means that we get L'/A if we replace all positions q in L^*/A^* with $A|_q$ if $A|_q = x_q$ and $|q| < d + \epsilon$. If we use this together with the previous fact for all variable positions $|q| < d + \epsilon$, then we get that any ground instance $L'\sigma$ of L' must be in $\Gamma' = \Gamma''$ and therefore Refine is not applicable. A contradiction.

5) Refine on A, A is unifiable with $\text{atom}(L)$, and $\text{depth}(A) > d+\epsilon$. This case is impossible by induction hypothesis! □

The preconditions of Theorem 10 may look unrelated to its conlusion at first sight. The first example shows why Γ needs to contain all ground instances $L\sigma$ of L of depth $\text{depth}(L)$. The reason is that Refine may lead to an atom K in $\mathcal{I}$ that is unifiable with $\text{comp}(L)$ but Γ contains no ground instances of $\text{comp}(K)$. In our example this is $P(x, f(y))$. The second example shows why Γ needs to contain all ground instances $L\sigma$ of L of depth $\text{depth}(L) + 1$ if L has a constant at a position p with depth $d = \text{depth}(L)$. The reason is that an application of Refine may lead to an atom K in $\mathcal{I}$ that is unifiable with $\text{comp}(L)$ but has no ground instances of $\text{comp}(K)$. In our example this is $P(f(x), y)$.

Example 11. (1) Let $\Gamma = \{\neg P(a, a), \neg P(f(a), a), P(a, f(a))\}$ with signature $\Sigma = (\{a/0, f/1\}, \{P\})$. Then for the input state $(\Gamma; \emptyset; P(x, y); \emptyset)$ the calculus

returns the model $M = \{\neg P(x,a), P(x,f(y))\}$ because we first need to apply Refine to position 2. Although for $\neg P(f(x),y)$ there is no inconsistent atom in Γ, $M \not\models \neg P(f(x),y)$.

(2) Let $\Gamma = \{\neg P(a,a), \neg P(a,b), \neg P(b,a), P(b,b)\}$ with signature $\Sigma = (\{a/0, b/0, f/1\}, \{P\})$. Then for the input state $(\Gamma; \emptyset; P(x,y); \emptyset)$ the calculus can return the model $M = \{\neg P(a,y), P(f(x),y), \neg P(b,a), P(b,b), P(b,f(y))\}$ if we first apply Refine to position 1. And although for $\neg P(x,a)$ there is no inconsistent atom in Γ, $M \not\models \neg P(x,a)$.

Theorem 12 (Non-ground Guarantees by Clean). *Let Γ be a consistent set of ground literals. Let M be a model generated by $\Rightarrow_{mod}$ from Γ. Let d be the maximal depth of any negative literal in Γ. Let A be a linear atom with* $\mathrm{depth}(A) \leq d$. *Let Γ contain all ground instances of $A\sigma$ with* $\mathrm{depth}(A\sigma) \leq d$. *Then $M \models A$.*

Proof. Note that the most general unifier of any two linear literals K, K' has depth $\mathrm{depth}(K\,\mathrm{mgu}(K,K')) = \max(\mathrm{depth}(K), \mathrm{depth}(K'))$. Firstly, we show that rule Solve can never add a negative literal $\neg B$ to the model that is unifiable with $\neg A$. In this case, Solve is only applicable if Γ contains a ground instance $\neg B\sigma$ and no ground instance $B\sigma$. The first condition implies $\mathrm{depth}(\neg B) \leq d$ and if B and A are unifiable $A\,\mathrm{mgu}(A,B)$ has depth $\leq d$. However, since Γ contains all ground instances of A with depth $\leq d$ this also means Γ contains all ground instances of $A\,\mathrm{mgu}(A,B)$ with depth $\leq d$. This means that our assumptions guarantee that the second condition for Solve is not satisfied if $\neg B$ is unifiable with $\neg A$. So Solve will not add a literal $\neg B$ that is unifiable with $\neg A$. Secondly, in addition to Solve, only Clean adds literals to M. All literals added by Clean are atoms, so they cannot unify with $\neg A$. Hence, $M \models L$. □

Lemma 13 (Lower Bound for SCL Refutations). *Let Γ be the ground partial model of an SCL stuck state for the input clause set N and bounded by $\preceq$ and β. This means in particular that (i) every literal $L \in \Gamma$ is ground and bounded by $\preceq$ and β (i.e., $L \preceq \beta$), (ii) every ground atom $A \preceq \beta$ is defined by Γ (i.e., $A \in \Gamma$ or $\neg A \in \Gamma$), and (iii) for every clause $C \in N$ and every grounding σ of C either $\Gamma \models C\sigma$ or there exists a literal $L \in C\sigma$ such that $L \not\preceq \beta$. Let M be a complete interpretation (i.e., for every ground atom A, $M \models A$ or $M \models \neg A$) that models Γ (i.e., $M \models \Gamma$) but not the clause set N (i.e., $M \not\models N$). Let β' be a smallest ground literal according to $\preceq$ such that there exists a clause $C \in N$, a grounding τ, where $L \preceq \beta'$ holds for any literal $L \in C\tau$, and $M \not\models C\tau$. Then there exists no $\beta^* \prec \beta'$ such that an SCL run on N and bounded by $\prec$ and β^* finds a refutation.*

Proof. The assumptions for β' and the completeness of M imply that $M \models C\sigma$ for all clauses $C \in N$ and all groundings σ, where $L \preceq \beta^*$ for any literal $L \in C\sigma$. This means one valid SCL run for N, $\preceq$, and β^* can simply decide all ground atoms $A \preceq \beta^*$ according to M, i.e., according to whether $M \models A$ or $M \models \neg A$, without encountering any conflicts and ending in a stuck state with a set Γ' such that $\Gamma' \models \mathrm{gnd}^{\preceq\beta^*}(N)$, where the function $\mathrm{gnd}^{\preceq\beta^*}$ computes the set of all

ground instances of a clause set where the grounding is restricted to produce literals L with $L \preceq \beta^*$. The existence of this stuck state proves that $\mathrm{gnd}^{\preceq\beta^*} N$ is satisfiable and that there exists no refutation for it. Hence, no SCL run for N, $\preceq$, and β^* can find a refutation. □

5 Conclusion and Future Work

Explicit model building is always a compromise between the expressivity of the used language, its computational properties and the effort to actually compute the model. Satisfiability of first-order logic clause sets is not even semi-decidable, so there cannot be a general solution. In the context of SCL, efficient model building and efficient clause evaluation are important aspects and our quite simple model building language, namely complete linear literal interpretations, nicely serves these two purposes. Still there may be room for improvement. For example, the three clauses

$$\begin{array}{cc} \neg R(x,x) & R(x,g(x)) \\ \neg R(x,y) \vee \neg R(y,z) \vee R(x,z) & \end{array}$$

do not have a finite model. Linear literal interpretations have the finite model property so there cannot be a finite representation of a model within this language. It needs a more expressive language. For example, assuming an additional constant a and a bound $\beta = R(g(a), g(g(a)))$ a partial model computed by SCL would be

$$[\neg R(a,a), R(a,g(a)), R(g(a),g(g(a))), R(a,g(g(a))), \neg R(g(a),g(a)), \neg R(g(a),a)^1]$$

The respective overall model could be represented by the linear Horn clause set

$$\begin{array}{cc} \neg R(a,a) & R(a,g(a)) \\ \neg R(x,y) \rightarrow \neg R(g(x),y) & R(x,y) \rightarrow R(x,g(y)) \\ \neg R(x,y) \rightarrow \neg R(g(x),g(y)) & R(x,y) \rightarrow R(g(x),g(y)) \end{array}$$

or by terms with exponents and constraints [4,13]

$$i \geq j \parallel \neg R(g^i(a), g^j(a)) \qquad i < j \parallel R(g^i(a), g^j(a))\,.$$

However, it is an open question how such representations can be actually computed out of a set of ground literals and how they can be used to efficiently test validity of clauses.

The rule Clean may actually add the respective literal wither positively or negatively to M. In practice, such literals could be marked in M. Then in case of starting from an SCL stuck trail where M is not a model for the clause set, a small but useful extension is to check whether flipping the sign of some of these literals turn M into a model.

In summary, we have presented an algorithm that computes in polynomial time out of a finite consistent set of ground literals Γ a complete linear literal

interpretation M such that $M \models \Gamma$. Furthermore, M can be effectively used to evaluate clauses and to determine a minimal extension to the ground literal restriction β out of an SCL stuck state.

Acknowledgements. We thank our anonymous reviewers for their constructive comments.

References

1. Armando, A., Bonacina, M.P., Ranise, S., Schulz, S.: New results on rewrite-based satisfiability procedures. ACM Trans. Comput. Logic **10**(1), 4:1–4:51 (2009)
2. Bachmair, L., Ganzinger, H., Waldmann, U.: Superposition with simplification as a decision procedure for the monadic class with equality. In: Gottlob, G., Leitsch, A., Mundici, D. (eds.) KGC 1993. LNCS, vol. 713, pp. 83–96. Springer, Heidelberg (1993). https://doi.org/10.1007/BFb0022557
3. Baumgartner, P., Fuchs, A., de Nivelle, H., Tinelli, C.: Computing finite models by reduction to function-free clause logic. J. Appl. Log. **7**(1), 58–74 (2009)
4. Bensaid, H., Peltier, N.: A complete superposition calculus for primal grammars. J. Autom. Reason. **53**(4), 317–350 (2014)
5. Bromberger, M., Desharnais, M., Weidenbach, C.: An Isabelle/HOL formalization of the SCL(FOL) calculus. In: Pientka, B., Tinelli, C. (eds.) Automated Deduction - CADE 29 - 29th International Conference on Automated Deduction. LNCS, vol. 14132, pp. 116–133. Springer (2023). https://doi.org/10.1007/978-3-031-38499-8_7
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 (2022)
8. Bromberger, M., Schwarz, S., Weidenbach, C.: Exploring partial models with SCL. In: Piskac, R., Voronkov, A. (eds.) Proceedings of 24th International Conference on Logic for Programming, Artificial Intelligence and Reasoning. EPiC Series in Computing, vol. 94, pp. 48–72. EasyChair (2023). https://doi.org/10.29007/8br1
9. Bromberger, M., Schwarz, S., Weidenbach, C.: SCL(FOL) revisited (2023). https://doi.org/10.48550/ARXIV.2302.05954, https://arxiv.org/abs/2302.05954
10. Caferra, R., Leitsch, A., Peltier, N.: Automated Model Building, Applied Logic Series, vol. 31. Kluwer (2004)
11. Cantone, D., Cutello, V.: A decidable fragment of the elementary theory of relations and some applications. In: Watanabe, S., Nagata, M. (eds.) Symbolic and Algebraic Computation, Proceedings of the International Symposium, Tokyo, Japan, pp. 24–29. ACM Press (August 1990)
12. Claessen, K., Soerensson, N.: New techniques that improve MACE-style finite model finding. In: Proceedings of the CADE-19 Workshop: Model Computation - Principles, Algorithms, Applications (2003)

13. Comon, H.: On unification of terms with integer exponents. Math. Syst. Theory **28**(1), 67–88 (1995)
14. Comon-Lundh, H., Cortier, V.: New decidability results for fragments of first-order logic and application to cryptographic protocols. In: Nieuwenhuis, R. (ed.) RTA 2003. LNCS, vol. 2706, pp. 148–164. Springer, Heidelberg (2003). https://doi.org/10.1007/3-540-44881-0_12
15. Fermüller, C.: A resolution variant deciding some classes of clause sets. In: Börger, E., Kleine Büning, H., Richter, M.M., Schönfeld, W. (eds.) CSL 1990. LNCS, vol. 533, pp. 128–144. Springer, Heidelberg (1991). https://doi.org/10.1007/3-540-54487-9_56
16. Fermüller, C.G., Leitsch, A.: Model building by resolution. In: Börger, E., Jäger, G., Kleine Büning, H., Martini, S., Richter, M.M. (eds.) CSL 1992. LNCS, vol. 702, pp. 134–148. Springer, Heidelberg (1993). https://doi.org/10.1007/3-540-56992-8_10
17. Fermüller, C.G., Leitsch, A.: Hyperresolution and automated model building. J. Log. Comput. **6**(2), 173–203 (1996)
18. Fermüller, C.G., Leitsch, A., Hustadt, U., Tamet, T.: Resolution decision procedures. In: Robinson, A., Voronkov, A. (eds.) Handbook of Automated Reasoning, vol. II, chap. 25, pp. 1791–1849. Elsevier (2001)
19. Fermüller, C., Leitsch, A., Tammet, T., Zamov, N. (eds.): Resolution Methods for the Decision Problem. LNCS, vol. 679. Springer, Heidelberg (1993). https://doi.org/10.1007/3-540-56732-1
20. Fermüller, C.G., Pichler, R.: Model representation over finite and infinite signatures. J. Log. Comput. **17**(3), 453–477 (2007)
21. 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
22. Ganzinger, H., de Nivelle, H.: A superposition decision procedure for the guarded fragment with equality. In: LICS, pp. 295–304 (1999)
23. Ge, Y., de Moura, L.: Complete instantiation for quantified formulas in satisfiabiliby modulo theories. In: Bouajjani, A., Maler, O. (eds.) CAV 2009. LNCS, vol. 5643, pp. 306–320. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-02658-4_25
24. Gebser, M., Sabuncu, O., Schaub, T.: An incremental answer set programming based system for finite model computation. AI Commun. **24**(2), 195–212 (2011)
25. Georgieva, L., Hustadt, U., Schmidt, R.A.: A new clausal class decidable by hyperresolution. In: Voronkov, A. (ed.) CADE 2002. LNCS (LNAI), vol. 2392, pp. 260–274. Springer, Heidelberg (2002). https://doi.org/10.1007/3-540-45620-1_21
26. Goubault-Larrecq, J.: Deciding $\mathcal{H}_1$ by resolution. In: Information Processing Letters, pp. 401–408 (2005)
27. 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
28. Horbach, M., Weidenbach, C.: Decidability results for saturation-based model building. In: Schmidt, R.A. (ed.) CADE 2009. LNCS (LNAI), vol. 5663, pp. 404–420. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-02959-2_30
29. Hustadt, U., Schmidt, R.A.: On evaluating decision procedures for modal logics. In: Proceedings of 15th International Joint Conference on Artificial Intelligence, IJCAI-97, pp. 202–207 (1997)
30. Hustadt, U., Schmidt, R.A., Georgieva, L.: A survey of decidable first-order fragments and description logics. J. Relational Methods Comput. Sci. **1**, 251–276 (2004)

31. Jacquemard, F., Meyer, C., Weidenbach, C.: Unification in extensions of shallow equational theories. In: Nipkow, T. (ed.) RTA 1998. LNCS, vol. 1379, pp. 76–90. Springer, Heidelberg (1998). https://doi.org/10.1007/BFb0052362
32. Janota, M., Suda, M.: Towards smarter MACE-style model finders. In: Barthe, G., Sutcliffe, G., Veanes, M. (eds.) LPAR-22. 22nd International Conference on Logic for Programming, Artificial Intelligence and Reasoning, Awassa, Ethiopia, 16-21 November 2018. EPiC Series in Computing, vol. 57, pp. 454–470. EasyChair (2018)
33. Joyner, W.H., Jr.: Resolution strategies as decision procedures. J. ACM **23**(3), 398–417 (1976)
34. Jr., R.J.B., Schrag, R.: Using CSP look-back techniques to solve real-world SAT instances. In: Kuipers, B., Webber, B.L. (eds.) Proceedings of the Fourteenth National Conference on Artificial Intelligence and Ninth Innovative Applications of Artificial Intelligence Conference, AAAI 97, IAAI 97, 27-31 July 1997, Providence, Rhode Island, USA, pp. 203–208 (1997)
35. Kovács, L., Voronkov, A.: First-order theorem proving and Vampire. In: Sharygina, N., Veith, H. (eds.) CAV 2013. LNCS, vol. 8044, pp. 1–35. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-39799-8_1
36. Lamotte-Schubert, M., Weidenbach, C.: BDI: a new decidable clause class. J. Log. Comput. **27**(2), 441–468 (2017)
37. Leidinger, H., Weidenbach, C.: SCL(EQ): SCL for first-order logic with equality. In: Blanchette, J., Kovács, L., Pattinson, D. (eds.) Automated Reasoning - 11th International Joint Conference, IJCAR 2022. LNCS, vol. 13385, pp. 228–247. Springer (2022). https://doi.org/10.1007/978-3-031-10769-6_14
38. Leitsch, A.: Deciding Horn classes by hyperresolution. In: Börger, E., Büning, H.K., Richter, M.M. (eds.) CSL 1989. LNCS, vol. 440, pp. 225–241. Springer, Heidelberg (1990). https://doi.org/10.1007/3-540-52753-2_42
39. Lynch, C.: Schematic saturation for decision and unification problems. In: Baader, F. (ed.) CADE 2003. LNCS (LNAI), vol. 2741, pp. 427–441. Springer, Heidelberg (2003). https://doi.org/10.1007/978-3-540-45085-6_37
40. Marques-Silva, J., Lynce, I., Malik, S.: Conflict-driven clause learning SAT solvers. In: Biere, A., Heule, M., van Maaren, H., Walsh, T. (eds.) Handbook of Satisfiability - Second Edition, Frontiers in Artificial Intelligence and Applications, vol. 336, pp. 133–182. IOS Press (2021)
41. McCune, W.: Mace4 reference manual and guide. CoRR **cs.SC/0310055** (2003)
42. Moskewicz, M.W., Madigan, C.F., Zhao, Y., Zhang, L., Malik, S.: Chaff: engineering an efficient SAT solver. In: Design Automation Conference 2001. Proceedings, pp. 530–535. ACM (2001)
43. Nieuwenhuis, R.: Basic paramodulation and decidable theories (extended abstract). In: Proceedings 11th IEEE Symposium on Logic in Computer Science, LICS 1996, pp. 473–482. IEEE Computer Society Press (1996)
44. de Nivelle, H., de Rijke, M.: Deciding the guarded fragments by resolution. J. Symb. Comput. **35**(1), 21–58 (2003)
45. Pichler, R.: Algorithms on atomic representations of herbrand models. In: Dix, J., del Cerro, L.F., Furbach, U. (eds.) JELIA 1998. LNCS (LNAI), vol. 1489, pp. 199–215. Springer, Heidelberg (1998). https://doi.org/10.1007/3-540-49545-2_14
46. Schmidt, R.A., Hustadt, U.: First-order resolution methods for modal logics. In: Voronkov, A., Weidenbach, C. (eds.) Programming Logics. LNCS, vol. 7797, pp. 345–391. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-37651-1_15

47. Schulz, S., Cruanes, S., Vukmirović, P.: Faster, higher, stronger: E 2.3. In: Fontaine, P. (ed.) CADE 2019. LNCS (LNAI), vol. 11716, pp. 495–507. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-29436-6_29
48. Shumsky, O., Wilkerson, R.W., McCune, W., Erçal, F.: Direct finite first-order model generation with negative constraint propagation heuristic. In: Bryant, B.R., Carroll, J.H., Oppenheim, D., Hightower, J., George, K.M. (eds.) Proceedings of the 1997 ACM symposium on Applied Computing, SAC 1997, San Jose, CA, USA, 28 February - 1 March, pp. 25–29. ACM (1997)
49. Silva, J.P.M., Sakallah, K.A.: Grasp - a new search algorithm for satisfiability. In: International Conference on Computer Aided Design, ICCAD, pp. 220–227. IEEE Computer Society Press (1996)
50. Slaney, J.: FINDER: finite domain enumerator system description. In: Bundy, A. (ed.) CADE 1994. LNCS, vol. 814, pp. 798–801. Springer, Heidelberg (1994). https://doi.org/10.1007/3-540-58156-1_63
51. Slaney, J.K., Surendonk, T.: Combining finite model generation with theorem proving: Problems and prospects. In: Baader, F., Schulz, K.U. (eds.) Frontiers of Combining Systems, First International Workshop FroCoS 1996, Munich, Germany, 26-29 March 1996, Proceedings. Applied Logic Series, vol. 3, pp. 141–155. Kluwer Academic Publishers (1996)
52. Sturm, T., Voigt, M., Weidenbach, C.: Deciding first-order satisfiability when universal and existential variables are separated. In: Grohe, M., Koskinen, E., Shankar, N. (eds.) Proceedings of the 31st Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2016, New York, USA, 5-8 July 2016, pp. 86–95. ACM (2016)
53. Suda, M., Weidenbach, C., Wischnewski, P.: On the saturation of YAGO. In: Giesl, J., Hähnle, R. (eds.) IJCAR 2010. LNCS (LNAI), vol. 6173, pp. 441–456. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-14203-1_38
54. Teucke, A., Weidenbach, C.: Decidability of the monadic shallow linear first-order fragment with straight dismatching constraints. In: de Moura, L. (ed.) CADE 2017. LNCS (LNAI), vol. 10395, pp. 202–219. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-63046-5_13
55. Weidenbach, C.: Towards an automatic analysis of security protocols in first-order logic. In: CADE 1999. LNCS (LNAI), vol. 1632, pp. 314–328. Springer, Heidelberg (1999). https://doi.org/10.1007/3-540-48660-7_29
56. Weidenbach, C., Dimova, D., Fietzke, A., Kumar, R., Suda, M., Wischnewski, P.: SPASS version 3.5. In: Schmidt, R.A. (ed.) CADE 2009. LNCS (LNAI), vol. 5663, pp. 140–145. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-02959-2_10