On the Expressivity and Applicability of Model Representation Formalisms

Andreas Teucke, Marco Voigt, Christoph Weidenbach

On the Expressivity and Applicability of Model Representation Formalisms

Andreas Teucke[1,2], Marco Voigt[1,2], and Christoph Weidenbach[1]

[1] Max-Planck Institut für Informatik, Saarland Informatics Campus E1 4 66123 Saarbrücken Germany
[2] Graduate School of Computer Science, Saarbrücken, Germany

Abstract. A number of first-order calculi employ an explicit model representation formalism in support of non-redundant inferences and for detecting satisfiability. Many of these formalisms can represent infinite Herbrand models. The first-order fragment of monadic, shallow, linear, Horn (MSLH) clauses, is such a formalism used in the approximation refinement calculus (AR). Our first result is a finite model property for MSLH clause sets. Therefore, MSLH clause sets cannot represent models of clause sets with inherently infinite models. Through a translation to tree automata, we further show that this limitation also applies to the linear fragments of implicit generalizations, which is the formalism used in the model-evolution calculus (ME), to atoms with disequality constraints, the formalisms used in the non-redundant clause learning calculus (NRCL), and to atoms with membership constraints, a formalism used for example in decision procedures for algebraic data types. Although these formalisms cannot represent models of clause sets with inherently infinite models, through an additional approximation step they can. This is our second main result. For clause sets including the definition of an equivalence relation with the help of an additional, novel approximation, called reflexive relation splitting, the approximation refinement calculus can automatically show satisfiability through the MSLH clause set formalism.
This work was published at FroCoS 2019 [25].

1 Introduction

Proving satisfiability of a first-order clause set is more difficult than proving unsatisfiability, in general. Still, for many applications the detection of failing refutations by establishing a counter model is more than desirable. In the past, several methods, calculi and systems have been presented that can detect satisfiability of a clause set, in particular, if there is a finite model that is not too large. The approaches can be separated into the following classes:

(1) the model building is integrated into a first-order calculus or a decision procedure for some fragment, directly operating on the first-order clause set, complete for unsatisfiability, e.g., [8,3,19,23,1,5,6],

(2) the model building is integrated into a first-order calculus that operates on the first-order clause set modulo an approximation, complete for unsatisfiability, e.g., [16,28,14],
(3) the model building aims at finding finite models without being complete for unsatisfiability, e.g., [21,17,9],
(4) the model building aims at finding finite and infinite models without being complete for unsatisfiability, e.g., [18],

where superposition [2] does not belong to any of the above classes, because the model building is implicit and reached by a finite saturation of the clause set modulo inferences and the elimination of redundant clauses.

The approaches in classes (1) and (2) select inferences with respect to the explicit (partial) model by identifying a false clause (instance). Therefore, the representation of models needs to be effective, e.g., falsity of a clause (instance) with respect to the model needs to be (efficiently) decidable.

For superposition it is undecidable whether a clause is false with respect to a saturated clause set, in general. This can be seen by a reduction through the Post Correspondence Problem (PCP) [20]. The clause set consisting of $R(\epsilon, \epsilon)$ and clauses $R(x, y) \rightarrow R(t_i[x], s_i[y])$ where the t_i, s_i are terms built over the monadic functions g, h and variables x, y, respectively, is saturated with strictly maximal atoms $R(t_i[x], s_i[y])$ and encodes the words $(\mathtt{w}_1, \ldots, \mathtt{w}_n)$, $(\mathtt{v}_1, \ldots, \mathtt{v}_n)$ generated by a PCP over letters g, h. That means words are represented by nestings of monadic functions. The PCP has a solution iff a ground atom $R(g(t), g(t))$ or $R(h(t), h(t))$ is a consequence of the above clause set. This corresponds to testing whether one of the clauses $R(g(x), g(x))$ or $R(h(x), h(x))$ has a false instance with respect to the implicit model of the saturated PCP clause set.

Reasoning with respect to a (partial) model assumption has advantages. The superposition completeness proof shows that an inference with a clause that is false in the current partial model is not redundant [2]. This has meanwhile also been shown for the CDCL [31] and the NRCL [1] calculus. The non-redundant inference property might also hold for other calculi of classes (1) and (2). It requires exhaustive model generation and eager conflict detection.

Our first contribution is showing that the model representation used in [28], monadic shallow linear Horn clauses (MSLH) has the finite model property, Section 3. This means that if a finite MSLH clause set has a model, it also has a finite model. Hence, MSLH clause sets cannot be used directly to represent models of clause sets with inherently infinite models. A further consequence is that any calculus in class (1), where the model representation can be represented by an MSLH clause set, cannot terminate on satisfiable clause sets with inherently infinite models. A more detailed discussion of this aspect is contained in Section 4.

The fact that MSLH clause sets have the finite model property does not mean that the approximation refinement (AR) calculus presented in [28] cannot be used for finding infinite models of clause sets with inherently infinite models. The reason is that the MSLH model representation in [28] does not directly relate to a model of the original clause set, but via an approximation. For the

approximation it is shown in a constructive way that it preserves satisfiability. This is done modulo the *minimal* Herbrand model of a saturated MSLH clause set. Such Herbrand models become infinite as soon as there are non-constant function symbols. So the question is whether AR can actually terminate on clause sets with inherently infinite models. In Section 5, we show that this is the case for certain classes of such clause sets relying on reflexivity of a binary (equivalence) relation. The technique we propose is an additional approximation called *reflexive relation splitting*. A similar relationship between a clause set and its approximation was already observed in [18] where an approximation of a first-order clause set into a class of tree automata is used in order to find finite and infinite models.

Our results concerning the MSLH fragment and the reflexive relation splitting modulo the AR calculus can be demonstrated by the following example. Consider the following three clauses defining a reflexive binary relation R (see [8], page 55 for further discussion of this example).

$$\{R(x,x), \quad R(g(x),g(y)) \to R(x,y), \quad \neg R(g(x),c)\}$$

This set has only infinite models. No resolution inference between $R(x,x)$ and $\neg R(g(x),c)$ is possible. Following the AR approach [28], the MSLH clause set

$$\{T(f_R(x,y)), \quad T(f_R(g(x),g(y))) \to T(f_R(x,y)), \quad \neg T(f_R(g(x),c))\}$$

is generated. We write unit clauses as single literals, and non-unit clauses as implications. The relation R is translated into a binary function f_R over a monadic predicate T. The approximation is the replacement of $R(x,x)$ by $T(f_R(x,y))$, where now the connection between the non-linear occurrences of x is lost. As a consequence, a refutation containing a resolution step between $T(f_R(x,y))$ and $\neg T(f_R(g(x),c))$ with substitution $\{x \mapsto g(v), y \mapsto c\}$ is possible, which cannot be lifted to the original clause set because $g(v)$ and c are not unifiable. The refinement then excludes this particular instance by generating $R(g(x),g(x))$, however, after approximating this clause, the empty clause can be derived again. This time the derivation also uses the second clause, where the substitution instance of the refutation contains one further nesting of g. The approximation refinement approach does not terminate on this example.

If in the approximation the inference between $T(f_R(x,x))$ and $\neg T(f_R(g(x),c))$ can be blocked, saturation will terminate without finding a contradiction. As said, in the original clause set this inference is not possible, because of the non-linear occurrence of x. Now the idea is to split the relation R into its reflexive and irreflexive part, denoted by the two predicates R_{ref} and R_{irr}. The original clause set is satisfiable if and only if the following clause set is satisfiable

$$\{R_{\text{ref}}(x,x), \quad R_{\text{irr}}(g(x),g(y)) \to R_{\text{irr}}(x,y), \quad \neg R_{\text{irr}}(g(x),c)\},$$

details are explained in Section 5. After approximation it becomes

$$\{T(f_{R_{\text{ref}}}(x,y))^*, T(f_{R_{\text{irr}}}(g(x),g(y)))^+ \to T(f_{R_{\text{irr}}}(x,y)), \neg T(f_{R_{\text{irr}}}(g(x),c))^*\} \quad (\dagger)$$

where * highlights maximal and $^+$ selected literals of the ordered resolution calculus used to decide MSLH clauses [27,28]. There are no possible inferences generating further clauses, i.e. the set is already saturated.

The infinite minimal Herbrand model is $\mathcal{I} = \{T(f_{R_{\text{ref}}}(g^i(c), g^j(c))) \mid i,j \geq 0\}$ which is also a model for the clause set before approximation [27,28] by simply

undoing the shift of R_{irr}, R_{ref} to the function level: $\mathcal{I} = \{R_{\text{ref}}(g^i(c), g^j(c))) \mid i, j \geq 0\}$. Nestings of functions in the Herbrand model representing relations, e.g., $f_{R_{\text{ref}}}$, can be prevented by adding further MSLH clauses. We omit these here for simplicity. This model can then be translated, see the proof of Lemma 9, into the Herbrand model $\mathcal{I} = \{R(g^i(c), g^i(c))) \mid i \geq 0\}$ of the original clause set.

In Section 3, we prove a finite model property for saturated, satisfiable MSLH clause sets. For the example, see (†), the thus constructed model has the domain $\mathsf{A} := \{\mathsf{a}_c, \mathsf{a}^{(1)}, \mathsf{a}^{(2)}, \mathsf{a}^{(3)}, \mathsf{b}^{(1)}, \mathsf{b}^{(2)}, \mathsf{b}^{(3)}\}$. The predicate T is interpreted with the set $\{\mathsf{b}^{(1)}, \mathsf{b}^{(2)}, \mathsf{b}^{(3)}\}$. For the constant c we use the distinguished element a_c. The interpretation of the function $f_{R_{\text{ref}}}$ is given in the following function table:

$$
\begin{array}{rll}
\langle \mathsf{a}_c, \mathsf{a}_c \rangle & \longmapsto \mathsf{b}^{(1)} & \\
\langle \mathsf{a}^{(i)}, \mathsf{a}^{(i)} \rangle & \longmapsto \mathsf{b}^{(j)} & \text{for every } i \text{ and some } j \neq i \\
\langle \mathsf{b}^{(i)}, \mathsf{b}^{(i)} \rangle & \longmapsto \mathsf{b}^{(j)} & \text{for every } i \text{ and some } j \neq i \\
\langle \mathsf{c}, \mathsf{d} \rangle & \longmapsto \mathsf{a}^{(j)} & \text{for any } \mathsf{c}, \mathsf{d} \in \mathsf{A} \text{ with } \mathsf{c} \neq \mathsf{d} \text{ and} \\
 & & \text{some } j \text{ chosen such that for any } i, \\
 & & \text{if } \mathsf{c} \text{ or } \mathsf{d} \text{ is equal to } \mathsf{a}^{(i)} \text{ or } \mathsf{b}^{(i)}, \text{ then } j \neq i
\end{array}
$$

For the function $f_{R_{\text{irr}}}$ we get a similar function table in which every pair $\langle \mathsf{c}, \mathsf{d} \rangle$ is mapped to some $\mathsf{a}^{(j)}$, where j is chosen such that $\mathsf{c}, \mathsf{d} \neq \mathsf{a}^{(j)}$. Finally, the interpretation of the function g is given by $g(\mathsf{a}_c) = \mathsf{a}^{(1)}$, $g(\mathsf{a}^{(i)}) = \mathsf{a}^{(j)}$ and $g(\mathsf{b}^{(i)}) = \mathsf{a}^{(j)}$ for every i and some $j \neq i$.

The paper is now organized as follows: after fixing some notions and notations, Section 2, the finite model property of MSLH clause sets is shown in Section 3. Consequences of this result for other model representation formalisms are discussed in Section 4. In Section 5 reflexive relation splitting is introduced and its application to AR investigated. The present paper ends with a discussion on the obtained results and future research directions, Section 6. Due to space limitations, not all proof details could be included. The interested reader will find the full details in the extended preprint [26].

2 Preliminaries

We consider a standard first-order language without equality where letters v, w, x, y, z denote variables, f, g, h functions, a, b, c constants, s, t terms, and Greek letters σ, τ, ρ are used for substitutions. S, P, Q, R denote predicates, A, B atoms, E, K, L literals, C, D clauses, N clause sets and $\mathcal{V}$ sets of variables. The notation $[\neg]A$ denotes A or its negation. The signature $\Sigma = (\mathcal{F}, \mathcal{P})$ consists of two disjoint, non-empty, in general infinite sets of function and predicate symbols $\mathcal{F}$ and $\mathcal{P}$, respectively. The set of all *terms* over the variables in $\mathcal{V}$ is $\mathcal{T}(\mathcal{F}, \mathcal{V})$. If there are no variables, then terms, literals and clauses are called *ground*, respectively. A *substitution* σ is denoted by pairs $\{x \mapsto t\}$. A substitution σ is a *grounding* substitution for a term, atom, literal, clause if the application of σ yields a ground term, ground atom, ground literal, ground clause, respectively.

The set of *free* variables of an atom A (term t, literal L, clause C) is denoted by $\text{vars}(A)$ ($\text{vars}(t)$, $\text{vars}(L)$, $\text{vars}(C)$). A predicate with exactly one argument is called *monadic*. A term is *complex* if it is not a variable and *shallow* if it is a

constant, a variable, or of the form $f(x_1, \ldots, x_n)$. A term, atom is called *linear* if there are no duplicate variable occurrences.

A *clause* is a multiset of literals which we write as an implication $\Gamma \to \Delta$ where the atoms in the multiset Δ (the *succedent*) denote the positive literals and the atoms in the multiset Γ (the *antecedent*) the negative literals. Alternatively, we write a clause also as a disjunction of its literals. We write $\Box$ for the empty clause. We abbreviate disjoint set union with sequencing, for example, we write $\Gamma, \Gamma' \to \Delta, L$ instead of $\Gamma \cup \Gamma' \to \Delta \cup \{L\}$. A clause $\Gamma \to \Delta$ is called an *MSLH* clause, if (i) Δ contains at most one atom, i.e., the clause is Horn, (ii) all occurring predicates are monadic, (iii) the argument of any monadic atom in Δ is shallow and linear. The first-order fragment consisting of finite MSLH clause sets we call *MSLH*.

An *atom ordering* $\prec$ is an irreflexive, well-founded, total ordering on ground atoms. It is lifted to literals by defining $A \prec \neg A \prec B$ for any atoms A, B with $A \prec B$. It is lifted to clauses by its multiset extension. The ordering is lifted from the ground level through ground instantiation: for two different atoms A, B containing variables, $A \prec B$ if $A\sigma \prec B\sigma$ for all grounding substitutions σ and the atoms are incomparable otherwise. A literal L is *maximal* (*strictly maximal*) in a clause $C \vee L$ if there is no literal $K \in C$ with $L \prec K$ ($L \preceq K$). The clause ordering is compatible with the atom ordering; if the maximal atom in C is greater than the maximal atom in D then $D \prec C$. We use $\prec$ simultaneously to denote an atom ordering and its multiset, literal, and clause extensions. For a ground clause set N and clause C, the set $N^{\prec C} = \{D \in N \mid D \prec C\}$ denotes the clauses of N smaller than C.

As usual, we interpret atoms, clauses, and clause sets with respect to *structures* $\mathcal{A}$, also called *interpretations*, consisting of a nonempty universe A and interpretations $c^{\mathcal{A}}$, $f^{\mathcal{A}}$, and $P^{\mathcal{A}}$ of all occurring constants, functions, and predicates. We often use a special kind of interpretations, called *Herbrand interpretations*, whose universe is the set of all ground terms. A *Herbrand interpretation* $\mathcal{I}$ is represented by a – possibly infinite – set of ground atoms. A ground atom A is *true* in $\mathcal{I}$ if $A \in \mathcal{I}$ and *false*, otherwise. $\mathcal{I}$ is said to *satisfy* a ground clause $C = \Gamma \to \Delta$, denoted by $\mathcal{I} \models C$, if $\Delta \cap \mathcal{I} \neq \emptyset$ or $\Gamma \not\subseteq \mathcal{I}$. A non-ground clause C is satisfied by $\mathcal{I}$ if $\mathcal{I} \models C\sigma$ for every grounding substitution σ. An interpretation $\mathcal{I}$ is called a *model* of N, $\mathcal{I} \models N$, if $\mathcal{I} \models C$ for every $C \in N$. A Herbrand model $\mathcal{I}$ of N is considered *minimal* (with respect to set inclusion) if there is no model $\mathcal{I}'$ with $\mathcal{I}' \subset \mathcal{I}$ and $\mathcal{I}' \models N$. A set of clauses N is *satisfiable*, if there exists a model that satisfies N. Otherwise, the set is *unsatisfiable*.

The superpositon calculus [2] restricted to first-order logic without equality results in the ordered resolution calculus together with the superposition redundancy criterion and partial model operator, see below. For ordered resolution, a selection function is assumed that may select negative literals in clauses. Then $(C \vee D)\sigma$ is an ordered resolution inference between a clause $C \vee A$ and a clause $D \vee \neg B$, if (i) σ is the mgu between A and B, (ii) $A\sigma$ is strictly maximal in $(C \vee A)\sigma$ and nothing is selected in $C \vee A$, (iii) $\neg B\sigma$ is maximal in $(D \vee \neg B)\sigma$ or selected. The clause $(C \vee A)\sigma$ is an ordered factoring inference on a clause

$C \vee A \vee A'$, if (i) σ is the mgu between A and A', (ii) $A\sigma$ is maximal in $(C \vee A)\sigma$ and nothing is selected in $C \vee A \vee A'$. Selection is stable under instantiation, i.e., if $\neg A$ is selected in $\neg A \vee C$ it is also selected in $(\neg A \vee C)\sigma$, for any substitution σ. A clause C is *redundant* with respect to a clause set N, if for all ground instances $C\sigma$ there are ground instances $D_1\sigma_1, \ldots, D_n\sigma_n$, $\{D_1, \ldots, D_n\} \subseteq N$, $D_i\sigma_i \prec C\sigma$ for all i, such that $D_1\sigma_1, \ldots, D_n\sigma_n \models C\sigma$, i.e., $C\sigma$ is implied by smaller ground instances from clauses in N. A clause set N is called *saturated* if all clauses generated by ordered resolution or ordered factoring from clauses in N are either redundant or contained in N. Given a ground clause set N and an ordering $\prec$ we can construct a (partial) Herbrand model $N_\mathcal{I}$ for N by the superposition (partial) model operator inductively as follows:

$$N_C := \bigcup_{D \prec C} \delta_D$$

$$\delta_D := \begin{cases} \{P(t_1, \ldots, t_n)\} & \text{if } D = D' \vee P(t_1, \ldots, t_n), P(t_1, \ldots, t_n) \text{ strictly maximal in } D, \text{ no literal selected in } D \text{ and } N_D \not\models D \\ \emptyset & \text{otherwise} \end{cases}$$

$$N_\mathcal{I} := \bigcup_{C \in N} \delta_C$$

Clauses C with $\delta_C \neq \emptyset$ are called *productive*. For a non-ground clause set N we define $N_\mathcal{I} := (\{C\sigma \mid C \in N, \sigma \text{ grounding for } C\})_\mathcal{I}$. The main completeness result of superposition is: for a clause set N let N^* be its (possibly infinite) saturation, then either $\square \in N^*$ and N is unsatisfiable, or $N^*_\mathcal{I} \models N$ [2].

Basically, inferences of the superposition calculus are restricted to maximal, or selected negative literals. If all non-redundant inferences of a clause set are performed, i.e., the clause set is saturated, then the superposition model operator generates an overall model for the clause set.

3 MSLH Model Properties

By definition, Herbrand models for MSLH clause sets with non-constant function symbols have an infinite domain. In what follows we show how to construct non-Herbrand models with finite domains for satisfiable finite MSLH clause sets. The constructed model is a finite representation of the *minimal Herbrand model*.

Consider a satisfiable finite MSLH clause set N. It is known that N can be finitely saturated using superposition (ordered resolution) with an appropriate ordering and selection strategy such that the following property holds for the obtained saturated clause set N^* [30]. Every clause C in N^* that is productive in the sense of the superposition model operator has the form $C = P_1(x_1), \ldots, P_n(x_n) \rightarrow S(f(y_1, \ldots, y_m))$ where $\{x_1, \ldots, x_n\} \subseteq \{y_1, \ldots, y_m\}$, $f(y_1, \ldots, y_m)$ is linear, and $S(f(y_1, \ldots, y_m))$ is strictly maximal in C. Such a saturation can, e.g., be obtained by choosing for $\prec$ a Knuth-Bendix-Ordering (KBO) with weight one for all function symbols, variables, and a selection strategy that selects a negative literal $P_i(t_i)$ in any clause $P_1(t_1), \ldots, P_n(t_n) \rightarrow S(f(y_1, \ldots, y_m))$ if t_i is not a variable, if t_i is a variable that does not occur in $f(y_1, \ldots, y_m)$, or if t_i is a variable in a clause $P_1(x), \ldots, P_n(x) \rightarrow S(x)$ [30,29].

Proposition 1 (Entailed by Lemma 4 from [30]). *Consider a satisfiable finite MSLH clause set N. There is a finite MSLH clause set N^* such that $N \subseteq N^*$ and $N \models N^*$ and there is a (minimal) Herbrand model $\mathcal{H} \models N^*$ such that for every ground atom A of the form $S\big(f(s_1, \ldots, s_m)\big)$ we have $\mathcal{H} \models A$ only if there is some clause C in N^* and a variable assignment β with the following properties (notice that for $m = 0$ f degenerates to a constant symbol):*

(a) C has the form $P_1(x_1), \ldots, P_n(x_n) \to S\big(f(y_1, \ldots, y_m)\big)$ where $\{x_1, \ldots, x_n\} \subseteq \{y_1, \ldots, y_m\}$, the $y_1, \ldots, y_m$ are pairwise distinct, and $m, n \geq 0$;
(b) we have $\beta(y_i) = s_i$ for every i, $1 \leq i \leq m$; and
(c) we have $\mathcal{H}, \beta \models P_j(x_j)$ for every j, $1 \leq j \leq n$.

Since N^* is satisfiable and all its clauses are Horn, it possesses a unique minimal Herbrand model $\mathcal{H}$ (cf. [12], Chapter XI, Theorem 3.8). The property described in Proposition 1 provides the key to construct a finite model for N and N^* from $\mathcal{H}$. The following example is intended to illustrate the ideas underlying the construction in a simplified form.

Example 2. Consider the following set of MSLH clauses with constants a and b:

$$\begin{aligned} N := \{ & P(a),\ Q(b), \quad \neg P(z) \vee \neg Q(z) \vee \neg R(z), \\ & \neg P(u) \vee \neg P(u') \vee P(f(u,u')), \quad \neg Q(v) \vee \neg Q(v') \vee Q(f(v,v')), \\ & \neg P(x) \vee R(f(x,y)), \quad \neg P(y) \vee R(f(x,y)), \\ & \neg Q(x) \vee R(f(x,y)), \quad \neg Q(y) \vee R(f(x,y))\} \ . \end{aligned}$$

The set N is satisfied by the minimal Herbrand interpretation $\mathcal{H}$ with

$$\begin{aligned} P^{\mathcal{H}} &:= \big\{a, f(a,a), f\big(a, f(a,a)\big), f\big(f(a,a), a\big), f\big(f(a,a), f(a,a)\big), \ldots\big\} \ , \\ Q^{\mathcal{H}} &:= \big\{b, f(b,b), f\big(b, f(b,b)\big), f\big(f(b,b), b\big), f\big(f(b,b), f(b,b)\big), \ldots\big\} \ , \\ R^{\mathcal{H}} &:= \big\{f(s,t) \;\big|\; s \in P^{\mathcal{H}} \text{ or } t \in Q^{\mathcal{H}}\big\} \ . \end{aligned}$$

The interpretation $\mathcal{H}$, together with $N^* := N$, satisfies the conditions of Proposition 1: for every term $f(s,t)$ that belongs to $R^{\mathcal{H}}$ we have that one of the clauses $\neg P(x) \vee R(f(x,y))$ or $\neg P(y) \vee R(f(x,y))$ or $\neg Q(x) \vee R(f(x,y))$ or $\neg Q(y) \vee R(f(x,y))$ enforces $\mathcal{H} \models R(f(s,t))$ because of $\mathcal{H} \models P(s)$ or $\mathcal{H} \models P(t)$ or $\mathcal{H} \models Q(s)$ or $\mathcal{H} \models Q(t)$, respectively. Similarly, the presence of any term $f(\ldots)$ in $P^{\mathcal{H}}$ or $Q^{\mathcal{H}}$ is enforced by one of the clauses $\neg P(u) \vee \neg P(u') \vee P(f(u,u'))$ or $\neg Q(v) \vee \neg Q(v') \vee Q(f(v,v'))$.

These requirements towards the minimality of $\mathcal{H}$ provide us with a certain knowledge about distinct terms $f(s,t)$ and $f(s',t')$. Suppose the terms s and s' are indistinguishable with respect to their membership in $P^{\mathcal{H}}, Q^{\mathcal{H}}, R^{\mathcal{H}}$. Further suppose that the same holds for the terms t and t'. Then, $f(s,t)$ and $f(s',t')$ are also indistinguishable with respect to their belonging to $P^{\mathcal{H}}, Q^{\mathcal{H}}$, and $R^{\mathcal{H}}$, because the arguments s, t and s', t' trigger the same productive clauses. A formal statement of this property is given in Lemma 3.

Based on this observation, we use $\mathcal{H}$ as a blueprint for a finite model $\mathcal{A}$, which is depicted in Figure 1. The domain of $\mathcal{A}$ shall be $\mathsf{A} := \{\mathsf{a}, \mathsf{b}, \mathsf{c}, \mathsf{d}, \mathsf{e}\}$,

and we set $a^{\mathcal{A}} := \mathsf{a}$ and $b^{\mathcal{A}} := \mathsf{b}$. The predicate symbols are interpreted by $P^{\mathcal{A}} := \{\mathsf{a}, \mathsf{c}\}, Q^{\mathcal{A}} := \{\mathsf{b}, \mathsf{d}\}, R^{\mathcal{A}} := \{\mathsf{c}, \mathsf{d}, \mathsf{e}\}$. Moreover, we define

$$f^{\mathcal{A}}(\mathsf{a},\mathsf{a}) := \mathsf{c} \qquad f^{\mathcal{A}}(\mathsf{a},\mathsf{c}) := \mathsf{c} \qquad f^{\mathcal{A}}(\mathsf{c},\mathsf{a}) := \mathsf{c} \qquad f^{\mathcal{A}}(\mathsf{c},\mathsf{c}) := \mathsf{c}$$
$$f^{\mathcal{A}}(\mathsf{b},\mathsf{b}) := \mathsf{d} \qquad f^{\mathcal{A}}(\mathsf{b},\mathsf{d}) := \mathsf{d} \qquad f^{\mathcal{A}}(\mathsf{d},\mathsf{b}) := \mathsf{d} \qquad f^{\mathcal{A}}(\mathsf{d},\mathsf{d}) := \mathsf{d}.$$

For all other inputs, $f^{\mathcal{A}}$ shall yield e as output. Every domain element in A represents one equivalence class of the terms in $\mathcal{H}$'s Herbrand domain with respect to membership in the sets $P^{\mathcal{H}}$, $Q^{\mathcal{H}}$, and $R^{\mathcal{H}}$. The domain element a represents the class $[a] := \{a\}$ of terms that belong to $P^{\mathcal{H}}$ and to no other set. Similarly, b represents $[b] := \{b\}$ of terms that belong to $Q^{\mathcal{H}}$ and to no other set. The element c represents the class of all terms belonging to $P^{\mathcal{H}} \cap R^{\mathcal{H}}$, i.e. to the class containing $f(a,a), f(a,f(a,a))$ and so on. The class of terms belonging to $Q^{\mathcal{H}} \cap R^{\mathcal{H}}$ is represented by d. Finally, e corresponds to the class of all terms that are member of $R^{\mathcal{H}}$ but of none of the other predicates, e.g. $f(a,b), f(a,f(b,a))$.

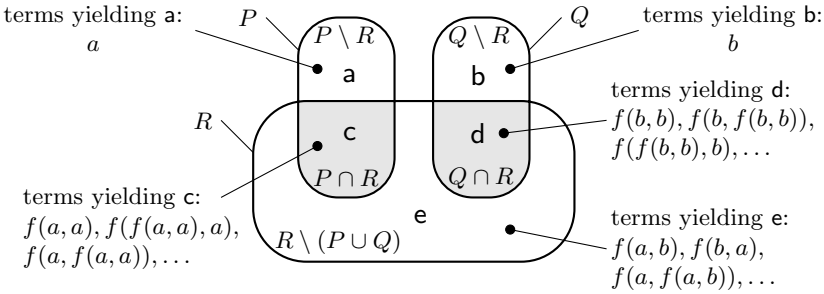

Fig. 1. Illustration of the model $\mathcal{A}$ of N from Example 2.

Next, we describe formally how to construct a finite model for the given satisfiable and finite MSLH clause set N. Let N^* and $\mathcal{H}$ be the objects described in Proposition 1. Then, we have $\mathcal{H} \models N^*$ and $\mathcal{H} \models N$. Let H be the domain of $\mathcal{H}$, i.e. H is the set of all ground terms over the vocabulary underlying N. We aim at constructing a finite model $\mathcal{A} \models N$ starting from $\mathcal{H}$.

Let Π denote the set of all predicates occurring in N, and recall that Π contains only unary predicate symbols. Let $\sim$ be the equivalence relation on H such that $s \sim t$ holds if and only if we have for every $P \in \Pi$ that $\mathcal{H} \models P(s)$ if and only if $\mathcal{H} \models P(t)$.

Lemma 3. *For every non-constant function symbol f in N of arity m and all tuples $\langle s_1, \ldots, s_m\rangle, \langle t_1, \ldots, t_m\rangle \in \mathsf{H}^m$ for which $s_i \sim t_i$ holds for every i we have $f(s_1, \ldots, s_m) \sim f(t_1, \ldots, t_m)$.*

Proof. By Definition of $\mathcal{H}$, $\mathcal{H} \models S\big(f(s_1, \ldots, s_m)\big)$ entails that there is a clause C of the form $\neg P_1(x_1) \vee \ldots \vee \neg P_n(x_n) \vee S\big(f(y_1, \ldots, y_m)\big)$ in N^* and a variable assignment β that satisfy Properties (a) to (c) from Proposition 1. Let

γ be a variable assignment for which we have $\gamma(y_i) := t_i$ for every i. Notice that such a γ with $\langle \gamma(y_1), \ldots, \gamma(y_m) \rangle = \langle t_1, \ldots, t_m \rangle$ always exists because the $y_1, \ldots, y_m$ are pairwise distinct. Since we assume $s_i \sim t_i$ for every i and because of $\{x_1, \ldots, x_n\} \subseteq \{y_1, \ldots, y_m\}$, Conditions (b) and (c) of Proposition 1 stipulate for every j that $\beta(x_j) \in P_j^{\mathcal{H}}$ and, hence, we also have $\gamma(x_j) \in P_j^{\mathcal{H}}$. Since $\mathcal{H}$ is a model of N^*, we have $\mathcal{H}, \gamma \models C$. This together with $\mathcal{H}, \gamma \models P_j(x_j)$, for every j, entails $\mathcal{H}, \gamma \models S(f(y_1, \ldots, y_m))$. Put differently, we have $\mathcal{H} \models S\big(f(t_1, \ldots, t_m)\big)$.

Consequently, for every S we observe that $\mathcal{H} \models S\big(f(s_1, \ldots, s_m)\big)$ entails $\mathcal{H} \models S\big(f(t_1, \ldots, t_m)\big)$. The converse direction can be shown by a symmetric argument. □

We now construct the finite structure $\mathcal{A}$. The universe of $\mathcal{A}$ shall be $\mathsf{A} := \big\{[s]_\sim \mid s \in \mathsf{H}\big\}$, where $[s]_\sim$ denotes the (unique) equivalence class with respect to $\sim$ which contains the term s. For every function symbol f (including constants) we set $f^{\mathcal{A}}\big([s_1]_\sim, \ldots, [s_m]_\sim\big) := \big[f(s_1, \ldots, s_m)\big]_\sim$ for all ground terms $s_1, \ldots, s_m$. Finally, we define each predicate P under $\mathcal{A}$ by $P^{\mathcal{A}} := \big\{[s]_\sim \mid \mathcal{H} \models P(s)\big\}$.

Lemma 4. *Let γ be any variable assignment over $\mathcal{A}$'s domain. Let β be some variable assignment over $\mathcal{H}$'s domain defined such that for every x we have $\gamma(x) = \big[\beta(x)\big]_\sim$. By definition of $\mathcal{H}$, such a β must exist. Then, for every term t in N and every predicate P we have $\mathcal{A}, \gamma \models P(t)$ if and only if $\mathcal{H}, \beta \models P(t)$.*

Proof (Sketch). We proceed by case distinction regarding the structure of the term t. If $t = x$ is a variable, then we have $\mathcal{A}, \gamma \models P(x)$ if and only if $\gamma(x) = \big[\beta(x)\big]_\sim \in P^{\mathcal{A}}$ if and only if $\beta(x) \in P^{\mathcal{H}}$ if and only if $\mathcal{H}, \beta \models P(x)$. If $t = c$ is a constant, then we have $\mathcal{A}, \gamma \models P(c)$ if and only if $c^{\mathcal{A}} = [c]_\sim \in P^{\mathcal{A}}$ if and only if $c \in P^{\mathcal{H}}$ if and only if $\mathcal{H}, \beta \models P(c)$.

Suppose $t = f(s_1, \ldots, s_m)$ for some function f of arity $m \geq 1$ and terms $s_1, \ldots, s_m$. Let $t_1, \ldots, t_m$ be ground terms such that $\mathcal{A}(\gamma)(s_i) = [t_i]_\sim$. Such terms exist by definition of $\mathcal{H}$. Then, $\mathcal{A}, \gamma \models P\big(f(s_1, \ldots, s_m)\big)$ if and only if $f^{\mathcal{A}}\big([t_1]_\sim, \ldots, [t_m]_\sim\big) = \big[f(t_1, \ldots, t_m)\big]_\sim \in P^{\mathcal{A}}$ if and only if $\mathcal{H} \models P\big(f(t_1, \ldots, t_m)\big)$. A straightforward induction on the structure of the terms s_i yields $t_i \sim \mathcal{H}(\beta)(s_i)$ for every i (see [26] for details), where $\mathcal{H}(\beta)(s_i)$ denotes the value of the term s_i under $\mathcal{H}$ and β. Then, by Lemma 3, we have $\mathcal{H} \models P\big(f(t_1, \ldots, t_m)\big)$ if and only if $\mathcal{H} \models P\big(f\big(\mathcal{H}(\beta)(s_1), \ldots, \mathcal{H}(\beta)(s_m)\big)\big)$ if and only if $\mathcal{H}, \beta \models P\big(f(s_1, \ldots, s_m)\big)$. □

For the special case of ground terms, there is a simpler form of Lemma 4:

Corollary 5. *For every ground term t and every predicate symbol P we have $\mathcal{A} \models P(t)$ if and only if $\mathcal{H} \models P(t)$.*

Using Lemma 4, it is easy to show that N is satisfied by the finite structure $\mathcal{A}$ (see [26] for details).

Theorem 6 (Finite Model Property for MSLH). *Every satisfiable finite MSLH clause set N has a finite model whose domain contains at most 2^p elements, where p is the number of predicates occurring in N.*

For a number of fragments enjoying the finite model property, such as the Bernays-Schoenfinkel fragment, the size of minimal models also depends on the number of constants. Ground unit clauses are sufficient to force the growth of models. Ground unit clauses such as the unit clause $T(f(a,b,c))$ are not admitted in MSLH. But they can be encoded. For the example, the unit clauses $S_1(a)$, $S_2(b)$, $S_3(c)$ together with the clause $\neg S_1(x) \vee \neg S_2(y) \vee \neg S_3(z) \vee T(f(x,y,z))$ entail the ground unit clause $T(f(a,b,c))$. Forcing the growth of models via ground unit clauses requires the introduction of additional predicates in MSLH.

4 Model Representation Formalisms

Many known explicit first-order model representation formalisms are built on sequences of literals, often enhanced with constraints, eventually representing Herbrand models, e.g., [3,19,5,1,6], so called constraints atomic representations (CARMs) [8]. A thorough discussion of all known CARM model representation formalisms is beyond the scope of this paper. We concentrate on three basic building blocks of known model representation formalisms: atoms with disequality constraints [8,11] (ADCs), implicit generalizations [13] (IGs) and atoms with membership constraints [8,10] (AMCs). They form the basis for a number of concrete model representation formalisms that actually appear in the above mentioned calculi. For this section we consider a fixed, finite signature $\Sigma = (\mathcal{F}, \mathcal{P})$, e.g., the function and predicate symbols occurring in some finite clause set N. The results in this section for all three model representations will be the same: if terms, literals are linear, the models represented by the respective approaches have the finite model property. We will prove this as follows: (i) we provide an effective linear time translation of atoms with disequality constraints to implicit generalizations; (ii) we provide a linear time translation of implicit generalizations to intersections of tree automata [10] or complements thereof; (iii) we represent an atom with membership constraints by a tree automaton. Then, because tree automata are closed under intersection and complement, potentially causing an exponential blow up in size [10], the atoms generated by ADCs and IGs can also be represented by the accepted language of a single tree automaton. The accepted language of a tree automaton can be represented by a finite MSLH clause set, e.g., see [15]. Thus, by Theorem 6, linear ADCs, linear IGs, and linear AMCs have all the finite model property, i.e., they cannot represent models for clause sets with inherently infinite models.

A *linear* ADC [8,11] has the form $(A\colon x_1 \not\approx t_1, \ldots, x_n \not\approx t_n)$ where the x_i are all different and occur in A, the x_i do not occur in any t_j, the variables of the t_j do not occur in A and A as well as all t_j are linear. The ground atoms generated by such an ADC are all ground atoms $A\sigma$ such that there is no δ with $x_i\sigma = t_i\delta$ for some i. A *linear* IG [13] is an expression $A/\{B_1, \ldots, B_n\}$ where A and the B_i are linear atoms. Every ground instance of A that is not an instance of any B_i is generated by the IG $A/\{B_1, \ldots, B_n\}$. The ground atoms generated by an ADC are exactly the ground atoms generated by the respective linear IG $A/\{A\{x_1 \mapsto t_1\}, \ldots, A\{x_n \mapsto t_n\}\}$.

A tree automaton [7,10] consists of a finite set $\mathcal{Q}$ of states, a finite set $\mathcal{O}$ of operators, a subset of accepting states $\mathcal{Q}_A \subseteq \mathcal{Q}$, and a finite set of rules $f(q_1 \ldots, q_n) \mapsto q$ where $q, q_i \in \mathcal{Q}$, $f \in \mathcal{O}$. The accepted language of a tree automaton is inductively defined by $f(t_1, \ldots, t_n) \in q^{\mathcal{A}}$ if there is a rule $f(q_1 \ldots, q_n) \mapsto q$ and $t_i \in q_i^{\mathcal{A}}$ for all i. The overall accepted language is then $\bigcup \{q^{\mathcal{A}} \mid q \in \mathcal{Q}_A\}$.

For example, the ground instances of the linear atom $R(x, g(a, y))$ over signature $\Sigma = (\{g, a, b\}, \{R\})$ is the accepted language of the tree automaton $\mathcal{O} = \{R, g, a, b\}$ with rules $a \mapsto q_1$, $b \mapsto q_1$, $g(q_1, q_1) \mapsto q_1$, hence state q_1 accepts all ground terms, $a \mapsto q_2$, $g(q_2, q_1) \mapsto q_3$, and $R(q_1, q_3) \mapsto q_4$ where q_4 is the only accepting state recognizing all ground instances of $R(x, g(a, y))$.

If ta is a function mapping linear atoms to a tree automata accepting the respective ground instances, then the ground atoms generated by an IG $A/\{B_1, \ldots, B_n\}$ are accepted by the tree automaton $\mathrm{ta}(A) \cap \neg\, \mathrm{ta}(B_1) \cap \ldots \cap \neg\, \mathrm{ta}(B_n)$. Recall that tree automata are closed under intersection ($\cap$) and complement ($\neg$), however the above tree automaton may be exponentially larger in size compared to the size of $\mathrm{ta}(A)$ and the $\mathrm{ta}(B_i)$.

A *linear* atom with membership constraint $A\colon x \in S$ is the pair of a linear atom A and a constraint $x \in S$ where x occurs in A and S is represented by a tree automaton. It generates all ground instances $A\sigma$ where $x\sigma$ is accepted by the tree automaton representing S. There is a function $\mathrm{ta}(A\colon x \in S)$ that computes in linear time a tree automaton accepting exactly the generated ground instances of $A\colon x \in S$. Basically, the state(s) representing the instances of x in A in $\mathrm{ta}(A)$ have to be replaced by the accepting states of the tree automaton representing S.

Finally, tree automata can be straight forwardly represented via MSLH clause sets. For example, the tree automaton representing the ground instances of $R(x, g(a, y))$ shown before, can be translated into the MSLH clause set $\to Q_1(a)$; $\to Q_1(b)$; $Q_1(x), Q_1(y) \to Q_1(g(x, y))$; $\to Q_2(a)$; $Q_2(x), Q_1(y) \to Q_3(g(x, y))$; and $Q_1(x), Q_3(y) \to Q_f(f_R(x, y))$. This, together with Theorem 6, eventually proves the following theorem.

Theorem 7. *Linear disequality constraints (ADCs), linear implicit generalizations (IGs) and linear atoms with membership constraints (AMCs) have the finite model property.*

This result can be easily generalized to any "Boolean combination" of linear ADCs, IGs, and AMCs, because tree automata are closed under Boolean operations. Our restriction on linearity does not imply that non-linear ADCs, IGs, and AMCs do *not* have the finite model property. This is an open problem. For example, the non-linear IG $R(x, x)/\{R(g(x), g(x))\}$ over signature $\Sigma = (\{g, h, a\}, \{R\})$ generates the infinite Herbrand model $\{R(a, a)\} \cup \{R(h(x), h(x))\sigma \mid \sigma \,\text{grounding}\}$. However, it also has a finite model over the domain $\mathsf{A} := \{\mathsf{a}, \mathsf{b}\}$ of two elements where a is interpreted by a, g maps constantly to b, h maps constantly to a, and R is the relation $\{(\mathsf{a}, \mathsf{a})\}$.

Non-linear MSH clause sets do not have the finite model property, because they are as expressive as full first-order logic.

5 Model Finding by Approximation Refinement

The AR calculus as it is used here works as follows [28]. Starting from a clause set N it is approximated into a MSLH clause set N' by the following steps: (i) all non monadic relation literals are turned into function terms below of a new monadic predicate T, (ii) all non-linear variable occurrences in positive literals are linearized, (iii) all nested function terms in positive literals are abstracted through the introduction of further monadic predicates, (iv) non Horn clauses are split into Horn clauses after removing variable dependencies between positive literals. Then if N' is satisfiable, so is N. If N' is unsatisfiable, the AR calculus tries to lift the proof to N. This may fail, in particular, because of the variable linearizations. In this case the respective clauses are instantiated and again approximated in order to get rid of the particular proof in N' and the AR calculus continues. If the proofs out of N' are generated in a fair way, the AR calculus is complete [28]. The Horn splitting can be omitted but then a decision procedure for MSL clause sets is needed [29].

The approximation refinement approach [28] cannot show satisfiability of the simple clause set with the two unit clauses

$$R(x,x), \quad \neg R(y,g(y)).$$

The approximated clause set consisting of the three clauses

$$T(f_R(x,y)), \quad \neg S(z) \vee \neg T(f_R(y,z)), \quad S(g(y))$$

immediately yields a refutation. The problem is that $R(x,x)$ cannot be refined in such a way that all instances of the conflict clause $\neg R(y,g(y))$ are excluded. The refinement loop instead ends up enumerating all $R(g^i(x),g^i(x))$ but $R(g^{i+1}(y),g^{i+2}(y))$ will always remain as a conflict clause.

The non-termination can be resolved, if the resolution inference in the abstracted clause set can be blocked. Our suggestion in case of reflexive relations is *reflexive relation splitting*, i.e., we split a reflexive relation into its reflexive part R_{ref} and irreflexive part R_{irr}. For the example, this yields

$$R_{\text{ref}}(x,x), \quad \neg R_{\text{irr}}(y,g(y))$$

and after approximation

$$T(f_{R_{\text{ref}}}(x,y)), \quad \neg S(z) \vee \neg T(f_{R_{\text{irr}}}(y,z)), \quad S(g(y)).$$

Now the approximation is saturated. The operation preserves satisfiability because the two R literals could not be resolved anyway.

In general, for each predicate R with a reflexivity axiom, all occurrences of atoms $R(s,t)$ are replaced with $R_{\text{ref}}(s,t)$ and/or $R_{\text{irr}}(s,t)$. If s and t are not unifiable, $R(s,t)$ is replaced with $R_{\text{irr}}(s,t)$. If there is an mgu σ of s and t, $R(s,t)$ is replaced with both $R_{\text{ref}}(s\sigma,t\sigma)$ and $R_{\text{irr}}(s,t)$. More precisely, any clause $C \vee [\neg]R(s,t)$ is replaced by two clauses: $C \vee [\neg]R_{\text{irr}}(s,t)$ and $C\sigma \vee [\neg]R_{\text{ref}}(s\sigma,t\sigma)$. The process is repeated until all atom occurrences with R have been replaced. In the final result, any clause that contains an atom of the form $R_{\text{irr}}(s,s)$ can be deleted.

More formally, the following transition system replaces a reflexive R by the two new predicates.

Irreflexive $N \uplus \{[\neg]R(s,t) \vee C\} \;\Rightarrow_{\text{RRS}}\; N \uplus \{[\neg]R_{\text{irr}}(s,t) \vee C\}$

provided s and t are not unifiable

Reflexive $N \uplus \{[\neg]R(s,t) \vee C\} \Rightarrow_{\mathrm{RRS}} N \uplus \{[\neg]R_{\mathrm{irr}}(s,t) \vee C, [\neg]R_{\mathrm{ref}}(s\sigma,t\sigma) \vee C\sigma\}$
provided s and t are unifiable by an mgu σ

Delete $N \uplus \{[\neg]R_{\mathrm{irr}}(s,s) \vee C\} \Rightarrow_{\mathrm{RRS}} N$

Lemma 8. *$\Rightarrow_{RRS}$ is terminating and confluent.*

Proof (Sketch). Termination is easy to prove. Each application of the rules Irreflexive or Reflexive reduces the multiset of the numbers of R-occurrences in all clauses, and no new occurrences of R are ever introduced when Delete is applied. Each application of the rule Delete reduces the number of occurrences of R_{irr}. Combining these two properties into a well-founded multi-set-based ordering completes the proof of termination. For local confluence, the non-obvious case is a clause $[\neg]R(s,t) \vee [\neg]R(s',t') \vee C$ where $R(s,t)$ and $R(s',t')$ share variables, and without loss of generality, s and t are unifiable by the mgu σ. Applying first the reflexive transformation to the first literal yields the two clauses $[\neg]R_{\mathrm{ref}}(s\sigma,t\sigma) \vee [\neg]R(s'\sigma,t'\sigma) \vee C\sigma$ and $[\neg]R_{\mathrm{irr}}(s,t) \vee [\neg]R(s',t') \vee C$. Now the interesting case is where s', t' are unifiable but $s'\sigma$ and $t'\sigma$ are not. Then we get with the mgu τ of s',t': $[\neg]R_{\mathrm{ref}}(s\sigma,t\sigma) \vee [\neg]R_{\mathrm{irr}}(s'\sigma,t'\sigma) \vee C\sigma$, $[\neg]R_{\mathrm{irr}}(s,t) \vee [\neg]R_{\mathrm{irr}}(s',t') \vee C$, and $[\neg]R_{\mathrm{irr}}(s\tau,t\tau) \vee [\neg]R_{\mathrm{ref}}(s'\tau,t'\tau) \vee C\tau$. This is also exactly the result we get when starting with a translation of $[\neg]R(s',t')$: if $s'\sigma,t'\sigma$ are not unifiable, then $s\tau,t\tau$ are not unifiable as well. For otherwise, $s\tau\tau',t\tau\tau'$ for unifier τ' is an instance of $s\sigma,t\sigma$, so $s'\sigma,t'\sigma$ must be unifiable as well, a contradiction to the above assumption. All other cases are similar to this case. By Newman's Lemma, termination plus local confluence implies confluence. □

Given any finite clause set N, we write $\mathrm{rrs}(N)$ to address the normal form of N after exhaustively applying $\Rightarrow_{\mathrm{RRS}}$. Notice that any clause $D \in \mathrm{rrs}(N)$ is an instance of a clause in N with respect to the renaming of R_{ref}, R_{irr} with R. Moreover, we use $\mathrm{rrs}(C)$ as shorthand for $\mathrm{rrs}(\{C\})$ for any clause C.

Lemma 9 (Reflexive Relation Splitting). *Let N be a finite clause set that does not contain the predicates R_{ref} and R_{irr}. N is satisfiable if and only if* $\mathrm{rrs}(N)$ *is satisfiable.*

Proof (Sketch). Consider the derivation $N \Rightarrow_{\mathrm{RRS}} N_1 \Rightarrow_{\mathrm{RRS}} \ldots \Rightarrow_{\mathrm{RRS}} \mathrm{rrs}(N)$ where we assume that the rule Delete is applied with priority whenever it is applicable. By confluence of $\Rightarrow_{\mathrm{RRS}}$, this is not a restriction.

We use an auxiliary result that is not hard to prove (see [26] for details):
Claim: Let M_1, M_2 be clause sets such that $M_1 \Rightarrow_{\mathrm{RRS}} M_2$. Moreover, let $\mathcal{I}$ be any Herbrand interpretation such that (1) for every ground term s we have $R_{\mathrm{ref}}(s,s) \in \mathcal{I}$ if and only if $R(s,s) \in \mathcal{I}$, (2) for all ground terms s,t and all $R_{\mathrm{ref}}(s,t) \in \mathcal{I}$ we have $s = t$, (3) for all ground terms s,t we have $R_{\mathrm{irr}}(s,t) \in \mathcal{I}$ if and only if $R(s,t) \in \mathcal{I}$. Then, $\mathcal{I} \models M_1$ if and only if $\mathcal{I} \models M_2$. ◊

Let $\mathcal{I}$ be a Herbrand model of N. Since N does not contain the predicates R_{ref} and R_{irr}, we can bring $\mathcal{I}$ into the shape that meets the conditions of the

above claim and still ensure that $\mathcal{I} \models N$. It then follows that $\mathcal{I} \models N$, $\mathcal{I} \models N_1, \ldots, \mathcal{I} \models \mathrm{rrs}(N)$. Symmetrically, let $\mathcal{I}$ be a Herbrand model of $\mathrm{rrs}(N)$. Since $\mathrm{rrs}(N)$ does not contain the predicate R, we can reshape $\mathcal{I}$ so that the above claim is applicable and $\mathcal{I}$ is still a model of $\mathrm{rrs}(N)$. Then, we get $\mathcal{I} \models \mathrm{rrs}(N), \ldots, \mathcal{I} \models N$. □

Notice that the above lemma holds independently of the fact whether there is a reflexivity clause in N or not. Such a clause would, of course, also be transformed by $\Rightarrow_{\mathrm{RRS}}$.

Let us take a look at an example that is a little bit more involved. Consider an equivalence relation R with the respective axiom clauses.

$$\begin{aligned} &\rightarrow R(x,x) \\ R(x,y) &\rightarrow R(y,x) \\ R(x,y), R(y,z) &\rightarrow R(x,z) \end{aligned}$$

Applying $\Rightarrow_{\mathrm{RRS}}$ exhaustively results in the clause set

$$\begin{aligned} &\rightarrow R_{\mathrm{ref}}(x,x) \\ R_{\mathrm{irr}}(x,y) &\rightarrow R_{\mathrm{irr}}(y,x) \\ R_{\mathrm{ref}}(x,x) &\rightarrow R_{\mathrm{ref}}(x,x) \\ R_{\mathrm{irr}}(x,y), R_{\mathrm{irr}}(y,z) &\rightarrow R_{\mathrm{irr}}(x,z) \\ R_{\mathrm{irr}}(x,y), R_{\mathrm{irr}}(y,x) &\rightarrow R_{\mathrm{ref}}(x,x) \\ R_{\mathrm{irr}}(x,y), R_{\mathrm{ref}}(y,y) &\rightarrow R_{\mathrm{irr}}(x,y) \\ R_{\mathrm{ref}}(x,x), R_{\mathrm{irr}}(x,z) &\rightarrow R_{\mathrm{irr}}(x,z) \\ R_{\mathrm{ref}}(x,x), R_{\mathrm{ref}}(x,x) &\rightarrow R_{\mathrm{ref}}(x,x). \end{aligned}$$

After removing redundant clauses, we are conveniently left with just

$$\begin{aligned} &\rightarrow R_{\mathrm{ref}}(x,x) \\ R_{\mathrm{irr}}(x,y) &\rightarrow R_{\mathrm{irr}}(y,x) \\ R_{\mathrm{irr}}(x,y), R_{\mathrm{irr}}(y,z) &\rightarrow R_{\mathrm{irr}}(x,z) \end{aligned}$$

which are no longer trivialized by the linear approximation $T(f_R(x,y))$ of the reflexivity axiom generated by the AR calculus. See also the example in the introduction, Section 1, for another application of reflexive relation splitting.

The rule Reflexive replaces a clause by two clauses and can, therefore, cause an exponential blow up in the number of generated clauses. However, this is only the case for a clause with several occurrences $R(s_i, t_i)$ such that the respective term pairs are all simultaneously unifiable. This situation can be detected and then reflexive relation splitting may not be efficiently applicable. However, the above example on the equivalence relation R shows that in the case of variable chains as they occur in the transitivity axiom, all of the eventually generated clauses become redundant, except one: $R_{\mathrm{irr}}(x,y), R_{\mathrm{irr}}(y,z) \rightarrow R_{\mathrm{irr}}(x,z)$. We have integrated reflexive relation splitting into SPASS-AR [28,24] and have run it on

the overall TPTP [22]. There is no example in TPTP v.7.2.0 showing the exponential blow up and the set of problems solved by SPASS-AR with reflexive relation splitting is strictly larger than without.

Nevertheless, reflexive relation splitting is, of course, not sufficient to transform all problems with inherently infinite models based on a (ir)reflexive relation into clause sets that can eventually be decided by AR via MSLH clause sets. Consider a strict partial ordering without endpoints:

$$\begin{aligned} R(x,x) &\rightarrow \\ &\rightarrow R(x,g(x)) \\ R(x,y), R(y,z) &\rightarrow R(x,z). \end{aligned}$$

Reflexive relation splitting yields

$$\begin{aligned} R_{\mathrm{ref}}(x,x) &\rightarrow \\ &\rightarrow R_{\mathrm{irr}}(x,g(x)) \\ R_{\mathrm{irr}}(x,y), R_{\mathrm{irr}}(y,z) &\rightarrow R_{\mathrm{irr}}(x,z) \\ R_{\mathrm{irr}}(x,y), R_{\mathrm{irr}}(y,x) &\rightarrow R_{\mathrm{ref}}(x,x) \end{aligned}$$

but after approximation, the abstraction refinement does not terminate on the example. The reason is the approximation of the clause $R_{\mathrm{irr}}(x,g(x))$ into the two clauses $S(g(x))$ and $S(y) \rightarrow T(f_{R_{\mathrm{irr}}}(x,y))$ where the property is lost that in any ground instance of $R_{\mathrm{irr}}(x,g(x))$ the first argument has one occurrence of g less than the second. This was resolved in [18] by the use of tuple tree automata.

6 Discussion

We have shown that the MSLH clause fragment has the finite model property and can therefore not represent models of clause sets with inherently infinite models. This applies to the model representation building blocks atoms with disequality constraints, implicit generalizations, and atoms with membership constraints as well, if atoms and terms are linear. For non-linear terms, our finite model property proof breaks, and, in fact, the example from the introduction shows already that non-linear atoms can represent models for clause sets with inherently infinite models.

Unsatisfiability of monadic shallow Horn clause sets is undecidable. One occurrence of a clause $\Gamma \rightarrow S(f(x,x))$ suffices to this end. This can be seen by a respective monadic reformulation of the PCP encoding from the introduction. On the other hand, models represented by ground instances of finite sets of (linear or non-linear) atoms are also restricted in expressivity, because they cannot express any recursive structure. For example, MSLH clause sets and extensions thereof have been successfully used for the analysis of security protocols [30,4] where (counter-) models cannot be expressed by ground instances of finite sets of atoms. In summary, and not surprisingly, there is currently no unique superior model representation formalism.

If models are eventually constructed through the reversal of an approximation, the used representation may have the finite model property and can still show satisfiability of clause sets with inherently infinite models. We obtained this result via reflexive relation splitting. This insight is already a consequence of [18]. There, an approximation into a theory of tuple tree automata is described and it is even complete with respect to models generated out of these automata. We can currently not provide such a completeness result although this would be highly desirable. On the other hand, our techniques are embedded into a refutationally complete procedure, whereas the approach in [18] can only show satisfiability.

Acknowledgments: This work was funded by DFG grant 389792660 as part of TRR 248. We thank our reviewers for their valuable comments.

References

1. Gábor Alagi and Christoph Weidenbach. NRCL - A model building approach to the Bernays-Schönfinkel fragment. In *Frontiers of Combining Systems (FroCoS'15)*, volume 9322 of *LNCS*, pages 69–84. Springer, 2015.
2. Leo Bachmair and Harald Ganzinger. Rewrite-based equational theorem proving with selection and simplification. *Journal of Logic and Computation*, 4(3):217–247, 1994.
3. Peter Baumgartner, Alexander Fuchs, and Cesare Tinelli. Lemma learning in the model evolution calculus. In *LPAR*, volume 4246 of *LNCS*, pages 572–586. Springer, 2006.
4. Bruno Blanchet. Automatic proof of strong secrecy for security protocols. In *2004 IEEE Symposium on Security and Privacy*, pages 86–100, 2004.
5. Maria Paola Bonacina, Ulrich Furbach, and Viorica Sofronie-Stokkermans. On first-order model-based reasoning. In *Logic, Rewriting, and Concurrency*, volume 9200 of *LNCS*, pages 181–204. Springer, 2015.
6. Maria Paola Bonacina and David A. Plaisted. Semantically-guided goal-sensitive reasoning: Model representation. *Journal of Automated Reasoning*, 56(2):113–141, 2016. Inference System and Completeness in 59(2), JAR.
7. Walther S. Brainerd. The minimalization of tree automata. *Information and Control*, 13:484–491, 1968.
8. Ricardo Caferra, Alexander Leitsch, and Nicholas Peltier. *Automated Model Building*, volume 31 of *Applied Logic Series*. Kluwer, 2004.
9. Koen Claessen and Niklas Soerensson. New techniques that improve mace-style finite model finding. In *Proceedings of the CADE-19 Workshop: Model Computation – Principles, Algorithms, Applications*, 2003.
10. H. Comon, M. Dauchet, R. Gilleron, C. Löding, F. Jacquemard, D. Lugiez, S. Tison, and M. Tommasi. Tree automata techniques and applications. Available on: `http://tata.gforge.inria.fr/`, 2007. release October, 12th 2007.
11. Hubert Comon. Disunification: A survey. In *Computational Logic - Essays in Honor of Alan Robinson*, pages 322–359. The MIT Press, 1991.
12. Heinz-Dieter Ebbinghaus, Jörg Flum, and Wolfgang Thomas. *Mathematical Logic*. Springer, second edition, 1994.

13. Christian G. Fermüller and Reinhard Pichler. Model representation over finite and infinite signatures. *J. Log. Comput.*, 17(3):453–477, 2007.
14. Julio Cesar Lopez Hernandez and Konstantin Korovin. Towards an abstraction-refinement framework for reasoning with large theories. In *IWIL@LPAR 2017*. EasyChair, 2017.
15. Florent Jacquemard, Christoph Meyer, and Christoph Weidenbach. Unification in extensions of shallow equational theories. In *Rewriting Techniques and Applications (RTA'98)*, volume 1379 of *LNCS*, pages 76–90. Springer, 1998.
16. Konstantin Korovin. Inst-gen - A modular approach to instantiation-based automated reasoning. In *Programming Logics*, volume 7797 of *LNCS*, pages 239–270. Springer, 2013.
17. William McCune. Mace4 reference manual and guide. *CoRR*, cs.SC/0310055, 2003.
18. Nicolas Peltier. Constructing infinite models represented by tree automata. *Annals of Mathematics and Artificial Intelligence*, 56:65–85, 2009.
19. Ruzica Piskac, Leonardo Mendonça de Moura, and Nikolaj Bjørner. Deciding effectively propositional logic using DPLL and substitution sets. *Journal of Automated Reasoning*, 44(4):401–424, 2010.
20. Emil L. Post. A variant of a recursively unsolvable problem. *Bulletin of the American Mathematical Society*, 52:264–268, 1946.
21. John K. Slaney. FINDER: finite domain enumerator - system description. In *Automated Deduction (CADE-12)*, volume 814 of *LNCS*, pages 798–801. Springer, 1994.
22. G. Sutcliffe. The TPTP Problem Library and Associated Infrastructure: The FOF and CNF Parts, v3.5.0. *Journal of Automated Reasoning*, 43(4):337–362, 2009.
23. Philippe Suter, Mirco Dotta, and Viktor Kuncak. Decision procedures for algebraic data types with abstractions. In *Proceedings of the 37th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2010, Madrid, Spain, January 17-23, 2010*, pages 199–210. ACM, 2010.
24. Andreas Teucke. *An Approximation and Refinement Approach to First-Order Automated Reasoning*. Doctoral thesis, Saarland University, 2018.
25. Andreas Teucke, Marco Voigt, and Christoph Weidenbach. On the expressivity and applicability of model representation formalisms. In Andreas Herzig and Andrei Popescu, editors, *Frontiers of Combining Systems - 12th International Symposium, FroCoS 2019, London, UK, September 4-6, 2019, Proceedings*, volume 11715 of *Lecture Notes in Computer Science*, pages 22–39. Springer, 2019.
26. Andreas Teucke, Marco Voigt, and Christoph Weidenbach. On the expressivity and applicability of model representation formalisms. *arXiv preprint*, arXiv:1905.03651 [cs.LO], 2019.
27. Andreas Teucke and Christoph Weidenbach. First-order logic theorem proving and model building via approximation and instantiation. In *Frontiers of Combining Systems (FroCoS'15)*, volume 9322 of *LNCS*. Springer, 2015.
28. Andreas Teucke and Christoph Weidenbach. Ordered resolution with straight dismatching constraints. In *Automated Reasoning (IJCAR'16)*, pages 95–109. Springer, 2016.
29. Andreas Teucke and Christoph Weidenbach. Decidability of the monadic shallow linear first-order fragment with straight dismatching constraints. In *Automated Deduction (CADE-26)*, volume 10395 of *LNCS*, pages 202–219. Springer, 2017.
30. Christoph Weidenbach. Towards an automatic analysis of security protocols in first-order logic. In *Automated Deduction (CADE-16)*, volume 1632 of *LNAI*, pages 314–328. Springer, 1999.

31. Christoph Weidenbach. Automated reasoning building blocks. In Roland Meyer, André Platzer, and Heike Wehrheim, editors, *Correct System Design*, volume 9360 of *LNCS*, pages 172–188. Springer, 2015.