Deciding the Bernays-Schoenfinkel Fragment over Bounded Difference Constraints by Simple Clause Learning over Theories

Martin Bromberger, Alberto Fiori, Christoph Weidenbach

Deciding the Bernays-Schoenfinkel Fragment over Bounded Difference Constraints by Simple Clause Learning over Theories

Martin Bromberger
Max Planck Institute for Informatics
Saarland Informatics Campus Saarbrücken

Alberto Fiori
Max Planck Institute for Informatics
Saarland Informatics Campus Saarbrücken and
Graduate School of Computer Science Saarbrücken, Germany

Christoph Weidenbach
Max Planck Institute for Informatics
Saarland Informatics Campus, Saarbrücken Germany

December 7, 2023

Abstract

Simple clause learning over theories SCL(T) is a decision procedure for the Bernays-Schoenfinkel fragment over bounded difference constraints BS(BD). The BS(BD) fragment consists of clauses built from first-order literals without function symbols together with simple bounds or difference constraints, where for the latter it is required that the variables of the difference constraint are bounded from below and above. The SCL(T) calculus builds model assumptions over a fixed finite set of fresh constants. The model assumptions consist of ground foreground first-order and ground background theory literals. The model assumptions guide inferences on the original clauses with variables. We prove that all clauses generated this way are non-redundant. As a consequence, expensive testing for tautologies and forward subsumption is completely obsolete and termination with respect to a fixed finite set of constants is a consequence. We prove SCL(T) to be sound and refutationally complete for the combination of the the Bernays Schoenfinkel fragment with any compact theory. Refutational completeness is obtained by enlarging the set of considered constants. For the case of BS(BD) we prove an abstract finite model property such that the size of a sufficiently large set of constants can be fixed a priori.

1 Introduction

Our work is motivated by the modeling, execution and verification of a "supervisor" [10], a component in a technical system that controls system functionality. Examples for a supervisor are the electronic control unit of a combustion engine or a control unit for the lane change assistant in a car. In this context we have been looking for a decsision procedure of a logical fragment that has sufficient expressivity to model supervisor functionality and properties; at the same time the fragment should run efficiently, i.e., generating consequences out of ground facts (inputs), and, finally, it should be push button verifiable (decidable). The Bernays Schoenfinkel fragment of first-order logic over linear arithmetic BS(LRA) is a candidate for such a fragment. The first-order part can be used to specify the rules and properties of a supervisor and the linear arithmetic part to deal with technicalities of the application. For example, computing the (real world) injection time of a charged combustion engine contains rules like the one below:

$P(x) \wedge 150 \leq x \leq 200 \wedge R(y) \wedge 6500 \leq y \leq 7000 \wedge T(x, y, z) \rightarrow I(z + 10)$

where x is the air pressure in the inlet manifold, y the speed of the engine, T a table lookup for the injection time z and finally 10 is added for engine heat protection. In the real world, the added heat protection part 10 is the result of an additional computation taking temperature of the engine, exhaust gas, inlet air and lambda value into account. In the supervisor context, a decision procedure should also deliver explanations, i.e., explicit (counter) models and proofs. In addition to efficiency, this is another reason why we designed SCL(T) to compute explicit models and proofs.

The combination of linear rational arithmetic (LRA) with the Bernays Schoenfinkel fragment of first-order logic (BS(LRA)) is already undecidable for a single monadic predicate [9, 13]. This can be shown by encoding the halting problem of a 2-counter machine [17]. For a number of universally quantified fragments there exist complete methods and some fragments are even decidable [12, 16, 24]. If the first-order part of BS(LRA) consists only of variables and predicates, then there exist refutationally complete calculi [1].

In this paper we introduce a new calculus SCL(T) (Simple Clause Learning over Theories) for the combination of a background theory with a foreground first-order logic without equality. As usual in a hierarchic setting, we assume the background theory to be term-generated and compact. In this paper we only consider *pure* clause sets where the only symbols occurring in the clause set from the foreground logic are predicates and variables. Reasoning in the SCL(T) calculus is driven by a partial, finite model assumption similar to conflict driven clause learning (CDCL) [23, 14, 18] and our previous work [11]. In contrast to SMT, the model assumption is not build on an a priori abstraction of ground literals to propositional logic, but from ground background and ground foreground theory literals generated by SCL(T) through instantiation and respecting their semantics. So any SCL(T) trail is always satisfiable in the combined theory. Inferences are performed on the original clauses with variables, similar to hierarchic superposition, where the ordering restrictions of hierarchic superposition are replaced by guidance through the partial ground model assumption. The main advantage of this approach is that learned clauses are never redundant, Lemma 22, and that the partial model assumption can be explored to derive an overall explicit model. SCL(T) is sound, Lemma 9 and

Lemma 12, and refutationally complete, Theorem 25. As a running example, we present the combination of linear rational arithmetic (LRA) with the Bernays Schoenfinkel fragment of first-order logic (BS(LRA)).

In order to demonstrate the potential of SCL(T) we prove it can decide the class BS(BD), the Bernays-Schoenfinkel fragment over simple bounds and bounded difference constraints, Section 3. This class was known to be decidable before [24], but not on the basis of a sound and complete calculus such as SCL(T).

Related Work: In contrast to variants of hierarchic superposition [1, 16, 4] SCL(T) selects clauses via a partial model assumption and not via an ordering. This has the advantage that SCL(T) does not generate redundant clauses. Where hierarchic superposition builds implicit models via saturated clause sets, SCL(T) builds explicit, finite model candidates that need to be extended to overall models of a clause set, see Example 13 and Section 4. One way to deal with universally quantified variables in an SMT setting is via instantiation [12, 21]. This has shown to be practically useful in many applications. It typically comes without completeness guarantees and it does not learn any new clauses with variables. While mcSAT [8] extends the SMT framework with the possibility to create new literals, its learning capabilities are also limited to the ground case. An alternative is to combine SMT techniques with superposition [7] where the ground literals from an SMT model assumption are resolved by superposition with first-order clauses. SCL(T) does not resolve with respect to its ground model assumption but on the original clauses with variables. Program verification through Horn clause reasoning [5] is another research direction related to SMT and SCL(T). Here constrained Horn clauses are considered and various reasoning methods have been developped. Our logic is not restricted to Horn clauses and our reasoning methods are different. In the same way SMT solving can be used to keep track of consistent SCL(T) trails, Horn clause reasoning could be used to explore the propagation space of an SCL(T) run. Background theories can also be built into first-order superposition in a kind of lazy way. This direction has been followed by SPASS+T [19] and Vampire [15]. The idea is to axiomatize part of the background theory in first-order logic and to direct ground literals of the background theory to SMT solver. Also this approach has shown to be practically useful but comes without any completeness guarantees and generated clauses may be redundant. Model evolution [2] has also been extended with linear integer arithmetic [3] where universally quantified integer variables are finitely bound from the beginning. A combination of first-order logic with linear integer arithmetic has also been built into a sequent calculus [22] that operates in the style of a free-variable tableau calculus with incremental closure. No new clauses are learned.

Organization of the Paper: After a section fixing notation, notions and some preliminary work, Section 2, the following Section 3 introduces the SCL(T) calculus and proves its properties. Missing proofs can be found in an arXiv publication [6]. Section 4 presents decidability of BS(BD) by SCL(T). The final Section 5 discusses extensions to model building, further improvements and summarizes the obtained results.

2 Preliminaries

Many-Sorted First-Order Logic without Equality: A *many-sorted signature* $\Sigma = (\mathcal{S}, \Omega, \Pi)$ is a triple consisting of a finite, non-empty set $\mathcal{S}$ of *sort symbols*, a non-empty set Ω of *operator symbols* (also called *function symbols*) over $\mathcal{S}$ and a finite set Π of *predicate symbols* over $\mathcal{S}$. For every sort from $\mathcal{S}$ there is at least one constant symbol in Ω of this sort. First-order terms, atoms, literals, clauses, formulas and substitutions are defined in the usual many-sorted way where an additional infinite set $\mathcal{X}$ of variables is assumed, such that for each sort from $\mathcal{S}$ there are infinitely many variables of this sort in $\mathcal{X}$. For each sort $S \in \mathcal{S}$, $T_S(\Sigma, \mathcal{X})$ denotes the set of all terms of sort S and $T_S(\Sigma)$ the set of all ground terms of sort S.

For notation, a, b, c are constants from Ω, w, x, y, z variables from $\mathcal{X}$, and if we want to emphasize the sort of a variable, we write x_S for a variable of sort S; t, s denote terms, P, Q, R predicates from Π, A, B atoms, L, K, H denote literals, C, D denote clauses, and N denotes a clause set. For substitutions we write σ, δ, ρ. Substitutions are well-sorted: if $x_s\sigma = t$ then $t \in T_S(\Sigma, \mathcal{X})$, they have a finite domain $\mathrm{dom}(\sigma) = \{x \mid x\sigma \neq x\}$ and their codomain is denoted by $\mathrm{codom}(\sigma) = \{x\sigma \mid x \in \mathrm{dom}(\sigma)\}$. The application of substitutions is homomorphically extended to non-variable terms, atoms, literals, clauses, and formulas. The complement of a literal is denoted by the function comp. For a literal L, $|L|$ denotes its respective atom. The function atoms computes the set of atoms from a clause or clause set. The function vars maps terms, literals, clauses to their respective set of contained variables. The function con maps terms, literals, clauses to their respective set of constants. A term, atom, clause, or a set of these objects is *ground* if it does not contain any variable, i.e., the function vars returns the empty set. A substitution σ is *ground* if $\mathrm{codom}(\sigma)$ is ground. A substitution σ is *grounding* for a term t, literal L, clause C if $t\sigma$, $L\sigma$, $C\sigma$ is ground, respectively. The function gnd computes the set of all ground instances of a literal, clause, or clause set. Given a set of constants B, the function gnd_B computes the set of all ground instances of a literal, clause, or clause set where the grounding is restricted to use constants from B. The function mgu denotes the *most general unifier* of two terms, atoms, literals. As usual, we assume that any mgu of two terms or literals does not introduce any fresh variables and is idempotent.

The semantics of many-sorted first-order logic is given by the notion of an algebra: let $\Sigma = (\mathcal{S}, \Omega, \Pi)$ be a many-sorted signature. A Σ-*algebra* $\mathcal{A}$, also called Σ-*interpretation*, is a mapping that assigns (i) a non-empty carrier set $S^{\mathcal{A}}$ to every sort $S \in \mathcal{S}$, so that $(S_1)^{\mathcal{A}} \cap (S_2)^{\mathcal{A}} = \emptyset$ for any distinct sorts $S_1, S_2 \in \mathcal{S}$, (ii) a total function $f^{\mathcal{A}} : (S_1)^{\mathcal{A}} \times \ldots \times (S_n)^{\mathcal{A}} \to (S)^{\mathcal{A}}$ to every operator $f \in \Omega$, $\mathrm{arity}(f) = n$ where $f : S_1 \times \ldots \times S_n \to S$, (iii) a relation $P^{\mathcal{A}} \subseteq ((S_1)^{\mathcal{A}} \times \ldots \times (S_m)^{\mathcal{A}})$ to every predicate symbol $P \in \Pi$ with $\mathrm{arity}(P) = m$. The semantic entailment relation $\models$ is defined in the usual way. We call a Σ-algebra $\mathcal{A}$ *term-generated* if $\mathcal{A}$ fulfills the following condition: whenever $\mathcal{A}$ entails all groundings $C\sigma$ of a clause C (i.e., $\mathcal{A} \models C\sigma$ for all grounding substitutions σ of a clause C), then $\mathcal{A}$ must also entail C itself (i.e., $\mathcal{A} \models C$).

Hierarchic Reasoning: The starting point of a hierarchic reasoning [1, 4] is a background theory $\mathcal{T}^{\mathcal{B}}$ over a many-sorted signature $\Sigma^{\mathcal{B}} = (\mathcal{S}^{\mathcal{B}}, \Omega^{\mathcal{B}}, \Pi^{\mathcal{B}})$ and a non-empty set of term-generated $\Sigma^{\mathcal{B}}$-algebras $\mathcal{C}^{\mathcal{B}}$: $\mathcal{T}^{\mathcal{B}} = (\Sigma^{\mathcal{B}}, \mathcal{C}^{\mathcal{B}})$. A

constant $c \in \Omega^{\mathcal{B}}$ is called a *domain constant* if $c^{\mathcal{A}} \neq d^{\mathcal{A}}$ for all $\mathcal{A} \in \mathcal{C}^{\mathcal{B}}$ and for all $d \in \Omega^{\mathcal{B}}$ with $d \neq c$. The background theory is then extended via a foreground signature $\Sigma^{\mathcal{F}} = (\mathcal{S}^{\mathcal{F}}, \Omega^{\mathcal{F}}, \Pi^{\mathcal{F}})$ where $\mathcal{S}^{\mathcal{B}} \subseteq \mathcal{S}^{\mathcal{F}}$, $\Omega^{\mathcal{B}} \cap \Omega^{\mathcal{F}} = \emptyset$, and $\Pi^{\mathcal{B}} \cap \Pi^{\mathcal{F}} = \emptyset$. Hierarchic reasoning is based on a background theory $\mathcal{T}^{\mathcal{B}}$ and a respective foreground signature $\Sigma^{\mathcal{F}}$: $\mathcal{H} = (\mathcal{T}^{\mathcal{B}}, \Sigma^{\mathcal{F}})$. It has its associated signature $\Sigma^{\mathcal{H}} = (\mathcal{S}^{\mathcal{F}}, \Omega^{\mathcal{B}} \cup \Omega^{\mathcal{F}}, \Pi^{\mathcal{B}} \cup \Pi^{\mathcal{F}})$ generating *hierarchic* $\Sigma^{\mathcal{H}}$-algebras. A $\Sigma^{\mathcal{H}}$-algebra $\mathcal{A}$ is called *hierarchic* with respect to its background theory $\mathcal{T}^{\mathcal{B}}$, if $\mathcal{A}^{\mathcal{H}}|_{\Sigma^{\mathcal{B}}} \in \mathcal{C}^{\mathcal{B}}$. As usual, $\mathcal{A}^{\mathcal{H}}|_{\Sigma^{\mathcal{B}}}$ is obtained from a $\mathcal{A}^{\mathcal{H}}$-algebra by removing all carrier sets $S^{\mathcal{A}}$ for all $S \in (\mathcal{S}^{\mathcal{F}} \setminus \mathcal{S}^{\mathcal{B}})$, all functions from $\Omega^{\mathcal{F}}$ and all predicates from $\Pi^{\mathcal{F}}$. We write $\models_{\mathcal{H}}$ for the entailment relation with respect to hierarchic algebras and formulas from $\Sigma^{\mathcal{H}}$ and $\models_{\mathcal{B}}$ for the entailment relation with respect to the $\mathcal{C}^{\mathcal{B}}$ algebras and formulas from $\Sigma^{\mathcal{B}}$.

Terms, atoms, literals build over $\Sigma^{\mathcal{B}}$ are called *pure background terms*, *pure background atoms*, and *pure background literals*, respectively. All terms, atoms, with a top-symbol from $\Omega^{\mathcal{B}}$ or $\Pi^{\mathcal{B}}$, respectively, are called *background terms*, *background atoms*, respectively. A background atom or its negation is a *background literal*. All terms, atoms, with a top-symbol from $\Omega^{\mathcal{F}}$ or $\Pi^{\mathcal{F}}$, respectively, are called *foreground terms*, *foreground atoms*, respectively. A foreground atom or its negation is a *foreground literal*. Given a set (sequence) of $\mathcal{H}$ literals, the function bgd returns the set (sequence) of background literals and the function fgd the respective set (sequence) of foreground literals. A substitution σ is called *simple* if $x_S\sigma \in T_S(\Sigma^{\mathcal{B}}, \mathcal{X})$ for all $x_S \in \mathrm{dom}(\sigma)$ and $S \in \mathcal{S}^{\mathcal{B}}$.

As usual, clauses are disjunctions of literals with implicitly universally quantified variables. We often write a $\Sigma^{\mathcal{H}}$ clause as a *constrained clause*, denoted $\Lambda \parallel C$ where Λ is a conjunction of background literals and C is a disjunction of foreground literals semantically denoting the clause $\neg\Lambda \vee C$. A *constrained closure* is denoted as $\Lambda \parallel C \cdot \sigma$ where σ is grounding for Λ and C. A constrained closure $\Lambda \parallel C \cdot \sigma$ denotes the ground constrained clause $\Lambda\sigma \parallel C\sigma$.

In addition, we assume a well-founded, total, strict ordering $\prec$ on ground literals, called an $\mathcal{H}$-order, such that background literals are smaller than foreground literals. This ordering is then lifted to constrained clauses and sets thereof by its respective multiset extension. We overload $\prec$ for literals, constrained clauses, and sets of constrained clause if the meaning is clear from the context. We define $\preceq$ as the reflexive closure of $\prec$ and $N^{\preceq \Lambda \parallel C} := \{D \mid D \in N \text{ and } D \preceq \Lambda \parallel C\}$. An instance of an LPO with according precedence can serve as $\prec$.

Definition 1 (Clause Redundancy)**.** A ground constrained clause $\Lambda \parallel C$ is *redundant* with respect to a set N of ground constrained clauses and an order $\prec$ if $N^{\preceq \Lambda \parallel C} \models_{\mathcal{H}} \Lambda \parallel C$. A clause $\Lambda \parallel C$ is *redundant* with respect to a clause set N, an $\mathcal{H}$-order $\prec$, and a set of constants B if for all $\Lambda' \parallel C' \in \mathrm{gnd}_B(\Lambda \parallel C)$ the clause $\Lambda' \parallel C'$ is redundant with respect to $\cup_{D \in N} \mathrm{gnd}_B(D)$.

Example 2 (BS(LRA))**.** The running example in this paper is the Bernays-Schoenfinkel clause fragment over linear arithmetic: BS(LRA). The background theory is linear rational arithmetic over the many-sorted signature $\Sigma^{\mathrm{LRA}} = (\mathcal{S}^{\mathrm{LRA}}, \Omega^{\mathrm{LRA}}, \Pi^{\mathrm{LRA}})$ with $\mathcal{S}^{\mathrm{LRA}} = \{\mathrm{LRA}\}$, $\Omega^{\mathrm{LRA}} = \{0, 1, +, -\} \cup \mathbb{Q}$, $\Pi^{\mathrm{LRA}} = \{\leq, <, \neq, =, >, \geq\}$) where LRA is the linear arithmetic sort, the function symbols consist of $0, 1, +, -$ plus the rational numbers and predicate symbols $\leq, <, =, \neq, >, \geq$. The linear arithmetic theory $\mathcal{T}^{\mathrm{LRA}} = (\Sigma^{\mathrm{LRA}}, \{\mathcal{A}^{\mathrm{LRA}}\})$ consists of the linear arithmetic signature together with the standard model $\mathcal{A}^{\mathrm{LRA}}$ of linear

arithmetic. This theory is then extended by the free (foreground) first-order signature $\Sigma^{\mathrm{BS}} = (\{\mathrm{LRA}\}, \Omega^{\mathrm{BS}}, \Pi^{\mathrm{BS}})$ where Ω^{BS} is a set of constants of sort LRA different from Ω^{LRA} constants, and Π^{BS} is a set of first-order predicates over the sort LRA. We are interested in hierarchic algebras $\mathcal{A}^{\mathrm{BS(LRA)}}$ over the signature $\Sigma^{\mathrm{BS(LRA)}} = (\{\mathrm{LRA}\}, \Omega^{\mathrm{BS}} \cup \Omega^{\mathrm{LRA}}, \Pi^{\mathrm{BS}} \cup \Pi^{\mathrm{LRA}})$ that are $\Sigma^{\mathrm{BS(LRA)}}$ algebras such that $\mathcal{A}^{\mathrm{BS(LRA)}}|_{\Sigma^{\mathrm{LRA}}} = \mathcal{A}^{\mathrm{LRA}}$.

Note that our definition of the BS(LRA) fragment restricted to the linear arithmetic sort does not restrict expressiveness compared to a definition adding further free sorts to Σ^{BS}. Free sorts containing only constants can be simulated by the linear arithmetic sort in a many-sorted setting.

We call a clause set N *abstracted* if the arguments of any predicate from $\Pi^{\mathcal{F}}$ are only variables. Abstraction can always be obtained by adding background constraints, e.g., the BS(LRA) clause $x > 1 \parallel R(x, 5)$ can be abstracted to $x > 1, y = 5 \parallel R(x, y)$, preserving satisfiability. Recall that even in the foreground signature we only consider background sorts and that the only operators in the foreground signature are constants.

A set N of $\mathcal{H}$ clauses is called *pure* if it does not contain symbols from $\Omega^{\mathcal{F}}$ ranging into a sort of $\mathcal{S}^{\mathcal{B}}$. In this case N is *sufficiently complete* according to [1], hence hierarchic superposition is complete for N [1, 4]. As a consequence, a pure clause set N is unsatisfiable iff $\mathrm{gnd}_B(N)$ can be refuted by hierarchic superposition for a sufficiently large set B of constants. We will make use of this result in the completeness proof for our calculus, Theorem 24.

Satisfiability of pure clause sets is undecidable. We already mentioned in the introduction that this can be shown through a reduction to the halting problem for two-counter machines [17, 13]. Clause redundancy for pure clause sets cannot be decided as well, still SCL(T) learns only non-redundant clauses.

Lemma 3 (Non-Redundancy for Pure Clause Sets is Undecidable)**.** For a pure clause set N it is undecidable whether some clause C is non-redundant with respect to N.

3 SCL(T)

Assumptions: For this section we consider only pure, abstracted clause sets N. We assume that the background theory $\mathcal{T}^{\mathcal{B}}$ is term-generated, compact, contains an equality $=$, and that all constants of the background signature are domain constants. We further assume that the set $\Omega^{\mathcal{F}}$ contains infinitely many constants for each background sort.

Example 4 (Pure Clauses)**.** With respect to BS(LRA) the unit clause $x \geq 5, 3x + 4y = z \parallel Q(x, y, z)$ is abstracted and pure while the clause $x \geq 5, 3x + 4y = a, z = a \parallel Q(x, y, z)$ is abstracted but not pure because of the foreground constant a of the LRA sort, and the clause $x \geq 5, 3x + 4y = 7 \parallel Q(x, y, 7)$ is not abstracted.

Note that for pure, abstracted clause sets, any unifier between two foreground literals is simple and its codomain consists of variables only.

In order for the SCL(T) calculus to be effective, decidability in $\mathcal{T}^{\mathcal{B}}$ is needed as well. For the calculus we implicitly use the following equivalence: A $\Sigma^{\mathcal{B}}$

sentence $\exists x_1, \ldots, x_n \phi$ where ϕ is quantifier free is true, i.e., $\models_{\mathcal{B}} \exists x_1, \ldots, x_n \phi$ iff the ground formula $\phi\{x_1 \mapsto a_1, \ldots, x_n \mapsto a_n\}$ where the a_i are $\Omega^{\mathcal{F}}$ constants of the respective background sorts is $\mathcal{H}$ satisfiable. Together with decidability in $\mathcal{T}^{\mathcal{B}}$ this guarantees decidability of the satisfiability of ground constraints from constrained clauses.

If not stated otherwise, satisfiability means satisfiability with respect to $\mathcal{H}$. The function $\mathrm{adiff}(B)$ for some finite sequence of background sort constants denotes a constraint that implies different interpretations for the constants in B. In case the background theory enables a strict ordering $<$ as LRA does, then the ordering can be used for this purpose. For example, $\mathrm{adiff}([a, b, c])$ is then the constraint $a < b < c$. In case the background theory does not enable a strict ordering, then inequations can express disjointness of the constants. For example, $\mathrm{adiff}([a, b, c])$ is then the constraint $a \neq b \wedge a \neq c \wedge b \neq c$. An ordering constraint has the advantage over an inequality constraint that it also breaks symmetries. Assuming all constants to be different will eventually enable a satisfiability test for foreground literals based on purely syntactic complementarity.

The inference rules of SCL(T) are represented by an abstract rewrite system. They operate on a problem state, a six-tuple $\Gamma = (M; N; U; B; k; D)$ where M is a sequence of annotated ground literals, the *trail*; N and U are the sets of *initial* and *learned* constrained clauses; B is a finite sequence of constants of background sorts for instantiation; k counts the number of decisions in M; and D is a constrained closure that is either $\top$, $\Lambda \parallel \bot \cdot \sigma$, or $\Lambda \parallel C \cdot \sigma$. Foreground literals in M are either annotated with a number, a level; i.e. , they have the form L^k meaning that L is the k-th guessed decision literal, or they are annotated with a constrained closure that propagated the literal to become true, i.e. , they have the form $(L\sigma)^{(\Lambda \parallel C \vee L) \cdot \sigma}$. An annotated literal is called a decision literal if it is of the form L^k and a propagation literal or a propagated literal if it of in the form $L \cdot \sigma^{(\Lambda \parallel C \vee L) \cdot \sigma}$. A ground foreground literal L is of *level* i with respect to a problem state $(M; N; U; B; k; D)$ if L or $\mathrm{comp}(L)$ occurs in M and the first decision literal left from L ($\mathrm{comp}(L)$) in M, including L, is annotated with i. If there is no such decision literal then its level is zero. A ground constrained clause $(\Lambda \parallel C)\sigma$ is of *level* i with respect to a problem state $(M; N; U; B; k; D)$ if i is the maximal level of a foreground literal in $C\sigma$; the level of an empty clause $\Lambda \parallel \bot \cdot \sigma$ is 0. A ground literal L is *undefined* in M if neither L nor $\mathrm{comp}(L)$ occur in M. The initial state for a first-order, pure, abstracted $\mathcal{H}$ clause set N is $(\epsilon; N; \emptyset; B; 0; \top)$, where B is a finite sequence of foreground constants of background sorts. These constants cannot occur in N, because N is pure. The final state $(\epsilon; N; U; B; 0; \Lambda \parallel \bot)$ denotes unsatisfiability of N. Given a trail M and its foreground literals $\mathrm{fgd}(M) = \{L_1, \ldots, L_n\}$ an $\mathcal{H}$ ordering $\prec$ *induced* by M is any $\mathcal{H}$ ordering where $L_i \prec L_j$ if L_i occurs left from L_j in M, or, L_i is defined in M and L_j is not.

The transition rules for SCL(T) are

Propagate $(M; N; U; B; k; \top) \Rightarrow_{\mathrm{SCL(T)}} (M, L\sigma^{(\Lambda \parallel C_0 \vee L)\delta \cdot \sigma}, \Lambda'\sigma; N; U; B; k; \top)$
provided $\Lambda \parallel C \in (N \cup U)$, σ is grounding for $\Lambda \parallel C$, $\mathrm{adiff}(B) \wedge \mathrm{bgd}(M) \wedge \Lambda\sigma$ is satisfiable, $C = C_0 \vee C_1 \vee L$, $C_1\sigma = L\sigma \vee \ldots \vee L\sigma$, $C_0\sigma$ does not contain $L\sigma$, δ is the mgu of the literals in C_1 and L, $\Lambda'\sigma$ are the background literals from $\Lambda\sigma$ that are not yet on the trail, $\mathrm{fgd}(M) \models \neg(C_0\sigma)$, $\mathrm{codom}(\sigma) \subseteq B$, and $L\sigma$ is undefined in M

The rule Propagate applies exhaustive factoring to the propagated literal with respect to the grounding substitution σ and annotates the factored clause to the propagation. By writing $M, L\sigma^{(\Lambda \| C_0 \vee L)\delta \cdot \sigma}, \Lambda'\sigma$ we denote that all background literals from $\Lambda'\sigma$ are added to the trail.

Decide $(M; N; U; B; k; \top) \Rightarrow_{\mathrm{SCL(T)}} (M, L\sigma^{k+1}, \Lambda\sigma; N; U; B; k+1; \top)$
provided $L\sigma$ is undefined in M, $|K\sigma| \in \mathrm{atoms}(\mathrm{gnd}_B(N \cup U))$ for all $K\sigma \in \Lambda\sigma$, $|L\sigma| \in \mathrm{atoms}(\mathrm{gnd}_B(N \cup U))$, σ is grounding for Λ, all background literals in $\Lambda\sigma$ are undefined in M, $\mathrm{adiff}(B) \wedge \mathrm{bgd}(M) \wedge \Lambda\sigma$ is satisfiable, and $\mathrm{codom}(\sigma) \subseteq B$

The number of potential trails of a run is finite because the rules Propagate and Decide make sure that no duplicates of background literals occur on the trail and that only undefined literals over a fixed finite sequence B of constants are added to the trail. Requiring the constants from B to be different by the $\mathrm{adiff}(B)$ constraint enables a purely syntactic consistency check for foreground literals.

Conflict $(M; N; U; B; k; \top) \Rightarrow_{\mathrm{SCL(T)}} (M; N; U; B; k; \Lambda \parallel D \cdot \sigma)$
provided $\Lambda \parallel D \in (N \cup U)$, σ is grounding for $\Lambda \parallel D$, $\mathrm{adiff}(B) \wedge \mathrm{bgd}(M) \wedge \Lambda\sigma$ is satisfiable, $\mathrm{fgd}(M) \models \neg(D\sigma)$, and $\mathrm{codom}(\sigma) \subseteq B$

Resolve $(M, L\rho^{\Lambda \| C \vee L \cdot \rho}; N; U; B; k; (\Lambda' \parallel D \vee L') \cdot \sigma) \Rightarrow_{\mathrm{SCL(T)}}$
$(M, L\rho^{\Lambda \| C \vee L \cdot \rho}; N; U; B; k; (\Lambda \wedge \Lambda' \parallel D \vee C)\eta \cdot \sigma\rho)$
provided $L\rho = \mathrm{comp}(L'\sigma)$, and $\eta = \mathrm{mgu}(L, \mathrm{comp}(L'))$

Note that Resolve does not remove the literal $L\rho$ from the trail. This is needed if the clause $D\sigma$ contains further literals complementary of $L\rho$ that have not been factorized.

Factorize $(M; N; U; B; k; (\Lambda \parallel D \vee L \vee L') \cdot \sigma) \Rightarrow_{\mathrm{SCL(T)}}$
$(M; N; U; B; k; (\Lambda \parallel D \vee L)\eta \cdot \sigma)$
provided $L\sigma = L'\sigma$, and $\eta = \mathrm{mgu}(L, L')$

Note that Factorize is not limited with respect to the trail. It may apply to any two literals that become identical by application of the grounding substitution σ.

Skip $(M, L; N; U; B; k; \Lambda' \parallel D \cdot \sigma) \Rightarrow_{\mathrm{SCL(T)}} (M; N; U; B; l; \Lambda' \parallel D \cdot \sigma)$
provided L is a foreground literal and $\mathrm{comp}(L)$ does not occur in $D\sigma$, or L is a background literal; if L is a foreground decision literal then $l = k - 1$, otherwise $l = k$

Note that Skip can also skip decision literals. This is needed because we won't eventually require exhaustive propagation. While exhaustive propagation in CDCL is limited to the number of propositional variables, in the context of our logic, for example BS(LRA), it is exponential in the arity of foreground predicate symbols and can lead to an unfair exploration of the space of possible inferences, harming completeness, see Example 7.

Backtrack $(M, K^{i+1}, M'; N; U; B; k; (\Lambda \parallel D \vee L) \cdot \sigma) \Rightarrow_{\mathrm{SCL(T)}}$
$(M, L\sigma^{(\Lambda \parallel D \vee L) \cdot \sigma}, \Lambda'\sigma; N; U \cup \{\Lambda \parallel D \vee L\}; B; i; \top)$
provided $L\sigma$ is of level k, and $D\sigma$ is of level i, $\Lambda'\sigma$ are the background literals from $\Lambda\sigma$ that are not yet on the trail

The definition of Backtrack requires that $L\sigma$ is the only literal of level k in $(D \vee L)\sigma$. Additional occurrences of $L\sigma$ in D have to be factorized first before Backtrack can be applied.

Grow $(M; N; U; B; k; \top) \Rightarrow_{\mathrm{SCL(T)}} (\epsilon; N; U; B \cup B'; 0; \top)$
provided B' is a non-empty sequence of foreground constants of background sorts distinct from the constants in B

In case the adiff constraint is implemented by a strict ordering predicate on the basis of the sequence B, it can be useful to inject the new constants B' into $B \cup B'$ such that the ordering of the constants from B is not changed. This can help caching background theory results for testing trail satisfiability.

Definition 5. The rules Propagate, Decide, Grow, and Conflict are called *conflict search* rules and the rules Resolve, Skip, Factorize, and Backtrack are called *conflict resolution* rules.

Recall that the goal of our calculus is to replace the ordering restrictions of the hierarchic superposition calculus with a guiding model assumption. All our inferences are hierarchic superposition inferences where the ordering restrictions are neglected.

The next two examples show that the adiff constraint is needed to produce satisfiable trails and that exhaustive propagation cannot be afforded, respectively.

Example 6 (Inconsistent Trail)**.** Consider a clause set $N = \{R(x, y), x \leq y \parallel \neg R(x, y) \vee P(x), x \geq y \parallel \neg R(x, y) \vee \neg P(y)\}$; if we were to remove the adiff(B) constraint from the side conditions of rule Propagate we would be able to obtain inconsistent trails. Starting with just $B = \{a, b\}$ as constants it is possible to propagate three times and obtain the trail $M = [R(a, b), P(a), a \leq b, \neg P(b), a \geq b]$, M is clearly inconsistent as $M \models P(a)$, $M \models \neg P(b)$ yet $a = b$.

Example 7 (Exhaustive Propagation)**.** Consider a BS(LRA) clause set $N = \{x = 0 \parallel \mathrm{Nat}(x),\ y = x + 1 \parallel \neg \mathrm{Nat}(x) \vee \mathrm{Nat}(y)\} \cup N'$ where N' is unsatisfiable and nothing can be propagated from N'. Let us further assume that N' is satisfiable with respect to any instantiation of variables with natural numbers. If propagation is not restricted, then the first two clauses will consume all constants in B. For example, if $B = [a, b, c]$ then the trail $[\mathrm{Nat}(a), a = 0, \mathrm{Nat}(b), b = a + 1, \mathrm{Nat}(c), c = b + 1]$ will be derived. Now all constants are fixed to natural numbers. So there cannot be a refutation of N' anymore. An application of Grow will not solve the issue, because again the first two rules will fix all constants to natural numbers via exhaustive propagation.

Definition 8 (Well-formed States)**.** A state $(M; N; U; B; k; D)$ is *well-formed* if the following conditions hold:

1. all constants appearing in $(M; N; U; B; k; D)$ are from B or occur in N.

2. $M \wedge \mathrm{adiff}(B)$ is satisfiable
3. $N \models_{\mathcal{H}} U$,
4. Propagating clauses remain propagating and conflict clauses remain false:
   (a) if $D = \Lambda \parallel C \cdot \sigma$ then $C\sigma$ is false in $\mathrm{fgd}(M)$ and $\mathrm{bgd}(M) \wedge \mathrm{adiff}(B) \wedge \Lambda\sigma$ is satisfiable,
   (b) if $M = M_1, L\sigma^{(\Lambda \parallel C \vee L) \cdot \sigma}, M_2$ then $C\sigma$ is false in $\mathrm{fgd}(M_1)$, $L\sigma$ is undefined in M_1, and $\mathrm{bgd}(M_1) \wedge \mathrm{adiff}(B) \wedge \Lambda\sigma$ is satisfiable.
5. All clauses in $N \cup U$ are pure. In particular, they don't contain any constants from B.

Lemma 9 (Rules preserve Well-Formed States)**.** The rules of SCL(T) preserve well-formed states.

Definition 10 (Stuck State)**.** A state $(M; N; U; B; k; D)$ is called *stuck* if $D \neq \Lambda \parallel \bot \cdot \sigma$ and none of the rules Propagate, Decide, Conflict, Resolve, Factorize, Skip, or Backtrack is applicable.

Proposition 11 (Form of Stuck States)**.** If a run (without rule Grow) where Conflict was applied eagerly ends in a stuck state $(M; N; U; B; k; D)$, then $D = \top$ and all ground foreground literals that can be build from the foreground literals in N by instantiation with constants from B are defined in M.

Lemma 12 (Stuck States Produce Ground Models)**.** Every stuck state $(M; N; U; B; k; \top)$ produces a ground model, i.e., $M \wedge \mathrm{adiff}(B) \models \mathrm{gnd}_B(N \cup U)$.

The next example shows that in some cases the finite partial model of a stuck state can be turned into an overall model. For some fragments of BS(LRA) this can be done systematically, see Section 4.

Example 13 (SCL(T) Model Extraction)**.** In some cases it is possible to extract an overall model from the ground trail of a stuck state of an SCL(T) derivation. Consider $B = [a, b, c]$ and a satisfiable BS(LRA) constrained clause set $N = \{x \geq 1 \parallel P(x), x < 0 \parallel P(x), 0 \leq x \wedge x < 1 \parallel \neg P(x), 2x \geq 1 \parallel P(x) \vee Q(x)\}$. Starting from state $(\epsilon; N; \emptyset; B; 0; \top)$ and applying Propagate fairly a regular run can derive the following trail

$$\begin{aligned} M = \quad & P(a)^{x \geq 1 \parallel P(x) \cdot \{x \mapsto a\}}, a \geq 1, P(b)^{x < 0 \parallel P(x) \cdot \{x \mapsto b\}}, b < 0, \\ & \neg P(c)^{0 \leq x \wedge x < 1 \parallel \neg P(x) \cdot \{x \mapsto c\}}, 0 \leq c, c < 1, Q(c)^{2x \geq 1 \parallel P \vee Q(x) \cdot \{x \mapsto c\}}, 2c \geq 1 \end{aligned}$$

The state $(M; N; \emptyset; B; 0; \top)$ is stuck and $M \models_{\mathcal{H}} \mathrm{gnd}_B(N)$. Moreover from M we can generate an interpretation $\mathcal{A}^{\mathrm{BS(LRA)}}$ of N by generalizing the foreground constants used for instantiation and interpreting the predicates P and Q as formulas over $\Sigma^{\mathcal{B}}$, $P^{\mathcal{A}} = \{q \in \mathbb{Q} \mid q < 0 \vee q \geq 1\}$ and $Q^{\mathcal{A}} = \{q \in \mathbb{Q} \mid 2q \geq 1 \wedge q < 1\}$.

Lemma 14 (Soundness)**.** If a derivation reaches the state $(M; N; U; B; k; \Lambda \parallel \bot \cdot \sigma)$, then N is unsatisfiable.

Definition 15 (Reasonable Run)**.** A sequence of SCL(T) rule applications is called a *reasonable run* if an application of rule Decide does not enable an application of rule Conflict.

Proposition 16 (Avoiding Conflicts after Decide)**.** Let N be a set of constrained clauses and $(M; N; U; B; k; \top)$ be a state derived from $(\epsilon; N; \emptyset; B; 0; \top)$. If an application of rule Decide to $(M; N; U; B; k; \top)$ enables an application of rule Conflict, then Propagate would have been applicable to $(M; N; U; B; k; \top)$.

Definition 17 (Regular Run)**.** A sequence of SCL(T) rule applications is called a *regular run* if it is a reasonable run, the rule Conflict has precedence over all other rules, and Resolve resolves away at least the rightmost foreground literal from the trail.

Proposition 18 (Stuck States at Regular Runs)**.** Lemma 12 also holds for regular runs.

Example 19 (SCL(T) Refutation)**.** Given a set of foreground constants $B = [a, b, c]$ and a BS(LRA) constrained clause set $N = \{C_1 \colon x = 0 \parallel P(x), C_2 \colon y = x + 1 \parallel \neg P(x) \vee P(y), C_3 \colon z = 2 \parallel \neg P(z)\}$ the following is a regular derivation

$$
\begin{aligned}
& (\epsilon; N; \emptyset; B; 0; \top) \\
\Rightarrow^{\text{Propagate}}_{\text{SCL(T)}} \quad & (P(a)^{C_1 \cdot \{x \mapsto a\}}, a = 0; N; \emptyset; B; 0; \top) \\
\Rightarrow^{\text{Propagate}}_{\text{SCL(T)}} \quad & (\ldots, P(b)^{C_2 \cdot \{x \mapsto a, y \mapsto b\}}, b = a + 1; N; \emptyset; B; 0; \top) \\
\Rightarrow^{\text{Propagate}}_{\text{SCL(T)}} \quad & (\ldots, P(c)^{C_2 \cdot \{x \mapsto b, y \mapsto c\}}, c = b + 1; N; \emptyset; B; 0; \top) \\
\Rightarrow^{\text{Conflict}}_{\text{SCL(T)}} \quad & (\ldots, P(c)^{C_2 \cdot \{x \mapsto b, y \mapsto c\}}, c = b + 1; N; \emptyset; B; 0; z = 2 \parallel \neg P(z) \cdot \{z \mapsto c\}) \\
\Rightarrow^{\text{Resolve}}_{\text{SCL(T)}} \quad & (\ldots, P(c)^{C_2 \cdot \{x \mapsto b, y \mapsto c\}}, c = b + 1; N; \emptyset; B; 0; \\
& z = x + 1 \wedge z = 2 \parallel \neg P(x) \cdot \{z \mapsto c, x \mapsto b\}) \\
\Rightarrow^{\text{Skip}}_{\text{SCL(T)}} \quad & (\ldots, P(b)^{C_2 \cdot \{x \mapsto a, y \mapsto b\}}, b = a + 1; N; \emptyset; B; 0; \\
& z = x + 1 \wedge z = 2 \parallel \neg P(x) \cdot \{z \mapsto c, x \mapsto b\}) \\
\Rightarrow^{\text{Resolve}}_{\text{SCL(T)}} \quad & (\ldots, P(b)^{C_2 \cdot \{x \mapsto a, y \mapsto b\}}, b = a + 1; N; \emptyset; B; 0; \\
& z = x + 1 \wedge z = 2 \wedge x = y + 1 \parallel \neg P(y) \cdot \{z \mapsto c, x \mapsto b, y \mapsto a\}) \\
\Rightarrow^{\text{Skip}}_{\text{SCL(T)}} \quad & (P(a)^{C_1 \cdot \{x \mapsto a\}}, a = 0; N; \emptyset; B; 0; \\
& z = x + 1 \wedge z = 2 \wedge x = y + 1 \parallel \neg P(y) \cdot \{z \mapsto c, x \mapsto b, y \mapsto a\}) \\
\Rightarrow^{\text{Resolve}}_{\text{SCL(T)}} \quad & (P(a)^{C_1 \cdot \{x \mapsto a\}}, a = 0; N; \emptyset; B; 0; \\
& z = x + 1 \wedge z = 2 \wedge x = y + 1 \wedge y = 0 \parallel \bot \cdot \{z \mapsto c, x \mapsto b, y \mapsto a\})
\end{aligned}
$$

N is proven unsatisfiable as we reach a state in the form $(M; N; U; B; k; \Lambda \parallel \bot \cdot \sigma)$.

Example 20 (SCL(T) Clause learning)**.** Given an initial constant set $B = [a]$ and a BS(LRA) constrained clause set $N = \{C_1 \colon x \geq y \parallel \neg P(x, y) \vee Q(z), C_2 \colon z = u + v \parallel \neg P(u, v) \vee \neg Q(z)\}$ the following is an example of a regular run

$$
\begin{array}{ll}
 & (\epsilon; N; \emptyset; B; 0; \top) \\
\Rightarrow^{Decide}_{\mathrm{SCL(T)}} & (P(a,b)^1; N; \emptyset; B; 1; \top) \\
\Rightarrow^{Propagate}_{\mathrm{SCL(T)}} & (P(a,a)^1, Q(a)^{C_1 \cdot \{x \mapsto a, y \mapsto a, z \mapsto a\}}, a \geq a; N; \emptyset; B; 1; \top) \\
\Rightarrow^{Conflict}_{\mathrm{SCL(T)}} & (P(a,a)^1, Q(a)^{C_1 \cdot \{u \mapsto a, v \mapsto a, z \mapsto a\}}, a \geq a; N; \emptyset; B; 1; \\
 & C_2 \cdot \{x \mapsto a, y \mapsto a, z \mapsto a\}) \\
\Rightarrow^{Resolve}_{\mathrm{SCL(T)}} & (P(a,a)^1, Q(a)^{C_1 \cdot \{x \mapsto a, y \mapsto a, z \mapsto a\}}, a \geq a; N; \emptyset; B; 1; x \geq y \wedge z = u + v \parallel \\
 & \neg P(x,y) \vee \neg P(u,v) \cdot \{x \mapsto a, y \mapsto a, z \mapsto a, u \mapsto a, v \mapsto a\}) \\
\Rightarrow^{Skip*}_{\mathrm{SCL(T)}} & (P(a,a)^1; N; \emptyset; B; 1; x \geq y \wedge z = u + v \parallel \\
 & \neg P(x,y) \vee \neg P(u,v) \cdot \{x \mapsto a, y \mapsto a, z \mapsto a, u \mapsto a, v \mapsto a\}) \\
\Rightarrow^{Factorize}_{\mathrm{SCL(T)}} & (P(a,a)^1; N; \emptyset; B; 1; x \geq y \wedge z = x + y \parallel \\
 & \neg P(x,y) \cdot \{x \mapsto a, y \mapsto a, z \mapsto a\}) \\
\Rightarrow^{Backtrack}_{\mathrm{SCL(T)}} & (\neg P(a,a)^{(x \geq y \wedge z = x+y \parallel \neg P(x,y)) \cdot \{x \mapsto a, y \mapsto a\}}, a \geq a, a = a + a; N; \\
 & \{x \geq y \wedge z = x + y \parallel \neg P(x,y)\}; B; 1; \top)
\end{array}
$$

The learned clause $x \geq y \wedge z = x + y \parallel \neg P(x,y)$ contains two distinct variables even if we had to use a single constant for instantiations in conflict search.

Corollary 21 (Regular Conflict Resolution)**.** Let N be a set of constrained clauses. Then any conflict in an SCL(T) regular run admits a regular conflict resolution if the run starts from state $(\epsilon; N; \emptyset; B; 0; \top)$.

Lemma 22 (Non-Redundant Clause Learning)**.** Let N be a set of constrained clauses, and let $\Lambda_n \parallel D \vee L$ be a clause learned in an SCL(T) regular run such that $(\epsilon; N; \emptyset; B; 0; \top) \Rightarrow^*_{\mathrm{SCL(T)}} \Rightarrow^{\mathrm{Backtrack}}_{\mathrm{SCL(T)}} (M, L\sigma^{(\Lambda_n \parallel D \vee L) \cdot \sigma}, \Lambda'_n \sigma; N; U \cup \{\Lambda_n \parallel D \vee L\}; B; i; \top)$. Then $\Lambda_n \parallel D \vee L$ is not redundant with respect to any $\mathcal{H}$ ordering $\prec$ induced by the trail M.

Of course, in a regular run the ordering of foreground literals on the trail will change, i.e., the ordering underlying Lemma 22 will change as well. Thus the non-redundancy property of Lemma 22 reflects the situation at the time of creation of the learned clause. A non-redundancy property holding for an overall run must be invariant against changes on the ordering. However, the ordering underlying Lemma 22 also entails a fixed subset ordering that is invariant against changes on the overall ordering. This means that our dynamic ordering entails non-redundancy criteria based on subset relations including forward redundancy. From an implementation perspective, this means that learned clauses need not to be tested for forward redundancy. Current resolution, or superposition based provers spent a reasonable portion of their time in testing forward redundancy of newly generated clauses. In addition, also tests for backward reduction can be restricted knowing that learned clauses are not redundant.

Lemma 23 (Termination of SCL(T))**.** Let N be a set of constrained clauses and B be a finite set of background constants. Then any regular run with start state $(\epsilon; N; \emptyset; B; 0; \top)$ that uses Grow only finitely often terminates.

Theorem 24 (Hierarchic Herbrand Theorem)**.** Let N be a finite set of clauses. N is unsatisfiable iff there exists a finite set $N' = \{\Lambda_1 \parallel C_1, \ldots, \Lambda_n \parallel C_n\}$ of variable renamed copies of clauses from N and a finite set B of fresh constants and a substitution σ, grounding for N' where $\mathrm{codom}(\sigma) = B$ such that $\bigwedge_i \Lambda_i \sigma$ is $\mathcal{T}^{\mathcal{B}}$ satisfiable and $\bigwedge_i C_i \sigma$ is first-order unsatisfiable over $\Sigma^{\mathcal{F}}$.

Finally, we show that an unsatisfiable clause set can be refuted by SCL(T) with any regular run if we start with a sufficiently large sequence of constants B and apply Decide in a fair way. In addition, we need a Restart rule to recover from a stuck state. Of course, an unrestricted use of rule Restart immediately leads to non-termination.

Restart $(M; N; U; B; k; \top) \Rightarrow_{\text{SCL(T)}} (\epsilon; N; U; B; 0; \top)$

Theorem 25 (Refutational Completeness of SCL(T))**.** Let N be an unsatisfiable clause set. Then any regular SCL(T) run will derive the empty clause provided (i) Rule Grow and Decide are operated in a fair way, such that all possible trail prefixes of all considered sets B during the run are eventually explored, and (ii) Restart is only applied to stuck states.

Condition (i) of the above theorem is quite abstract. It can, e.g., be made effective by applying rule Grow only after all possible trail prefixes with respect to the current set B have been explored and to make sure that Decide does not produce the same stuck state twice.

4 SCL(T) Decides BS(BD)

As mentioned in Lemma 12, all stuck states produce ground models for $\text{gnd}_B(N)$. This does not mean that all stuck states produce a full hierarchic algebra $\mathcal{A}$ that satisfies N, i.e., a model over the full and potentially infinite carrier sets of our theory (e.g., $\mathbb{R}$) instead of a model over a finite set of sample elements (i.e., our constants B). In this section, we explain on the example of the Bernays-Schoenfinkel clause fragment with bounded difference constraints (BS(BD)) how to formalize an extraction criterion, i.e., a condition that guarantees that a satisfying algebra $\mathcal{A}$ can be extracted from a stuck state that fulfills the condition. We also explain how $\mathcal{A}$ can be constructed explicitly from such a stuck state and which conditions on N guarantee that SCL(T) finds a stuck state that fulfills the extraction criterion.

Definition 26 (BS(BD))**.** The Bernays-Schoenfinkel fragment with bounded difference constraints is a subset of BS(LRA) that only allows theory atoms of the form $x \triangleleft c$, $x \triangleleft y$, or $x - y \triangleleft c$ where c may be any integer number, $x, y \in \mathcal{X}$, and $\triangleleft \in \{\leq, <, \neq, =, >, \geq\}$. Moreover, we require for all considered clauses $\Lambda \| C$ that the theory part Λ may only contain an atom of the form $x - y \triangleleft c$ if Λ also bounds x and y, i.e., Λ also contains atoms $c_x \leq x$, $x \leq d_x$, $c_y \leq y$, and $y \leq d_y$, where c_x, d_x, c_y, d_y are integers.

For the rest of this section we fix a finite set of BS(BD) clauses N, where κ is the maximal absolute value of any integer occurring in N and η is the maximal number of distinct variables in any single clause in N. Moreover, we define the function $\text{fr}(b) = b - \lfloor b \rfloor$ that returns the *fractional/decimal part* of a real number.

The first step of defining an extraction criterion is the definition of an equivalence relation that ranges over all possible argument tuples/grounding substitutions for literals and clauses in N. This equivalence relation must fulfill two conditions: (i) it has only finitely many equivalence classes and (ii) for every theory atom A in N it holds that A is satisfied by all elements in an equivalence class or by none. Unbounded region equivalence $\widehat{\simeq}_\kappa^\eta$ is such an equivalence class for BS(BD):

Definition 27 (Unbounded Region Equivalence $\widehat{\simeq}_\kappa^\eta$ [24, 25])**.** We define the equivalence relation $\widehat{\simeq}_\kappa^\eta$ on $\mathcal{R} = \bigcup_{k=0}^{\eta} \mathbb{Q}^k$ in such a way that $\bar{r} \,\widehat{\simeq}_\kappa^\eta\, \bar{s}$ for $\bar{r}, \bar{s} \in \mathcal{R}$ if and only if

1. $\bar{r}$ and $\bar{s}$ have the same dimension, i.e., $\bar{r} = \langle r_1, \ldots, r_m \rangle$ and $\bar{s} = \langle s_1, \ldots, s_m \rangle$;
2. for every i
   (a) $r_i > \kappa$ if and only if $s_i > \kappa$,
   (b) $r_i < -\kappa$ if and only if $s_i < -\kappa$,
   (c) if $-\kappa < r_i, s_i < \kappa$, then $\lfloor r_i \rfloor = \lfloor s_i \rfloor$, and
   (d) if $-\kappa < r_i, s_i < \kappa$, then $\mathrm{fr}(r_i) = 0$ if and only if $\mathrm{fr}(s_i) = 0$;
3. for all i, j
   (a) if $r_i, r_j > \kappa$ or $r_i, r_j < -\kappa$, then $r_i \leq r_j$ if and only if $s_i \leq s_j$,
   (b) if $-\kappa < r_i, r_j < \kappa$, then $\mathrm{fr}(r_i) \leq \mathrm{fr}(r_j)$ if and only if $\mathrm{fr}(s_i) \leq \mathrm{fr}(s_j)$.

Corollary 28. Let A be a BD atom and let $\sigma_r = \{x_1 \mapsto r_1, \ldots, x_m \mapsto r_m\}$ and $\sigma_s = \{x_1 \mapsto s_1, \ldots, x_m \mapsto s_m\}$ be two grounding assignments for A such that $\bar{r} \,\widehat{\simeq}_\kappa^\eta\, \bar{s}$. Then $A \cdot \sigma_r$ is satisfied if and only if $A \cdot \sigma_s$ is satisfied.

The first condition for our equivalence class is necessary so we can express all argument tuples over our theories potentially infinite carrier sets with argument tuples over just a finite set of sample elements (i.e., our constants B). The second condition is necessary because the algebras we are looking for are supposed to be uniform in a given equivalence class, i.e., for every atom A in N it holds that A is satisfied by all elements in an equivalence class or by none.

Definition 29 ($\widehat{\simeq}_\kappa^\eta$-Uniform Algebras [24, 25])**.** Consider an algebra $\mathcal{A}$ for N. We call $\mathcal{A}$ *$\widehat{\simeq}_\kappa^\eta$-uniform* over N if it interprets all $\widehat{\simeq}_\kappa^\eta$-equivalence classes uniformly, i.e., for all predicates P in N and all $\bar{r} \,\widehat{\simeq}_\kappa^\eta\, \bar{s}$ with $m = \mathrm{arity}(P)$ and $\bar{r}, \bar{s} \in \mathbb{Q}^m$ it holds that $\bar{r} \in P^{\mathcal{A}}$ if and only if $\bar{s} \in P^{\mathcal{A}}$.

Based on these definitions, an extraction criterion guarantees the following properties for a stuck state $(M'; N; U; B; k; \top)$: (i) our trail can be extended to $M = M', M^p$ in such a way that there exists an argument tuple in B^m for every equivalence class and (ii) the literals in $\mathrm{fgd}(M')$ describe a uniform model, i.e., for every literal $|L| \in \mathrm{atoms}(N)$ it holds that $L \cdot \sigma \in M'$ if and only if $L \cdot \tau \in M'$, where σ and τ are two grounding substitutions over B that belong to the same equivalence class.

Definition 30 ($\widehat{\simeq}_\kappa^\eta$-Extraction-Criterion)**.** A stuck state $(M'; N; U; B; k; \top)$ fulfills the *$\widehat{\simeq}_\kappa^\eta$-extraction-criterion* if:

1. $B = \{b_1, ..., b_{|B|}\}$ is large enough, i.e., $|B| \geq 2\kappa \cdot (\eta + 1) + 2\eta + 1$
2. M' has a *$\widehat{\simeq}_\kappa^\eta$-uniform trail extension* M, M^p such that $M' \wedge M^p \wedge \mathrm{adiff}(B)$ is satisfiable and M^p is a sequence of theory atoms constructed as follows:[1]
   (a) Let $\pi : \mathbb{N} \to \{-\kappa - 1, -\kappa, \ldots, \kappa - 1, \kappa\}$ be the function with $\pi(i) = -\kappa - 1 + \lfloor i/(\eta + 1) \rfloor$ for $1 \leq i \leq 2\kappa \cdot (\eta + 1) + \eta$, and $\pi(i) = \kappa$ for $2\kappa \cdot (\eta + 1) + \eta < i \leq |B|$.

[1] The added theory atoms correspond exactly to the different cases in the unbounded region equivalence relation.

(b) Let $\rho : \mathbb{N} \to \mathbb{N}$ be the function with $\rho(i) = i\%(\eta+1)$ for $1 \leq i \leq 2\kappa\cdot(\eta+1)+\eta$, and $\rho(i) = i-2\kappa\cdot(\eta+1)+\eta+1$ for $2\kappa\cdot(\eta+1)+\eta < i \leq |B|$.

(c) Intuitively, we use $\pi(i)$ and $\rho(i)$ to partition the constants in B over the intervals $(-\infty,-\kappa), [-\kappa,-\kappa+1), \ldots, [\kappa-1,-\kappa)[\kappa,\infty)$ such that the interval $(-\infty,-\kappa)$ contains η constants, each interval $[i,i+1)$ with $-\kappa \leq i < \kappa$ contains $\eta+1$ constants, and the interval $[\kappa,\infty)$ contains at least $\eta+1$ constants.

(d) M^p contains $b_i < b_{i+1}$ for $1 \leq i < |B|$.

(e) M^p contains $b_i = k$ for $-\kappa \leq \pi(i) = k \leq \kappa$ and $\rho(i) = 0$.

(f) M^p contains $b_i < k+1$ for $-\kappa - 1 \leq \pi(i) = k < \kappa$ and $\rho(i) > 0$.

(g) M^p contains $b_i > k$ for $-\kappa \leq \pi(i) = k \leq \kappa$ and $\rho(i) > 0$.

(h) M^p contains $b_i - b_j \triangleleft 1$ for $-\kappa \leq \pi(j) = \pi(i) - 1 < \kappa - 1$, $\rho(i) \triangleleft \rho(j)$, and $\triangleleft \in \{<, =, >\}$.

3. $\mathrm{fgd}(M', M^p)$ is $\widehat{\simeq}^\eta_\kappa$-uniform, i.e., if $P(\bar{r}) \in \mathrm{fgd}(M', M^p)$ and $\bar{r} \mathrel{\widehat{\simeq}^\eta_\kappa} \bar{s}$ for $m = \mathrm{arity}(P)$ and $\bar{r}, \bar{s} \in B^m$, then $P(\bar{s}) \in \mathrm{fgd}(M', M^p)$.[2]

As a result of these properties, any assignment β for the constants B defines an algebra uniform to our equivalence relation.

Lemma 31 ($\widehat{\simeq}^\eta_\kappa$-Uniform Model Extraction)**.** Let $(M'; N; U; B; k; \top)$ be a stuck SCL(T) state that fulfills the $\widehat{\simeq}^\eta_\kappa$*-extraction-criterion* and let $\beta : B \to \mathbb{Q}$ be a satisfying assignment for the $\widehat{\simeq}^\eta_\kappa$-uniform trail extension $M = M', M^p$. Then there exists a $\widehat{\simeq}^\eta_\kappa$-uniform algebra $\mathcal{A}$ satisfying N such that

$$P^{\mathcal{A}} = \{\bar{s} \in \mathbb{R}^m \mid \exists \bar{t} \in B^m, \bar{s} \mathrel{\widehat{\simeq}^\eta_\kappa} \langle \beta(t_1), \ldots, \beta(t_m)\rangle \text{ and } P(\bar{t}) \in M\}$$

for all predicates P of arity m in N.

Proof. We have to prove that $\mathcal{A}$ satisfies all clauses $C \in N$ for all substitutions $\sigma : \mathrm{vars}(C) \to \mathbb{R}$. We do so by selecting one arbitrary substitution σ and by defining $Q = \{q_j \in (0,1) \mid \exists q' \in \mathrm{codom}(\sigma) \cap [-\kappa,\kappa], \mathrm{fr}(q') = q_j \neq 0\}$, i.e., the set of fractional parts that occur in σ's codomain. Moreover, we assume that $q_1 < \ldots < q_n$ is the order of the elements in $Q = \{q_1, \ldots, q_n\}$. Since we chose B large enough, we can now create a substitution $\tau : \mathrm{codom}(\sigma) \to B$ such that $\tau(q') = b_i$ if:

1. $-\kappa \leq \lfloor q' \rfloor < \kappa$, $\pi(i) = \lfloor q' \rfloor$, $\mathrm{fr}(q') = q_j \in Q$, and $\rho(i) = j$,
2. $-\kappa \leq \lfloor q' \rfloor \leq \kappa$, $\pi(i) = \lfloor q' \rfloor$, $\mathrm{fr}(q') = 0$, and $\rho(i) = 0$,
3. if q' is the j-th smallest element in $\mathrm{codom}(\sigma)$ that is smaller than $-\kappa$ and $\pi(i) = -\kappa - 1$, or
4. if q' is the j-th smallest element in $\mathrm{codom}(\sigma)$ that is larger than κ and $\pi(i) = \kappa$.

If we concatenate σ and τ, we get $M \models C \cdot \sigma \cdot \tau$ because $(M'; N; U; B; k; \top)$ is a stuck state (i.e., $M \models M' \models \mathrm{gnd}_B(N)$, Lemma 12). If we concatenate σ, τ, and β together, we get a $\widehat{\simeq}^\eta_\kappa$-equivalent substitution $\sigma \cdot \tau \cdot \beta$ for all literals L in C, i.e., if $\mathrm{vars}(\bar{x}) = \mathrm{vars}(L)$, $\bar{x} \cdot \sigma = \bar{s}$ and $\bar{x} \cdot \sigma \cdot \tau \cdot \beta = \bar{r}$, then $\bar{s} \mathrel{\widehat{\simeq}^\eta_\kappa} \bar{r}$. The way we constructed $\mathcal{A}$ entails that $\mathcal{A} \models P(\bar{x}) \cdot \sigma$ if and only if $P(\bar{x}) \cdot \sigma \cdot \tau \in M$

[2] $\bar{r} \mathrel{\widehat{\simeq}^\eta_\kappa} \bar{s}$ can be checked by comparing the atoms in M^p or by fixing a satisfying assignment for $M' \wedge M^p \wedge \mathrm{adiff}(B)$.

because $\bar{x} \cdot \sigma \mathrel{\widehat{\simeq}}^{\eta}_{\kappa} \bar{x} \cdot \sigma \cdot \tau \cdot \beta$. Similarly, Corollary 28 entails that a theory atom $A \cdot \sigma$ is satisfied if and only if $A \cdot \sigma \cdot \tau \cdot \beta$ is satisfied. Hence, $\mathcal{A} \models C \cdot \sigma$ because $M \models C \cdot \sigma \cdot \tau$. □

If the rules of SCL(T) are applied in a way that all trail prefixes are explored, then SCL(T) is also guaranteed to visit a stuck state that fulfills the extraction criterion whenever there exists an algebra $\mathcal{A}$ that is uniform to our equivalence relation and satisfies N.

Lemma 32 ($\widehat{\simeq}^{\eta}_{\kappa}$-Uniform Model Guarantee)**.** Let $\mathcal{A}$ be a $\widehat{\simeq}^{\eta}_{\kappa}$-uniform algebra that satisfies N and let B be a sequence of constants that is large enough, i.e., $|B| \geq 2\kappa \cdot (\eta + 1) + 2\eta + 1$. Then any regular SCL(T) run starting in state $(\epsilon; N; U; B; 0; \top)$ (with $N \models_{\mathcal{H}} U$) will encounter a stuck state that satisfies the $\widehat{\simeq}^{\eta}_{\kappa}$-extraction-criterion if SCL(T) explores all possible trail prefixes for B.

Proof. We can construct a trail prefix M' for B that corresponds to a stuck state that satisfies the $\widehat{\simeq}^{\eta}_{\kappa}$-extraction-criterion and from which we can extract $\mathcal{A}$. The trail prefix M' is constructed as follows: First construct a set of numbers $Q = \{q_0, \ldots, q_\eta\} \subseteq [0, 1)$ such that $q_0 = 0$ and $q_0 < \ldots < q_\eta$. Next we construct a set of numbers

$$\widehat{Q} = \left\{\widehat{q}^{\,k}_j \mid q_j \in Q, k \in \{-\kappa - 1, \ldots, \kappa\}, \text{ and } \widehat{q}^{\,k}_j = q_j + k\right\} \cup$$
$$\left\{\widehat{q}^{\,\kappa}_j \mid i \in \mathbb{N}, 2\kappa \cdot (\eta + 1) + \eta < i \leq |B|, j = i - 2\kappa \cdot (\eta + 1) + \eta + 1, \widehat{q}^{\,\kappa}_j = j + \kappa\right\}.$$

Each element in $\widehat{q}^{\,k}_j \in \widehat{Q}$ corresponds to one constant b_i. We denote this by an assignment $\beta : B \to \widehat{Q}$ such that $\beta(b_i) = \widehat{q}^{\,k}_j$ if $\pi(i) = k$ and $\rho(i) = j$. Now given these sets our trail prefix M' contains $P(\bar{s})/\mathrm{comp}(P(\bar{s}))$ as a decision literal if and only if (i) P is a predicate in N, $m = \mathrm{arity}(P)$, $\bar{s} \in B^m$, $\bar{r} \in \widehat{Q}^m$, (ii) if $s_t = b_i$, then $r_t = \beta(b_i)$, and (iii) $\bar{r} \in P^{\mathcal{A}}/\bar{r} \notin P^{\mathcal{A}}$. Moreover, we add to M' the following theory atoms as part of the first decision: we add $L \cdot \sigma$ to M' if there exists a grounding substitution σ for $|L| \in \mathrm{bgd}(\mathrm{atoms}(\mathrm{gnd}_B(N)))$ such that $L \cdot \sigma \cdot \beta$ is satisfied.

The state $(M'; N; U; B; k; \top)$ (with $k = |\,\mathrm{fgd}(M')|$) is a reachable stuck state because (i) all atoms $|L| \in \mathrm{atoms}(\mathrm{gnd}_B(N))$ are defined in M', (ii) $M' \wedge \mathrm{adiff}(B)$ is satisfiable, e.g., by β, and (iii) $M' \wedge \mathrm{adiff}(B) \models \mathrm{gnd}_B(N)$ because for all $C \in N$, (iii.i) $\mathcal{A} \models C \cdot \sigma \cdot \beta$ and (iii.ii) $L \cdot \sigma \in C \cdot \sigma$ is in M' if and only if $\mathcal{A} \models L \cdot \sigma \cdot \beta$.

The state $(M'; N; U; B; k; \top)$ also has a $\widehat{\simeq}^{\eta}_{\kappa}$-uniform trail extension $M' \wedge M^p$ because β by definition also satisfies the atoms in M^p and $\mathrm{fgd}(M', M^p)$ is $\widehat{\simeq}^{\eta}_{\kappa}$-uniform because the literals $\mathrm{fgd}(M', M^p) \cdot \beta$ are $\widehat{\simeq}^{\eta}_{\kappa}$-uniform. □

The above lemma explains how we can construct conditions for clause sets N that guarantee that SCL(T) finds a satisfying algebra $\mathcal{A}$ for N (under certain fairness conditions). The condition just has to imply that N is satisfied by an algebra $\mathcal{A}$ that is uniform to our equivalence relation. In the case of BS(BD), we can prove this for all satisfiable clause sets N. This means SCL(T) can be turned into a decision procedure for BS(BD).

Lemma 33 (BS(BD) Uniform Satisfiability [24, 25])**.** If N is satisfiable, then it is satisfied by an $\widehat{\simeq}^{\eta}_{\kappa}$-uniform algebra $\mathcal{A}$.

Corollary 34 (SCL(T) Decides BS(BD))**.** SCL(T) is a decision procedure for BS(BD) if (i) Restart is only applied to stuck states, (ii) Grow is only applied

after a stuck state has been encountered for the current sequence of constants B, (iii) rules Grow, Restart, and Decide are operated in a fair way, such that no stuck state is visited more than once, and (iv) it explores all possible trail prefixes for some B with $|B| \geq 2\kappa \cdot (\eta + 1) + 2\eta + 1$.

Example 35 (B too Small for Model Extraction)**.** If our set of constants B is too small, then a stuck state might not imply an interpretation for all relevant $\widehat{\simeq}^{\eta}_{\kappa}$-equivalence-classes. The same is true, if our constants are distributed unfairly over $\mathbb{Q}$, e.g., there exist no or not enough constants to represent an interval $[i, i+1)$ for $-\kappa \leq i, < \kappa$. Consider $B = [a, b]$ and a satisfiable BS(BD) clause set

$$\begin{aligned} N = \{ \quad & 0 \leq x \wedge x < 1 \wedge -1 \leq y \wedge y < 0 \wedge x - y = 1 \| P(x,y), \\ & 0 \leq x \wedge x < 1 \wedge -1 \leq y \wedge y < 0 \wedge x - y \neq 1 \| \neg P(x,y) \}. \end{aligned}$$

Starting from state $(\epsilon; N; \emptyset; B; 0; \top)$, a regular run can derive the following trail

$$\begin{aligned} M = \quad & [P(b,a)^{0 \leq x \wedge x < 1 \wedge -1 \leq y \wedge y < 0 \wedge x - y = 1 \| P(x,y) \cdot \{y \mapsto a, x \mapsto b\}}, 0 \leq b, b < 1, \\ & -1 \leq a, a < 0, b - a = 1, P(a,a)^1, P(b,b)^2, P(a,b)^3] \end{aligned}$$

The state $(M; N; \emptyset; B; 0; \top)$ is stuck and $M \models_{\mathcal{H}} \mathrm{gnd}_B(N)$. However, M does not satisfy the $\widehat{\simeq}^{\eta}_{\kappa}$-extraction-criterion because B is too small. This makes sense because M does not define P for all $\widehat{\simeq}^{\eta}_{\kappa}$-equivalence-classes, e.g., in all algebras $P(x,y)$ should be false for $-1 \leq y < 0, 0 \leq x < 1, x - y \neq 1$. We need at least one additional constant for each of the intervals $[-1, 0)$ and $[0, 1)$, and two for the intervals $(-\infty, -1)$ and $[1, \infty)$.

Example 36 (Successful BS(BD) model extraction)**.** Consider $B = [a, b, c, d, e, f, g, h, i, j, k]$ and a satisfiable BS(BD) clause set $N = \{0 \leq x \wedge x < 1 \wedge -1 \leq y \wedge y < 0 \wedge x - y = 1 \| P(x,y), 0 \leq x \wedge x < 1 \wedge -1 \leq y \wedge y < 0 \wedge x - y \neq 1 \| \neg P(x,y)\}$. Starting from state $(\epsilon; N; \emptyset; B; 0; \top)$, a regular run can derive the trail $M = M', M''$, where M'' contains decisions $\neg P(x,y) \cdot \sigma$ for all groundings of $P(x,y)$ not defined in M' and

$$\begin{aligned} M' = \quad & [P(f,c)^{0 \leq x \wedge x < 1 \wedge -1 \leq y \wedge y < 0 \wedge x - y = 1 \| P(x,y) \cdot \{x \mapsto f, y \mapsto c\}}, \\ & 0 \leq f, f < 1, -1 \leq c, c < 0, f - c = 1, \\ & P(g,d)^{0 \leq x \wedge x < 1 \wedge -1 \leq y \wedge y < 0 \wedge x - y = 1 \| P(x,y) \cdot \{x \mapsto g, y \mapsto d\}}, \\ & 0 \leq g, g < 1, -1 \leq d, d < 0, g - d = 1, \\ & P(h,e)^{0 \leq x \wedge x < 1 \wedge -1 \leq y \wedge y < 0 \wedge x - y = 1 \| P(x,y) \cdot \{x \mapsto h, y \mapsto e\}}, \\ & 0 \leq h, h < 1, -1 \leq e, e < 0, h - e = 1, \\ & \neg P(f,d)^{0 \leq x \wedge x < 1 \wedge -1 \leq y \wedge y < 0 \wedge x - y \neq 1 \| \neg P(x,y) \cdot \{x \mapsto f, y \mapsto d\}}, f - d \neq 1, \\ & \neg P(f,e)^{0 \leq x \wedge x < 1 \wedge -1 \leq y \wedge y < 0 \wedge x - y \neq 1 \| \neg P(x,y) \cdot \{x \mapsto f, y \mapsto e\}}, f - e \neq 1, \\ & \neg P(g,c)^{0 \leq x \wedge x < 1 \wedge -1 \leq y \wedge y < 0 \wedge x - y \neq 1 \| \neg P(x,y) \cdot \{x \mapsto g, y \mapsto c\}}, g - c \neq 1, \\ & \neg P(g,e)^{0 \leq x \wedge x < 1 \wedge -1 \leq y \wedge y < 0 \wedge x - y \neq 1 \| \neg P(x,y) \cdot \{x \mapsto g, y \mapsto e\}}, g - e \neq 1, \\ & \neg P(h,c)^{0 \leq x \wedge x < 1 \wedge -1 \leq y \wedge y < 0 \wedge x - y \neq 1 \| \neg P(x,y) \cdot \{x \mapsto h, y \mapsto c\}}, h - c \neq 1, \\ & \neg P(h,d)^{0 \leq x \wedge x < 1 \wedge -1 \leq y \wedge y < 0 \wedge x - y \neq 1 \| \neg P(x,y) \cdot \{x \mapsto h, y \mapsto d\}}, h - d \neq 1] \end{aligned}$$

The state $(M; N; \emptyset; B; 0; \top)$ is stuck and $M \models_{\mathcal{H}} \mathrm{gnd}_B(N)$. Moreover, M satisfies the $\widehat{\simeq}^{\eta}_{\kappa}$-extraction-criterion with the $\widehat{\simeq}^{\eta}_{\kappa}$-uniform extension M^p such that:

$$\begin{aligned} M^p = \quad & [a < b, b < c, c < d, d < e, e < f, f < g, g < h, h < i, i < j, j < k, \\ & a < -1, b < -1, c = -1, -1 < d, d < 0, -1 < e, e < 0, f = 0, \\ & 0 < g, g < 1, 0 < h, h < 1, i = 1, 1 < j, 1 < k, \\ & h - e = 1, h - d > 1, g - d = 1, g - e < 1] \end{aligned}$$

One satisfying assignment for M, M^p is $\beta = \{a \mapsto -1.7, b \mapsto -1.3, c \mapsto -1, d \mapsto -0.7, e \mapsto -0.3, f \mapsto 0, g \mapsto 0.3, h \mapsto 0.7, i \mapsto 1, j \mapsto 2, k \mapsto 3\}$ The extracted algebra $\mathcal{A}$ looks as follows:

$$
\begin{aligned}
P^{\mathcal{A}} = \;& \{(x,y) \in \mathbb{R}^2 \mid (x,y) \mathrel{\widehat{\simeq}}^{\eta}_{\kappa} (0.7,-0.3)\} \cup \{(x,y) \in \mathbb{R}^2 \mid (x,y) \mathrel{\widehat{\simeq}}^{\eta}_{\kappa} (0.3,-0.7)\} \cup \\
& \{(x,y) \in \mathbb{R}^2 \mid (x,y) \mathrel{\widehat{\simeq}}^{\eta}_{\kappa} (0,-1)\} \\
= \;& \{(x,y) \in \mathbb{R}^2 \mid 0 \leq x < 1, -1 \leq y, y < 0, \mathrm{fr}(x) = \mathrm{fr}(y)\} \cup \\
& \{(x,y) \in \mathbb{R}^2 \mid 0 \leq x < 1, -1 \leq y, y < 0, \mathrm{fr}(x) = \mathrm{fr}(y)\} \cup \\
& \{(x,y) \in \mathbb{R}^2 \mid 0 \leq x < 1, -1 \leq y, y < 0, \mathrm{fr}(x) = \mathrm{fr}(y)\} \\
= \;& \{(x,y) \in \mathbb{R}^2 \mid 0 \leq x < 1, -1 \leq y, y < 0, x - y = 1\}
\end{aligned}
$$

Example 37 (Stuck State is not $\widehat{\simeq}^{\eta}_{\kappa}$-Uniform)**.** Consider almost the same run as in the previous example. The only difference is that M'' contains the decision $P(a,c)$. Then the state $(M; N; \emptyset; B; 0; \top)$ is reachable by a reasonable run, still stuck, $M \models_{\mathcal{H}} \mathrm{gnd}_B(N)$, and $\beta = \{a \mapsto -1.7, b \mapsto -1.3, c \mapsto -1, d \mapsto -0.7, e \mapsto -0.3, f \mapsto 0, g \mapsto 0.3, h \mapsto 0.7, i \mapsto 1, j \mapsto 2, k \mapsto 3\}$ is still a satisfying assignment for M, M^p. However, M does not satisfy the $\widehat{\simeq}^{\eta}_{\kappa}$-extraction-criterion because the predicates in M are not uniformly defined. To be more precise, we have two different definitions for P and the equivalence class $\{(x,y) \in \mathbb{Q}^2 \mid x < -\kappa = -1, y = -\kappa = -1\}$, viz., $P(a,c), \neg P(b,c) \in M$ and $(a,c) \mathrel{\widehat{\simeq}}^{\eta}_{\kappa} (b,c)$.

It might seem like a reasonable idea to improve SCL(T) for BS(BD) by adding the $\widehat{\simeq}^{\eta}_{\kappa}$-uniform trail extension M^p directly at the beginning to the trail. Intuitively, this will only remove stuck states that cannot fulfill the extraction criterion and we will still find a stuck state that satisfies the extraction criterion if there exists one. However, it is not always possible to get a resolution proof for unsatisfiability if we add the $\widehat{\simeq}^{\eta}_{\kappa}$-uniform trail extension M^p greedily.

Example 38 (No resolution within extraction criterion)**.** Consider the unsatisfiable BS(BD) clause set $N = \{0 < x \wedge x < 1 \wedge 0 < y \wedge y < 1 \wedge x - y < 0 \| P(y),$ $0 < x \wedge x < 1 \wedge 0 < y \wedge y < 1 \wedge x - y < 0 \| \neg P(x)\}$. The partition of constants defined in the $\widehat{\simeq}^{\eta}_{\kappa}$-uniform trail extension M^p assigns only two constants a, b to the interval $(0,1)$ because no clause contains more than two variables. It is however impossible to get a refutation proof for the unsatisfiability of the above two clauses if we only have two constants $a < b$ in the interval $(0,1)$. If we add the $\widehat{\simeq}^{\eta}_{\kappa}$-uniform trail extension M^p directly at the beginning of our regular SCL(T) run, then we must end up in a stuck state that contains the literals $P(b)$ and $\neg P(a)$ on the trail. This means that SCL(T) neither derives a clause $\Lambda \| \bot$, nor does it encounter a stuck state that has a uniform model.

5 SCL(T) Extensions and Discussion

We have presented the new calculus SCL(T) for pure clause sets of first-order logic modulo a background theory. The calculus is sound and refutationally complete. It does not generate redundant clauses. Moreover, it constitutes a decision procedure for certain decidable fragments of pure clause sets, such as BS(BD), and can even return an explicit satisfying algebra $\mathcal{A}$ in the case that the clause set is satisfiable.

There are further extensions to pure clause sets that still enable a refutationally complete calculus. In particular, first-order function symbols that do not range into a background theory sort and equality. The properties of the SCL(T) calculus rely on finite trails with respect to a fixed, finite set B of constants. By adding non-constant first-order function symbols trails will typically be infinite without further restrictions. Finite trails can, e.g., still be obtained by limiting

nestings of function symbols in terms. Thus it seems to us that an extension to first-order function symbols that do not range into a background theory sort should be possible while keeping the properties of SCL(T). From an abstract point of view, also the addition of equality on the first-order side should be possible, because there exist complete procedures such as hierarchic superposition [1, 4]. Then also foreground function symbols may range into a background theory sort, but the respective terms have to satisfy further conditions in order to preserve completeness. However, even in the pure first-order case there has not been a convincing solution so far of how to combine equational reasoning with explicit model building. One challenge is how to learn a clause from a conflict out of a partial model assumption that enjoys ordering restrictions on terms occurring in equations. If this can be sufficiently solved, the respective calculus should also be extendable to a hierarchic set up.

An efficient implementation of SCL(T) requires efficient algorithmic solutions to a number of concepts out of the theory. For fast model building an efficient implementation of Propagate is needed. This was our motivation for adding the all-different constraints on the constants, because they enable syntactic testing for complementary or defined literals. In addition, satisfiability of constraints needs to be tested. The trail behaves like a stack and it is ground. This fits perfectly the strengths of SMT-style satisfiability testing. Dealing with the non-domain constants out of the set B needs some care. They behave completely symmetric with respect to the instantiation of clauses in $(N \cup U)$. An easy way to break symmetry here is the addition of linear ordering constraints on these constants. If more is known about the specific fragment some clause set N belongs to, additional constraints with respect to the constraints or domain constants out of $(N \cup U)$ may be added as well. This is for example the case for the BS(BD) fragment. We could simply add the atoms of the $\widehat{\simeq}^{\eta}_{\kappa}$-uniform trail extension M^p at the beginning to the trail. This would exclude many stuck states that cannot possible fulfill the extraction criterion and would therefore reduce the search space for SCL(T). Completeness would also be preserved because we either find a stuck state that satisfies the extraction criterion or the problem is unsatisfiable. Note, however, that we would not always get a refutation proof as a certificate of unsatisfiability.

If we add the $\widehat{\simeq}^{\eta}_{\kappa}$-uniform trail extension M^p at the beginning to the trail, then all groundings of theory atoms can be automatically simplified to true or false. This means we could implement propagation for SCL(T) over BS(BD) by feeding a SAT solver with all groundings of clauses and using its propagation module. We might even reduce the search space of the SAT solver by replacing all ground literals $|L| = P(\bar{s})$ to $|L'| = P(\bar{r})$ such that $\bar{r}$ is the minimal element in the $\bar{s}$ equivalence class. This would guarantee that only $\widehat{\simeq}^{\eta}_{\kappa}$-uniform literals are propagated. Conflict analysis would still need to be handled outside of the SAT solver so we can learn the much stronger non-ground clauses.

Checking whether a trail is $\widehat{\simeq}^{\eta}_{\kappa}$-uniform (as required by Definition 30) can be done efficiently (in run-time $O(|M| \log(|M|))$). We just have to sort the foreground literals $|L| = P(\bar{s})$ first by predicate P and then by the smallest vector $\bar{r}$ (according to a lexicographic order over B) such that $\bar{s} \,\widehat{\simeq}^{\eta}_{\kappa}\, \bar{r}$. The trail is $\widehat{\simeq}^{\eta}_{\kappa}$-uniform if and only if no two neighbors $P(\bar{r})$, $\neg P(\bar{s})$ in the sorted list have $\bar{s} \,\widehat{\simeq}^{\eta}_{\kappa}\, \bar{r}$. We could also add a new rule to the calculus that adds all $\widehat{\simeq}^{\eta}_{\kappa}$-uniform instances of the same literal as soon as one has been derived.

Exploring all trail prefixes, as required by Theorem 25 and Corollary 34,

requires book-keeping on visited stuck states and an efficient implementation of the rule Restart. The former can be done by actually learning new clauses that represent stuck states. Such clauses are not logical consequences out of N, so they have to be treated specially. In case of an application of Grow all these clauses and all the consequences thereof have to be updated. An easy solution would be to forget the clauses generated by stuck states. This can be efficiently implemented. Concerning the rule Restart, from the SAT world it is known that restarts do not have to be total [20], i.e., if a certain prefix of a trail will be reproduced after a restart, it can be left on the trail. It seems possible to extend this concept towards SCL(T).

As future work, we plan to implement SCL(T) and define extraction criteria for other (arithmetic) theories.

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

References

[1] L. Bachmair, H. Ganzinger, and U. Waldmann. Refutational theorem proving for hierarchic first-order theories. *Applicable Algebra in Engineering, Communication and Computing, AAECC*, 5(3/4):193–212, 1994.

[2] P. Baumgartner, A. Fuchs, and C. Tinelli. Lemma learning in the model evolution calculus. In *LPAR*, volume 4246 of *Lecture Notes in Computer Science*, pages 572–586. Springer, 2006.

[3] P. Baumgartner, A. Fuchs, and C. Tinelli. (LIA) - model evolution with linear integer arithmetic constraints. In *Logic for Programming, Artificial Intelligence, and Reasoning, 15th International Conference, LPAR 2008, Doha, Qatar, November 22-27, 2008. Proceedings*, volume 5330 of *Lecture Notes in Computer Science*, pages 258–273. Springer, 2008.

[4] P. Baumgartner and U. Waldmann. Hierarchic superposition revisited. In C. Lutz, U. Sattler, C. Tinelli, A. Turhan, and F. Wolter, editors, *Description Logic, Theory Combination, and All That - Essays Dedicated to Franz Baader on the Occasion of His 60th Birthday*, volume 11560 of *Lecture Notes in Computer Science*, pages 15–56. Springer, 2019.

[5] N. Bjørner, A. Gurfinkel, K. L. McMillan, and A. Rybalchenko. Horn clause solvers for program verification. In L. D. Beklemishev, A. Blass, N. Dershowitz, B. Finkbeiner, and W. Schulte, editors, *Fields of Logic and Computation II - Essays Dedicated to Yuri Gurevich on the Occasion of His 75th Birthday*, volume 9300 of *Lecture Notes in Computer Science*, pages 24–51. Springer, 2015.

[6] M. Bromberger, A. Fiori, and C. Weidenbach. SCL with theory constraints. *CoRR*, abs/2003.04627, 2020.

[7] L. M. de Moura and N. Bjørner. Engineering DPLL(T) + saturation. In A. Armando, P. Baumgartner, and G. Dowek, editors, *Automated Reasoning, 4th International Joint Conference, IJCAR 2008*, volume 5195 of *LNCS*, pages 475–490. Springer, 2008.

[8] L. M. de Moura and D. Jovanovic. A model-constructing satisfiability calculus. In R. Giacobazzi, J. Berdine, and I. Mastroeni, editors, *Verification, Model Checking, and Abstract Interpretation, 14th International Conference, VMCAI 2013, Rome, Italy, January 20-22, 2013. Proceedings*, volume 7737 of *Lecture Notes in Computer Science*, pages 1–12. Springer, 2013.

[9] P. J. Downey. Undecidability of presburger arithmetic with a single monadic predicate letter. Technical report, Center for Research in Computer Technology, Harvard University, 1972.

[10] R. Faqeh, C. Fetzer, H. Hermanns, J. Hoffmann, M. Klauck, M. A. Köhl, M. Steinmetz, and C. Weidenbach˙ Towards dynamic dependable systems through evidence-based continuous certification. In T. Margaria and B. Steffen, editors, *Leveraging Applications of Formal Methods, Verification and Validation: Discussion, Dissemination, Applications - 9th International Symposium, ISoLA 2020, Proceedings, Part II*, volume 12477 of *Lecture Notes in Computer Science*, 2020.

[11] A. Fiori and C. Weidenbach. SCL clause learning from simple models. In *Automated Deduction - CADE 27 - 27th International Conference on Automated Deduction, Natal, Brazil, August 27-30, 2019, Proceedings*, pages 233–249, 2019.

[12] Y. Ge and L. M. de Moura. Complete instantiation for quantified formulas in satisfiabiliby modulo theories. In *Computer Aided Verification, 21st International Conference, CAV 2009, Grenoble, France, June 26 - July 2, 2009. Proceedings*, volume 5643 of *Lecture Notes in Computer Science*, pages 306–320. Springer, 2009.

[13] M. Horbach, M. Voigt, and C. Weidenbach. The universal fragment of presburger arithmetic with unary uninterpreted predicates is undecidable. *CoRR*, abs/1703.01212, 2017.

[14] R. J. B. Jr. and R. Schrag. Using CSP look-back techniques to solve real-world SAT instances. In B. Kuipers and B. L. Webber, editors, *Proceedings of the Fourteenth National Conference on Artificial Intelligence and Ninth Innovative Applications of Artificial Intelligence Conference, AAAI 97, IAAI 97, July 27-31, 1997, Providence, Rhode Island, USA.*, pages 203–208, 1997.

[15] L. Kovács and A. Voronkov. First-order theorem proving and vampire. In N. Sharygina and H. Veith, editors, *Computer Aided Verification - 25th International Conference, CAV 2013, Saint Petersburg, Russia, July 13-19, 2013. Proceedings*, volume 8044 of *Lecture Notes in Computer Science*, pages 1–35. Springer, 2013.

[16] E. Kruglov and C. Weidenbach. Superposition decides the first-order logic fragment over ground theories. *Mathematics in Computer Science*, 6(4):427–456, 2012.

[17] M. L. Minsky. *Computation: Finite and Infinite Machines*. Automatic Computation. Prentice-Hall, 1967.

[18] R. Nieuwenhuis, A. Oliveras, and C. Tinelli. Solving SAT and SAT modulo theories: From an abstract Davis–Putnam–Logemann–Loveland procedure to DPLL(T). *Journal of the ACM*, 53:937–977, November 2006.

[19] V. Prevosto and U. Waldmann. SPASS+T. In G. Sutcliffe, R. Schmidt, and S. Schulz, editors, *ESCoR: FLoC'06 Workshop on Empirically Successful Computerized Reasoning*, volume 192 of *CEUR Workshop Proceedings*, pages 18–33, Seattle, WA, USA, 2006.

[20] A. Ramos, P. van der Tak, and M. Heule. Between restarts and backjumps. In K. A. Sakallah and L. Simon, editors, *Theory and Applications of Satisfiability Testing - SAT 2011 - 14th International Conference, SAT 2011, Ann Arbor, MI, USA, June 19-22, 2011. Proceedings*, volume 6695 of *Lecture Notes in Computer Science*, pages 216–229. Springer, 2011.

[21] A. Reynolds, H. Barbosa, and P. Fontaine. Revisiting enumerative instantiation. In D. Beyer and M. Huisman, editors, *Tools and Algorithms for the Construction and Analysis of Systems - 24th International Conference, TACAS 2018, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2018, Thessaloniki, Greece, April 14-20, 2018, Proceedings, Part II*, volume 10806 of *Lecture Notes in Computer Science*, pages 112–131. Springer, 2018.

[22] P. Rümmer. A constraint sequent calculus for first-order logic with linear integer arithmetic. In I. Cervesato, H. Veith, and A. Voronkov, editors, *LPAR*, volume 5330 of *LNCS*, pages 274–289. Springer, 2008.

[23] J. P. M. Silva and K. A. Sakallah. Grasp - a new search algorithm for satisfiability. In *International Conference on Computer Aided Design, ICCAD*, pages 220–227. IEEE Computer Society Press, 1996.

[24] M. Voigt. The Bernays-Schönfinkel-Ramsey fragment with bounded difference constraints over the reals is decidable. In C. Dixon and M. Finger, editors, *Frontiers of Combining Systems - 11th International Symposium, FroCoS 2017, Brasília, Brazil, September 27-29, 2017, Proceedings*, volume 10483 of *Lecture Notes in Computer Science*, pages 244–261. Springer, 2017.

[25] M. Voigt. *Decidable fragments of first-order logic and of first-order linear arithmetic with uninterpreted predicates*. PhD thesis, Saarland University, Saarbrücken, Germany, 2019.