Journal of Automated Reasoning manuscript No.
DOI 10.1007/s10817-022-09656-w in 67(1):6, 42 pages, March 2023

Semantically-Guided Goal-Sensitive Reasoning: Decision Procedures and the Koala Prover

Maria Paola Bonacina and Sarah Winkler

Received: 10 January 2022 / Accepted: 1 October 2022 / Published online: 11 January 2023

Abstract The main topic of this article are *SGGS decision procedures* for fragments of first-order logic without equality. SGGS (*Semantically-Guided Goal-Sensitive* reasoning) is an attractive basis for decision procedures, because it generalizes to first-order logic the Conflict-Driven Clause Learning (CDCL) procedure for propositional satisfiability. As SGGS is both refutationally complete and model-complete in the limit, SGGS decision procedures are *model-constructing.* We investigate the termination of SGGS with both positive and negative results: for example, SGGS decides *Datalog* and the *stratified fragment* (including *Effectively PRopositional logic*) that are relevant to many applications. Then we discover several *new decidable fragments*, by showing that SGGS decides them. These fragments have the *small model property*, as the cardinality of their SGGS-generated models can be upper bounded, and for most of them *termination tools* can be applied to test a set of clauses for membership. We also present the *first implementation of SGGS* – the Koala theorem prover – and we report on experiments with Koala.

Keywords SGGS · Decidable fragments · First-order logic · Hyperresolution · Ordered resolution · Rewriting · Termination tools

1 Introduction

Many applications of automated reasoning require to combine the decidability of satisfiability with an expressive logic. In first-order logic (FOL), validity, or equivalently unsatisfiability, is semidecidable, whereas satisfiability is not

This research was funded in part by grant "Ricerca di base 2017" of the Università degli Studi di Verona. Authors are listed alphabetically.

Maria Paola Bonacina, Università degli Studi di Verona, Strada Le Grazie 15, 37134 Verona, Italy, E-mail: mariapaola.bonacina@univr.it · Sarah Winkler, Free University of Bolzano/Bozen, Piazza Domenicani 3, 39100 Bolzano/Bozen, Italy, E-mail: sarwinkler@unibz.it

even semidecidable. Therefore, the quest for decidable fragments of FOL is key in advancing the field, and many classes of formulae were shown decidable (e.g., [35,25,43] for surveys). An approach to prove the decidability of a class is to show that a refutationally complete inference system for first-order theorem proving is guaranteed to terminate on all inputs in that class. It follows that any theorem proving strategy given by that inference system and a fair search plan is a decision procedure for satisfiability in that class.

In this paper we apply this approach to SGGS, or *Semantically-Guided Goal-Sensitive reasoning* [22,23]. Similar to *semantic resolution* [76] and *hyperresolution* [71], SGGS is *semantically guided* by a fixed *initial interpretation.* However, SGGS generates primarily instances of clauses, not resolvents. By this characteristic, SGGS is a descendant of *hyperlinking* [53] and *ordered semantic hyperlinking* [67]. Other methods with this characteristic include *hypertableaux* [10,8,11] and *Inst-Gen* [39,46]. Nonetheless, the essential features of SGGS set it apart from the theorem-proving methods based on resolution, instance generation, or tableaux.

SGGS is a generalization to FOL of the CDCL (*Conflict-Driven Clause Learning*) procedure for propositional satisfiability (SAT) [56]. Indeed, SGGS searches for a model of the input set of clauses by building candidate models, represented by *selected literals* on a *trail* of clauses. In this sense SGGS is *model-based*, meaning that the state of an SGGS-derivation is a representation of a candidate model (cf. [17] for a survey of first-order model-based methods). The initial interpretation in SGGS acts as a starting point, as the candidate models are built by selecting preferably literals that are not true in the initial interpretation, but may be needed to get a model of the clauses. In this process, a *conflict* may arise in the form of a *conflict clause*. SGGS applies a restricted form of resolution only to *explain* the conflict, which is then *solved* by moving the conflict clauses and flipping the sign of its selected literal. In this sense SGGS is also *conflict-driven* (cf. [15] for a survey of conflict-driven methods).

By these features, SGGS is a *first-order search-based satisfiability procedure*, similar to the methods that generalize CDCL to satisfiability modulo theories (SMT), such as CDCL($\mathcal{T}$) [63][1], CDCL(Γ+$\mathcal{T}$) [21], MCSAT [28], and CDSAT [18,19]. For CDCL, termination descends from the finitary nature of the SAT problem. The termination of CDCL($\mathcal{T}$) in its original formulation stems from the fact that it does not create new atoms. As CDCL(Γ+$\mathcal{T}$) integrates superposition in CDCL($\mathcal{T}$), the result is a semidecision procedure whose termination is guaranteed only under suitable hypotheses and using *speculative inferences* [21]. MCSAT, CDSAT, and an extension of CDCL($\mathcal{T}$) [7] create new terms and atoms, and they are proved to be terminating by showing that all new objects come from a *finite basis* [7,28,18].

The termination of SGGS is challenging, because SGGS is refutationally complete for FOL, and especially significant, since SGGS is *model complete in the limit*: given a satisfiable input, the limit of any fair SGGS-derivation

[1] The name of the procedure in [63] is DPLL($\mathcal{T}$), but the recent literature calls it CDCL($\mathcal{T}$), since the DPLL (Davis-Putnam-Logemann-Loveland) [27] and CDCL procedures have been recognized as distinct. The same remark applies to DPLL(Γ+$\mathcal{T}$) [21].

represents a model. Thus, model generation is guaranteed if termination is, and *SGGS-based decision procedures* are *model-constructing*, a standard feature for SAT and SMT procedures, but not for first-order theorem-proving methods.

With this motivation, we apply the *finite basis* approach to SGGS. A finite basis for SGGS is a finite subset of the Herbrand base of the input set of clauses. An SGGS-derivation *is in* a finite basis if all ground instances of all clauses generated during the derivation are made of atoms coming from the finite basis. Previous work showed that if the length of the trail during a fair SGGS-derivation is upper bounded, the derivation is finite [23, Thm. 6 and Cor. 2]. We show that if a fair SGGS-derivation is in a finite basis, the length of the trail is upper bounded, and hence the derivation is finite. Also, we prove that given a satisfiable input, the cardinality of the SGGS-generated model is upper bounded. These results hold *regardless of the initial Herbrand interpretation guiding SGGS*. If for all clause sets in a class $\mathcal{F}$ it is possible to identify a finite basis, $\mathcal{F}$ is SGGS-decidable and has the *small model property*.

It follows that SGGS with any guiding interpretation decides all fragments where the Herbrand base itself is finite, including the *Datalog language* (e.g., [26]), the *Bernays-Schönfinkel class* [14,68], whose clausal version is known as EPR for *Effectively PRopositional logic* [65,3,37], and the *stratified fragment* [1,47], which is the generalization of EPR to many-sorted logic.[2] EPR and the stratified fragment find application in verification (e.g., [64,57]), while Datalog is a fundamental language for deductive databases and knowledge representation, and has been applied also in connection with neural networks [58].

These positive results are balanced by negative ones: we show by counterexamples that SGGS with *sign-based semantic guidance* (i.e., either *all-negative* – all negative literals are true – or *all-positive* – all positive literals are true) does *not* decide the *Ackermann* [2,44,36], *monadic* [2,44,38], FO^2 [41,38], and *guarded* [4,29] fragments. Since the sets of clauses in these counterexamples admit *finite* model, these counterexamples also show that the existence of a finite model does *not* imply the termination of SGGS with sign-based semantic guidance. However, we also give examples where SGGS terminates and represents with a finite trail an infinite Herbrand model. Thus, the termination of SGGS does *not* imply the existence of a finite Herbrand model.

A clause is positively/negatively *ground-preserving*, or *range-restricted*, if all its variables occur in its negative/positive literals. This property is used in deductive databases [61,78], as it is also a property of Datalog clauses, in theorem proving [55,49,50], model building [34,25], and decision procedures [34,25,21,52,12]. The role of sign in the definition of ground-preserving clauses suggests to adopt sign-based semantic guidance for SGGS and compare it with hyperresolution, that is semantic resolution with sign-based semantic guidance. Under the assumption that the input clauses are ground-preserving, we prove two results: first, similar to hyperresolution, SGGS generates only ground clauses; second, SGGS *terminates whenever hyperresolution does*. It

[2] In this paper *stratified* is used in the sense of *sort-stratified* [1,47,57], not in the sense of stratified logic programs (e.g., [26]).

follows that SGGS decides all ground-preserving fragments decided by hyperresolution, such as the *positive variable dominated* (PVD) [34, 25] and the *bounded depth increase* (BDI) [52] fragments. However, many theorem-proving problems do not belong to any known decidable class.

Example 1 Problem HWV036-2 from TPTP 7.3.0 [77] specifies a full-adder in 51 clauses, including for instance:

$$\neg\mathsf{and}_{\mathsf{ok}}(x) \vee \neg\mathsf{1}(\mathsf{in}_1(x)) \vee \neg\mathsf{1}(\mathsf{in}_2(x)) \vee \mathsf{1}(\mathsf{out}_1(x)) \quad \neg\mathsf{fulladd}(x) \vee \mathsf{halfadd}(\mathsf{h}_1(x))$$
$$\neg\mathsf{halfadd}(x) \vee \mathsf{connection}(\mathsf{in}_1(x), \mathsf{in}_1(\mathsf{or}_1(x))) \quad \neg\mathsf{lor}(x) \vee \mathsf{or}_{\mathsf{ok}}(x) \vee \mathsf{error}(x).$$

This set is satisfiable, but it does not belong to any known decidable fragments.

In the second part of the paper we apply SGGS to find *new* decidable fragments. We define the *positively/negatively restrained* fragments by adding to ground-preservingness an ordering-based restriction. By distinguishing between sorts populated by finitely or infinitely many ground terms, we define the *positively/negatively sort-restrained* classes, where restrainedness is imposed only to the literals having infinitely many ground instances. These fragments generalize the respective restrained fragments and the stratified fragment, which represents the special case where there are finitely many ground terms for all sorts. The *sort-refined-PVD* class is defined analogously with the PVD restrictions in place of restrainedness, so that it generalizes the PVD and the stratified fragments. We show that *SGGS with sign-based semantic guidance decides all these new classes* by the finite basis approach, so that the new classes have the *small model property*. We prove that sign-based resolution strategies (e.g., hyperresolution and PO-resolution) decide the restrained fragments. However, they do *not* decide the sort-restrained and sort-refined-PVD classes, because they do not decide the stratified fragment.

The introduction of a new decidable class poses the problem of how to determine that a clause set belongs to the class and whether this test is decidable. We reduce the problem of deciding whether a clause set is restrained or sort-restrained to that of deciding whether rewriting by an associated rewrite system terminates. It follows that membership in these fragments is undecidable in general, but can be tested in practice by *termination tools for rewriting* such as TTT$_2$ [48] and AProVE [40].

In the experiments, we applied these tools to discover restrained and sort-restrained problems in the TPTP library [77]. The other decidability criteria (e.g., stratification, PVD) can also be tested automatically, resulting in a classification of TPTP problems. This allows us to evaluate empirically the relevance of the new classes and discover problems not previously known to be decidable. For instance, the axiomatization in Example 1 and all the TPTP problems that include it are restrained. Then we describe the Koala theorem prover, which is the *first implementation of SGGS*. We report on applying Koala to TPTP problems, including both SGGS-decidable and semidecidable problems. We present and analyze these experiments, which show promising performances especially on satisfiable problems.

The paper is organized as follows. After the basic definitions (Section 2) we give an overview of SGGS (Section 3). Section 4 presents the finite basis approach, and all the results about SGGS and already known decidable fragments. Section 5 introduces the new decidable fragments and contains the results showing that they are SGGS-decidable. Section 6 covers the reduction of membership in the new fragments to the termination of rewriting, the application of the termination tools, the Koala prover, and the experiments. Discussions of related and future work conclude the paper. A short version of this paper appeared [24].

2 Basic Definitions

A *signature* is given by a set Σ of sorts and a set of constant, function, and predicate symbols. We use $s, s_1, s_2, \ldots$ for sorts, $\mathsf{a}, \mathsf{b}, \mathsf{0}, \mathsf{1}$ for constants, $\mathsf{P}, \mathsf{Q}, \mathsf{R}$ for predicates, $\mathsf{f}, \mathsf{g}, \mathsf{h}, \mathsf{s}$ for functions, v, w, x, y, z for variables, t, u for terms, $\mathcal{V}ar(t)$ for the set of variables in t, $\mathcal{V}ar_s(t)$ for those of sort s, $top(t)$ for the top symbol of t. Sorts are nonempty (there is a ground term for every sort), and $t\colon s$ says that t has sort s. We use L, M, P, Q for literals, $at(L)$ for L's atom, $\alpha, \sigma, \vartheta, \tau$ for substitutions, C, D, E for clauses, that are disjunctions of literals where all variables are implicitly universally quantified, and S for a (finite) set of clauses, understood as the conjunction of its elements.

The *top* notation is extended to atoms, $\mathcal{V}ar$ and $\mathcal{V}ar_s$ to atoms, literals, and clauses, and *at* to sets of literals, clauses, and sets of clauses. A clause C is *positive* if all its literals are positive, *negative* if all its literals are negative, and *mixed* otherwise. C^+ and C^- denote the disjunctions of the positive and negative literals in C. A *unit clause* has exactly one literal, and a *Horn clause* has at most one positive literal. A Horn clause is a *fact* if it is a positive unit, a *query* if it is negative, and a *rule* if it is mixed. We use I and J for Herbrand interpretations and $\mathcal{I}$ for other interpretations. The symbol $\models$ is overloaded to mean satisfaction of a clause or set of clauses in an interpretation, validity in the sense of satisfaction in all Herbrand interpretations, and logical entailment.

Viewing terms as trees, the *depth* of a term t is defined as $\mathrm{depth}(t) = 0$, if t is a constant or a variable, and $\mathrm{depth}(t) = 1 + max\{\mathrm{depth}(t_i) : 1 \leqslant i \leqslant n\}$, if t is a compound term $\mathsf{f}(t_1, \ldots, t_n)$. The depth of a literal L is defined as $\mathrm{depth}(L) = \mathrm{depth}(at(L)) = 1 + max\{\mathrm{depth}(t_i) : 1 \leqslant i \leqslant n\}$ if $t_1, \ldots, t_n$ are the predicate's arguments in $at(L)$. By labeling arcs with natural numbers, every subterm has a *position* defined as the string of natural numbers from the root to the subterm. We use p, q, r, and o for positions. The subterm of t at position p, denoted as $t|_p$, is defined by $t|_\Lambda = t$, where Λ is the top position, and $f(t_1, \ldots, t_n)|_{ip} = t_i|_p$ for all i, $1 \leqslant i \leqslant n$. The notation $t = c[u]_p$ says that t is equal to a *context* c where u occurs as subterm at position p. A term t has *occurrence depth* k in atom L if $L|_p = t$ and k is the length of position p.

An ordering $>$ on terms is *well-founded*, if it admits no infinite descending chain, *stable*, if $t > u$ implies $t\sigma > u\sigma$ for all substitutions σ, *monotonic*, if $t > u$ implies $c[t]_p > c[u]_p$ for all contexts c and positions p, and has the *subterm property*, if $c[t]_p > t$ for all contexts c and positions p with $p \neq \Lambda$. A *simplification ordering* is stable, monotonic, and has the subterm property. A *complete simplification ordering* (CSO) is also total on ground terms. A simplification ordering is well-founded [30]. *Recursive path orderings* (RPO's) [30], *lexicographic (recursive) path orderings* (LPO's), and *Knuth-Bendix orderings* (KBO's) [45,54] employ a *precedence*, which is a partial ordering $\succ_p$ on the symbols in the signature. A KBO attributes non-negative *weights* to terms: all variables have weight w_0, a weight function w attributes a weight to every non-variable symbol, and the weight of a term is the sum of the weights of its symbols. RPO's, LPO's, and KBO's are simplification orderings. If $\succ_p$ is total, KBO's and LPO's are CSO's (see e.g., [31] for a survey on orderings).

We recapitulate resolution [72], because resolution-based strategies appear in later sections. *Binary resolution* generates from *parents* $\neg L \vee C$ and $L' \vee D$ the *resolvent* $(C \vee D)\sigma$, if $L\sigma = L'\sigma$ with most general unifier (mgu) σ. *Factoring* generates from *parent* $L_1 \vee \ldots \vee L_k \vee C$ the *factor* $(L_1 \vee C)\sigma$, if $L_1\sigma = L_2\sigma = \ldots = L_k\sigma$ with mgu σ. Many refinements of resolution preserve its refutational completeness. *Positive resolution*, also known as the *P1-strategy* [71,42] or *P1-deduction* [66], requires that every binary resolution step has a positive parent. *Negative resolution*, also known as *all-negative-resolution* [66], requires that every binary resolution step has a negative parent. *Semantic resolution* [76] generates only resolvents that are false in a fixed guiding interpretation I. *Hyperresolution* [71] is semantic resolution where I is either the *all-negative interpretation* I^- or the *all-positive interpretation* I^+. *Positive hyperresolution* resolves a non-positive clause C, called the *nucleus*, with as many positive clauses, termed *satellites*, as needed to resolve away with a simultaneous mgu all literals in C^- and get a positive clause, which is false in I^-. *Negative hyperresolution* is defined dually. *Ordered resolution* [42] assumes a CSO $>$ on literals and requires that in every binary resolution step $\neg L\sigma$ is $>$-maximal in $(\neg L \vee C)\sigma$ and $L'\sigma$ is $>$-maximal in $(L' \vee D)\sigma$; and in every factoring step $L_1\sigma$ is $>$-maximal in $(L_1 \vee \ldots \vee L_k \vee C)\sigma$. PO-resolution adds the requirement that $L' \vee D$ is positive and drops the $>$-maximality constraint on $\neg L\sigma$.

3 SGGS: an Overview

SGGS [22,23] works with constrained clauses, written $A \rhd C$, where A is a constraint and C is a clause. The atomic constraints are *true*, *false*, $top(t) = f$, and $t \equiv u$, where $\equiv$ is identity. For ground terms t and u, $\models top(t) = f$ if the top symbol of t is f, and $\models t \equiv u$ if t and u are the same term. The negation, conjunction, and disjunction of constraints is a constraint. Any variable that appears in A and not in C is implicitly existentially quantified. Thus, if A is ground, either $\models A$ or $\models \neg A$, and if A is not ground, $\models A$ means that the existential closure of A is valid. A constraint is in *standard form* if it is *true*, *false*, or a conjunction of distinct atomic constraints of the form $x \not\equiv y$ or $top(x) \neq f$. SGGS keeps constraints in standard form [23, Sect. 7]. Substitutions are *sort-preserving* (i.e., $x\sigma$ has the same sort as x) so that instantiation respects sorts.

For a constrained clause $A \rhd C$ the set of its *constrained ground instances* (cgi's) is $Gr(A \rhd C) = \{C\vartheta : C\vartheta$ is ground and $\models A\vartheta\}$. Thus, $Gr(\mathit{false} \rhd C) = \emptyset$, $Gr(\mathit{true} \rhd C) = Gr(C)$, and $\mathit{true} \rhd C$ can be written C. Literals $A \rhd L$ and $B \rhd M$ *intersect* if $at(Gr(A \rhd L)) \cap at(Gr(B \rhd M)) \neq \emptyset$ and are *disjoint* otherwise. The notation Gr is extended to sets of atoms. Constraints can be omitted if irrelevant or for brevity.

Example 2 Given a signature with constant symbols $\mathsf{a}\colon s_1$ and $\mathsf{b}\colon s_2$, function symbol $\mathsf{f}\colon s_1 \to s_2$, and predicate symbol $\mathsf{P} \subseteq s_2 \times s_2$, the only term of sort s_1 is a, and the only terms of sort s_2 are b and $\mathsf{f}(\mathsf{a})$. Thus, $Gr(\mathsf{P}(x,y)) = \{\mathsf{P}(\mathsf{b},\mathsf{b}),\ \mathsf{P}(\mathsf{f}(\mathsf{a}),\mathsf{b}),\ \mathsf{P}(\mathsf{b},\mathsf{f}(\mathsf{a})),\ \mathsf{P}(\mathsf{f}(\mathsf{a}),\mathsf{f}(\mathsf{a}))\}$. Then, $\mathit{top}(x) \neq \mathsf{a} \rhd \mathsf{P}(\mathsf{f}(x),y)$ is equivalent to $\mathit{false} \rhd \mathsf{P}(\mathsf{f}(x),y)$, while $\mathit{top}(y) \neq \mathsf{a} \rhd \mathsf{P}(\mathsf{f}(x),y)$ is equivalent to $\mathit{true} \rhd \mathsf{P}(\mathsf{f}(x),y)$ with cgi's $\mathsf{P}(\mathsf{f}(\mathsf{a}),\mathsf{b})$ and $\mathsf{P}(\mathsf{f}(\mathsf{a}),\mathsf{f}(\mathsf{a}))$.

SGGS is *semantically guided* by an *initial interpretation* I: if $I \not\models S$, SGGS seeks a Herbrand model of S, by building candidate partial interpretations different from I, and using I as the default to complete them. If the empty clause $\bot$ arises in the process, unsatisfiability is reported. While I can be any Herbrand interpretation, in this section I is either I^- or I^+. If I is I^- (I^+) SGGS discovers which positive (negative) literals need to be true to satisfy S.

A literal L is *uniformly false* in an interpretation J if $J \models \neg L$, that is, if $J \models \neg L'$ for all $L' \in Gr(L)$. Then, L is said to be *I-true* if it is true in I, and *I-false* if it is uniformly false in I. A clause is *I-all-true* if all its literals are I-true, and *I-all-false* if all its literals are I-false. If $I = I^-$ negative literals are I-true, positive literals are I-false, negative clauses are I-all-true, positive clauses are I-all-false, and mixed clauses are neither. If $I = I^+$ positive literals are I-true, negative literals are I-false, positive clauses are I-all-true, negative clauses are I-all-false, and mixed clauses are neither.

SGGS builds a *trail Γ* of constrained clauses with *selected literals*. The selected literals form the partial interpretation represented by the trail. Initially the trail is empty, written ε. Then SGGS adds clauses forming a sequence $A_1 \rhd C_1[L_1], \ldots, A_n \rhd C_n[L_n]$ ($n \geq 1$). Clause $A_i \rhd C_i[L_i]$ is the clause at *index* i in Γ. The notation $C_i[L_i]$ means that L_i is the literal of C_i that is *selected*. SGGS requires that every literal in Γ is either I-true or I-false (trivial if I is I^+ or I^-). SGGS also requires that if clause $C_i[L_i]$ has I-false literals, then L_i is an I-false literal. The selected literal L_i is I-true only if $C_i[L_i]$ is an I-all-true clause. I-false literals are preferred for selection because $I \not\models S$ and hence SGGS tries to change something wrt I towards finding a model of S.

In order to see how the selected literals form a partial interpretation, let the *length* of a trail Γ, written $|\Gamma|$, be the number of clauses in Γ, and let $\Gamma|_j$ denote the *prefix* of length j of Γ. Then, the *partial interpretation* $I^p(\Gamma)$ represented by Γ is defined inductively as follows. If $\Gamma = \varepsilon$, then $I^p(\Gamma) = \emptyset$. If $\Gamma = A_1 \rhd C_1[L_1], \ldots, A_n \rhd C_n[L_n]$, we define $I^p(\Gamma)$ in terms of $I^p(\Gamma|_{n-1})$. Consider a cgi $C[L]$ of $A_n \rhd C_n[L_n]$. If $I^p(\Gamma|_{n-1}) \cap C[L] \neq \emptyset$, it means that $I^p(\Gamma|_{n-1})$ already satisfies $C[L]$. If $I^p(\Gamma|_{n-1}) \cap C[L] = \emptyset$, it means that $I^p(\Gamma|_{n-1})$ does not satisfy $C[L]$: if $\neg L \notin I^p(\Gamma|_{n-1})$, we can satisfy $C[L]$ by adding L to $I^p(\Gamma|_{n-1})$ to form $I^p(\Gamma)$. Such an instance $C[L]$ is a *proper (or productive)* sat : $\Gamma \rightsquigarrow$ satisfiable if $I[\Gamma] \models S$

extend : $\Gamma \rightsquigarrow \Gamma, A \rhd E[L]$ where $A \rhd E[L]$ is an extension clause

delete : $\Gamma \rightsquigarrow \Gamma'$ where Γ' is Γ with all disposable clauses removed

Assuming $B \rhd D[M] \in dp(\Gamma)$ and $at(Gr(A \rhd L)) \cap at(Gr(B \rhd M)) \neq \emptyset$:

s-split : $\Box B \rhd D[M] \Box A \rhd C[L] \Box \rightsquigarrow \Box B \rhd D[M] \Box split(C, D) \Box$
if M and L have the same sign

d-split : $\Box B \rhd D[M] \Box A \rhd C[L] \Box \rightsquigarrow \Box B \rhd D[M] \Box split(C, D) \Box$
if M and L have opposite sign

Note: $\Box$ is a place holder for anonymous subsequences that do not change

Fig. 1 SGGS rules for model search

constrained ground instance (pcgi) of $A_n \rhd C_n[L_n]$ and L is a pcgi of $A_n \rhd L_n$. Then, $I^p(\Gamma) = I^p(\Gamma|_{n-1}) \cup pcgi(A_n \rhd L_n, \Gamma)$, where $pcgi(A_n \rhd L_n, \Gamma)$ is the set of all the pcgi's of $A_n \rhd L_n$ in Γ.

From the partial interpretation $I^p(\Gamma)$ we get the *interpretation* $I[\Gamma]$ represented by trail Γ as follows: for all ground literals L, if $I^p(\Gamma)$ determines the truth value of L, then $I[\Gamma] \models L$ iff $I^p(\Gamma) \models L$; otherwise, $I[\Gamma] \models L$ iff $I \models L$. Suppose that all cgi's of $A_n \rhd C_n[L_n]$ are pcgi's: this clause contributes *all* the ground instances of its selected literal to $I^p(\Gamma)$. The longest prefix of Γ that is made of clauses with this property is called the *disjoint prefix* of Γ and is denoted $dp(\Gamma)$. The name descends from the fact that the selected literals of clauses in $dp(\Gamma)$ are all disjoint. Suppose that $I^p(\Gamma|_{n-1})$ satisfies *all* the cgi's of $A_n \rhd C_n[L_n]$ and hence $A_n \rhd C_n[L_n]$ itself. Such a clause is *disposable* and can be removed by *SGGS-deletion* (rule delete in Fig. 1). A clause $A_n \rhd C_n[L_n]$ is a *conflict clause*, if *all* the literals of C_n are uniformly false in $I[\Gamma]$.

An *SGGS-derivation* (named Θ if needed) is a series of trails $\Gamma_0 \vdash \Gamma_1 \vdash \ldots \Gamma_j \vdash \ldots$, where $\Gamma_0 = \varepsilon$, and $\forall j,\ j > 0$, SGGS generates Γ_j from Γ_{j-1} and S by applying either a *model-search rule* in Fig. 1 or a *conflict-solving rule* in Fig. 2. If $I[\Gamma] \models S$, a model has been found, and rule sat in Fig. 1 fires to report the success of the model search. If $\bot \in \Gamma$, it means that in the attempt to solve a conflict, the conflict-solving rule *SGGS-resolution* (resolve in Fig. 2) has generated the empty clause, which signals that the conflict cannot be solved, because S itself is unsatisfiable, so that rule unsat in Fig. 2 fires to report that a refutation has been found.

Otherwise, SGGS *makes progress* in two ways. If $\Gamma = dp(\Gamma)$, the trail is in order, but since $I[\Gamma] \not\models S$, there exists some $C' \in Gr(C)$ for $C \in S$, such that $I[\Gamma] \not\models C'$. Then, SGGS applies *SGGS-extension* (rule extend in Fig. 1) to generate from C and Γ a clause $A \rhd E$, called *extension clause*, such that E is an instance of C and $C' \in Gr(A \rhd E)$, in order to extend $I^p(\Gamma)$ to try to satisfy C'. If $\Gamma \neq dp(\Gamma)$, the trail needs repair: either there are disposable clauses and *SGGS-deletion* removes them, or there are intersections between selected literals that can be exposed by *SGGS-splitting* (rules s-split and d-split in Fig. 1), or there is a conflict clause to be handled by the conflict-solving rules of Fig. 2. The following example illustrates the model-search rules, demonstrating how SGGS halts on the input set used to show that hyperresolution cannot decide EPR ([35, Ex. 4.8] and [25, Ex. 3.17]).

Example 3 The set S consisting of clauses

$$\begin{array}{llll} \mathsf{P}(x,x,\mathsf{a}) & (i) & \mathsf{P}(x,y,w) \vee \mathsf{P}(y,z,w) \vee \neg\mathsf{P}(x,z,w) & (ii) \\ \neg\mathsf{P}(x,x,\mathsf{b}) & (iii) & \mathsf{P}(x,z,w) \vee \neg\mathsf{P}(x,y,w) \vee \neg\mathsf{P}(y,z,w) & (iv) \end{array}$$

defeats hyperresolution, because (iv) is obtained from (ii) by flipping signs, and likewise for (iii) and (i) plus renaming the constant. Positive hyperresolution generates infinitely many clauses of the form $\mathsf{P}(x_1,x_2,\mathsf{a}) \vee \mathsf{P}(x_2,x_3,\mathsf{a}) \vee \cdots \vee \mathsf{P}(x_{n-1},x_n,\mathsf{a}) \vee \mathsf{P}(x_n,x_1,\mathsf{a})$, for $n \geqslant 2$, using (ii) as nucleus, (i) as initial satellite, and then each resulting hyperresolvent as next satellite. Negative hyperresolution generates infinitely many clauses of the form $\neg\mathsf{P}(x_1,x_2,\mathsf{b}) \vee \neg\mathsf{P}(x_2,x_3,\mathsf{b}) \vee \cdots \vee \neg\mathsf{P}(x_{n-1},x_n,\mathsf{b}) \vee \neg\mathsf{P}(x_n,x_1,\mathsf{b})$, for $n \geqslant 2$, using (iv) as nucleus, (iii) as initial satellite, and then each resulting hyperresolvent as next satellite. In contrast, SGGS detects that S is satisfiable with either I^- or I^+. Assume that $I = I^-$: all input clauses are satisfied except (i). Thus, SGGS-extension puts it on the trail selecting its single literal:

$$\Gamma_0 \colon \varepsilon \ \vdash \Gamma_1 \colon [\mathsf{P}(x,x,\mathsf{a})] \qquad\qquad \mathsf{extend}\ (i)$$

At this point, $I[\Gamma_1]$ satisfies $\mathsf{P}(x,x,\mathsf{a})$, but not the ground instances of clause (ii) where the third literal is an instance of $\neg\mathsf{P}(x,x,\mathsf{a})$. Thus, SGGS-extension unifies the third literal of clause (ii) with $[\mathsf{P}(x,x,\mathsf{a})]$, and adds to the trail the resulting instance of clause (ii):

$$\vdash \Gamma_2 \colon [\mathsf{P}(x,x,\mathsf{a})],\ \mathsf{P}(x,y,\mathsf{a}) \vee [\mathsf{P}(y,x,\mathsf{a})] \vee \neg\mathsf{P}(x,x,\mathsf{a}) \qquad \mathsf{extend}\ (ii)$$

An I^--false (i.e., positive) literal is selected in the added clause (choosing the other makes no difference). Also, SGGS *assigns* the literal $\neg\mathsf{P}(x,x,\mathsf{a})$ to $[\mathsf{P}(x,x,\mathsf{a})]$ to record that $\neg\mathsf{P}(x,x,\mathsf{a})$ (which is true in I^-) is uniformly false in $I^p(\Gamma_2)$, and hence in $I[\Gamma_2]$, due to the selection of $\mathsf{P}(x,x,\mathsf{a})$. $I[\Gamma_2]$ satisfies $\mathsf{P}(y,x,\mathsf{a})$ and hence all the ground instances of clause (ii) that had been lost in order to satisfy $\mathsf{P}(x,x,\mathsf{a})$. However, the selected literals in Γ_2 intersect. Thus, SGGS-splitting *partitions* the second clause into two clauses:

$$\begin{array}{ll} \vdash \Gamma_3 \colon [\mathsf{P}(x,x,\mathsf{a})],\ \mathsf{P}(x,x,\mathsf{a}) \vee [\mathsf{P}(x,x,\mathsf{a})] \vee \neg\mathsf{P}(x,x,\mathsf{a}), & \\ \qquad y \neq x \rhd \mathsf{P}(x,y,\mathsf{a}) \vee [\mathsf{P}(y,x,\mathsf{a})] \vee \neg\mathsf{P}(x,x,\mathsf{a}) & \mathsf{s\text{-}split} \end{array}$$

where both occurrences of $\neg\mathsf{P}(x,x,\mathsf{a})$ are assigned to $[\mathsf{P}(x,x,\mathsf{a})]$. A *partition* of a clause $A \rhd C[L]$ is a set of clauses $\{A_i \rhd C_i[L_i]\}_{i=1}^n$ that covers the same ground instances (i.e., $Gr(A \rhd C) = \bigcup_{i=1}^n \{Gr(A_i \rhd C_i)\}$), but such that the selected literals $A_i \rhd L_i$ are disjoint (cf. [23, Def. 13]). SGGS-splitting replaces a clause by the partition dictated by another clause: given a clause $B \rhd D[M]$

at a smaller index on the trail ($[\mathsf{P}(x,x,\mathsf{a})]$ in Γ_2) and a clause $A \rhd C[L]$ at a larger index on the trail ($\mathsf{P}(x,y,\mathsf{a}) \vee [\mathsf{P}(y,x,\mathsf{a})] \vee \neg\mathsf{P}(x,x,\mathsf{a})$ in Γ_2) such that their selected literals intersect, SGGS-splitting replaces $A \rhd C[L]$ by a partition $\{A_i \rhd C_i[L_i]\}_{i=1}^n$ where one of the selected literals $A_j \rhd L_j$ captures exactly the intersection between $B \rhd M$ and $A \rhd L$ (cf. [23, Def. 14]). This partition is called a *splitting* of $A \rhd C[L]$ by $B \rhd D[M]$ and is denoted *split*(C,D). Clause $A \rhd C[L]$ is the *split clause*. Clause $A_j \rhd C_j[L_j]$ is the *representative* of $B \rhd D[M]$ in *split*(C,D). The SGGS-splitting rule applied here is *s-splitting* (abbreviated s-split) for *splitting by similar literals*, because L and M have the same sign (cf. [23, Def. 23]). Now the second clause in Γ_3 is disposable, and SGGS-deletion removes it eliminating the intersection:

$$\vdash \Gamma_4\colon [\mathsf{P}(x,x,\mathsf{a})],\ y \neq x \rhd \mathsf{P}(x,y,\mathsf{a}) \vee [\mathsf{P}(y,x,\mathsf{a})] \vee \neg\mathsf{P}(x,x,\mathsf{a}) \qquad \mathsf{delete}$$

This holds in general: after an s-splitting, D's representative in *split*(C,D) is disposable (cf. [23, Lemma 3]). As $I[\Gamma_4] \models S$, rule sat reports satisfiable. The derivation with I^+ proceeds dually with (*iii*) and (*iv*).

As seen in Example 3, a derivation starts with an SGGS-extension that puts on the trail an I-all-false input clause and selects one of its literals. All such steps can be done as one and we assume they are. In general, SGGS-extension adds to the trail an instance of an input clause C called *main premise*. SGGS-extension generates this instance by simultaneously unifying the I-true literals $L_1,\ldots,L_n$ of C with as many I-false selected literals of opposite sign in the *side premises* $B_1 \rhd D_1[M_1],\ldots,B_n \rhd D_n[M_n]$ in *dp*(Γ).

Definition 1 (SGGS-extension scheme) Let S be the input clause set and Γ be the current trail. Let $C \in S$ be a clause such that $L_1,\ldots,L_n$ $(n \geq 0)$ are all its I-true literals, and $B_1 \rhd D_1[M_1],\ldots,B_n \rhd D_n[M_n]$ be clauses in *dp*(Γ) such that $M_1,\ldots,M_n$ are I-false. If $\forall j,\ 1 \leqslant j \leqslant n,\ L_j\alpha = \neg M_j\alpha$ with simultaneous mgu α, then *SGGS-extension* adds the *extension clause* $A \rhd E = (\bigwedge_{j=1}^n B_j\alpha) \rhd C\alpha$ to Γ.

An SGGS-extension is *conflicting* if the extension clause $A \rhd E$ is a conflict clause, *non-conflicting* otherwise, that is, if $A \rhd E$ has *pcgi*'s that get added to $I^p(\Gamma)$. Definition 1 is a simplification of the original [23, Def. 12] under the assumption that $I = I^-$ or $I = I^+$, and rule extend in Fig. 1 abstracts away for simplicity the details of the *SGGS-extension rules* [23, Defs. 18, 19, 20, and 21] that instantiate the SGGS-extension scheme.

As seen in Example 3, if the selection of an I-false literal M makes an I-true literal L on the trail uniformly false in $I^p(\Gamma)$, SGGS *assigns* L to the clause where M is selected (cf. [23, Defs. 8, 9]). These assignments record why literals that are true in I are uniformly false in $I[\Gamma]$. SGGS requires that an I-true literal L on the trail is assigned unless it is selected. Therefore, the SGGS rules establish or preserve assignments. For example, the I-true literals of an extension clause are assigned to the side premises; the I-true literals in the clauses of a partition inherit the assignments from the split clause;

unsat: $\Gamma \quad \rightsquigarrow \quad$ unsatisfiable $\qquad$ if $\bot \in \Gamma$

resolve: $\square B \rhd D[M] \square A \rhd C[L] \Gamma \quad \rightsquigarrow \quad \square B \rhd D[M] \square \mathit{Res}(C, D) \square \Gamma'$
where Γ' is Γ with all clauses with literals assigned to C removed

Assuming $A \rhd C[L] \in \mathit{dp}(\Gamma)$, $D[M]$ is I-all-true, and M is assigned to $A \rhd C[L]$:

move: $\square A \rhd C[L] \square B \rhd D[M] \square \quad \rightsquigarrow \quad \square B \rhd D[M] \; A \rhd C[L] \square\square$
if $\neg \mathit{Gr}(B \rhd M) = \mathit{pcgi}(A \rhd L, \Gamma)$ and no other literal of D is assigned to C

factor: $\square A \rhd C[L] \square B \rhd D[M] \square \quad \rightsquigarrow \quad \square A \rhd C[L] \square \mathit{split}(D, D_f) \square$
if there is a literal $Q \in D$, $Q \neq M$, s.t. Q is assigned to C and $Q\vartheta = M\vartheta$ (†)
where D_f is the factor $B\vartheta \rhd D[M]\vartheta$

l-split: $\square A \rhd C[L] \square B \rhd D[M] \square \quad \rightsquigarrow \quad \square \mathit{split}(C, D) \square B \rhd D[M] \square$
if $\neg \mathit{Gr}(B \rhd M) \subset \mathit{pcgi}(A \rhd L, \Gamma)$ and condition (†) does not hold

Fig. 2 SGGS rules for conflict solving

and the clauses with literals assigned to a split clause can be deleted after the splitting [23, Def. 36]. This assignment mechanism achieves *first-order propagation* in SGGS: for all I-all-true clauses $C[L]$ on the trail, either all literals of $C[L]$ are assigned (with L assigned rightmost) and $C[L]$ is a *conflict clause*, or all literals of $C[L]$ except L are assigned, which means that L is an *implied literal*, $C[L]$ is its *justification*, and $C[L]$ is in $\mathit{dp}(\Gamma)$. First-order propagation applies to I-all-true clauses because it is relative to I.

CDCL [56] uses the *two-watched-literals scheme* to detect conflict clauses and implied literals without checking the truth value (true, false, or undefined) of every literal in every clause. It suffices to watch two non-false (i.e., either true or undefined) literals per clause: if the clause has zero non-false literals, it is a conflict clause; if it has one, the literal is implied and the clause is its justification. Since the assignment mechanism is *built into the SGGS rules*, SGGS does *not* need a first-order analogue of the two-watched-literals scheme to compute propagations ex post. The dependencies among literals that determine the propagations are stored with the clauses.

In SGGS the assignment of I-true literals to clauses also drives the application of the conflict-solving rules in Fig. 2. Suppose that SGGS-extension appends to the trail a conflict clause $A \rhd C[L]$ with I-false literals. This means that L is I-false. Then *SGGS-resolution* [23, Def. 26] (rule resolve in Fig. 2) *explains* the conflict by resolving upon L in $A \rhd C[L]$ and M in $B \rhd D[M]$, where $B \rhd D[M]$ is the I-all-true clause in $\mathit{dp}(\Gamma)$ to which L is assigned.

Definition 2 (SGGS-Resolution) Let $B \rhd D[M]$ and $A \rhd C[L]$ be clauses in Γ such that $B \rhd D[M]$ is I-all-true, is in $\mathit{dp}(\Gamma)$, and occurs at a smaller index than $A \rhd C[L]$, L is I-false, $L = \neg M\vartheta$ for some substitution ϑ, and $A \models B\vartheta$. Then *SGGS-resolution* replaces $A \rhd C[L]$ by the *SGGS-resolvent* $\mathit{Res}(C, D) = A \rhd R$, where R is $(C \setminus \{L\}) \cup (D \setminus \{M\})\vartheta$.

The I-true literals in the resolvent inherit their assignments from the I-true literals in the parents. Since M is the implied literal in $B \rhd D[M]$, the resolvent is still a conflict clause. SGGS-extension ensures that every I-false literal in a conflict clause is assigned to a justification in $dp(\Gamma)$ (cf. [23, Def. 19]) and therefore can be resolved away. Thus, conflict explanation by one or more SGGS-resolution steps generates either $\bot$ or an I-all-true conflict clause.

Suppose that SGGS-extension appends to the trail an I-all-true conflict clause $B \rhd D[M]$, or that $B \rhd D[M]$ is the result of conflict explanation by SGGS-resolution. Then *SGGS-move* [23, Def. 25] (rule move in Fig. 2) moves $B \rhd D[M]$ to the left of the clause $A \rhd C[L]$ in $dp(\Gamma)$ to which M is assigned. The effect is to solve the conflict by *flipping* the I-true literal M from being uniformly false in $I[\Gamma]$ to being an implied literal with justification $B \rhd D[M]$. This is why the selected literal in an I-all-true conflict clause is the one assigned rightmost: when the clause moves left, all other I-true literals remain assigned. After the move, $B \rhd D[M]$ resolves with $A \rhd C[L]$. Prior to the move, $B \rhd D[M]$ may split $A \rhd C[L]$ by *left-splitting* [23, Def. 25] (rule l-split in Fig. 2), and then move to the left of its representative in the splitting. If $B \rhd D[M]$ has another literal Q that is assigned to $A \rhd C[L]$, has the same sign as M, and unifies with M, *SGGS-factoring* [23, Def. 27] (rule factor in Fig. 2) applies to avoid a situation where Q has nowhere to be assigned after the move.

The *fairness* of an SGGS-derivation involves several properties: an inference is applied whenever $\bot \notin \Gamma$ and $I[\Gamma] \not\models S$; no splitting is trivial;[3] SGGS-deletion is applied eagerly; all conflicting SGGS-extensions are followed right away by conflict solving; and inferences applying to shorter prefixes of the trail are never neglected in favor of others applying to longer prefixes (cf. [23, Defs. 32, 37, and 49]). The *limit* of a fair derivation $\Gamma_0 \vdash \Gamma_1 \vdash \dots \Gamma_j \vdash \Gamma_{j+1} \vdash \dots$ is the longest trail Γ_∞ such that $\forall i$, $i \leqslant |\Gamma_\infty|$, there exists an n_i such that $\forall j$, $j \geqslant n_i$, if $|\Gamma_j| \geq i$ then $\Gamma_j|_i \approx^c \Gamma_\infty|_i$, where $\approx^c$ is the equivalence associated to a *convergence ordering* $>^c$ on trails ([23, Defs. 46, 50]). In words, all prefixes of the trail stabilize eventually. Both the derivation and its limit Γ_∞ may be infinite, but if the derivation halts at stage k, then $\Gamma_\infty = \Gamma_k$. The following results are used in this paper:

1. *Finiteness of descending chains of length-bounded trails* [23, Thm. 6 and Cor. 2]: A chain $\Gamma_0 >^c \Gamma_1 >^c \dots \Gamma_j >^c \Gamma_{j+1} \dots$ where $\forall j$, $j \geq 0$, $|\Gamma_j| \leqslant n$, for some $n \geq 0$, is finite.
2. *Descending chain theorem* [23, Thm. 8]: A fair SGGS-derivation forms a descending chain $\Gamma_0 >^c \Gamma_1 >^c \dots >^c \Gamma_j >^c \Gamma_{j+1} \dots$.
3. *Completeness* [23, Thm. 9 and 11]: For all input clause sets S, initial interpretations I, and fair SGGS-derivations, if S is satisfiable, $I[\Gamma_\infty] \models S$ (*model completeness in the limit*), and if S is unsatisfiable, $\bot \in \Gamma_k$ for some k (*refutational completeness*).

Results (1) and (2) above lead to prove termination and decidability by showing that the length of trails in a fair SGGS-derivation is upper bounded.

[3] An SGGS-splitting is trivial if it produces a singleton partition, such as when trying to split a ground clause or trying to split a clause by a more general one.

4 SGGS and Known Decidable Fragments

In this section we use the concept of *finite basis* to ensure that the length of trails in a fair SGGS-derivation is upper bounded, so that the derivation is guaranteed to halt. If the input is satisfiable, the cardinality of the finite basis offers an upper bound on the cardinality of the generated model. The SGGS-decidability of *Datalog* and of the *stratified fragment* is a straightforward consequence. On the other hand, counterexamples show that SGGS with sign-based semantic guidance cannot decide other known decidable fragments. These derivations are useful to understand SGGS. Then we show that if clauses are *ground-preserving*, SGGS terminates whenever hyperresolution does.

4.1 SGGS-Derivations in a Finite Basis Are Finite

Let S be the input set of clauses, $\mathcal{H}$ its Herbrand universe, and $\mathcal{A}$ its Herbrand base. A finite subset $\mathcal{B} \subseteq \mathcal{A}$ is a *finite basis* for an SGGS-derivation if all cgi's of all clauses on the trail during the derivation are made of atoms in $\mathcal{B}$.

Definition 3 (SGGS-Derivation in a basis) A clause $A \rhd C$ *is in* $\mathcal{B}$ if $at(Gr(A \rhd C)) \subseteq \mathcal{B}$. A trail *is in* $\mathcal{B}$ if all its clauses are. An SGGS-derivation *is in* $\mathcal{B}$ if all its trails are.

The next lemma shows that the cardinality of $\mathcal{B}$ provides an upper bound on the length of the trail during a fair SGGS-derivation.

Lemma 1 *If a fair SGGS-derivation* $\Gamma_0 \vdash \Gamma_1 \vdash \ldots \Gamma_j \vdash \Gamma_{j+1} \vdash \ldots$ *is in a finite basis* $\mathcal{B}$*, then* $\forall j,\ j \geqslant 0,\ |\Gamma_j| \leqslant |\mathcal{B}|+1$*, and if* $dp(\Gamma_j) = \Gamma_j$ *then* $|\Gamma_j| \leqslant |\mathcal{B}|$.

Proof SGGS cannot do worse than generating a ground trail where every atom in $\mathcal{B}$ appears selected with either positive or negative sign: any trail with non-ground clauses cannot be longer, since a non-ground clause covers many (possibly infinitely many) ground instances. By fairness, if the trail contains an intersection given by clauses $C[L]$ and $D[L]$, or $C[L]$ and $D[\neg L]$ with $L \in \mathcal{B}$, the clause on the right is either deleted eagerly by SGGS-deletion, or replaced with a resolvent by SGGS-resolution before SGGS-extension applies. Thus, there can be at most one such intersection, and the first claim follows. The second claim holds, because $dp(\Gamma_j) = \Gamma_j$ implies that there is no such intersection.

By the descending chain theorem and the finiteness of descending chains of length-bounded trails, the following general result follows:

Theorem 1 *A fair SGGS-derivation in a finite basis is finite.*

If for all sets S of clauses in a fragment $\mathcal{F}$ there exists a finite basis $\mathcal{B}$, which may depend on S, such that all SGGS-derivations from S are in $\mathcal{B}$, all fair SGGS-derivations from problems in $\mathcal{F}$ terminate, and SGGS decides $\mathcal{F}$. Assuming for simplicity the one-sorted case, where the cardinality of a model is that of its domain, we show that $\mathcal{F}$ also has the *small model property*: every satisfiable clause set in $\mathcal{F}$ admits a model whose cardinality is upper bounded. Let $\mathcal{H}(\mathcal{B}) = \{t : t \text{ is a strict subterm of } L \text{ for } L \in \mathcal{B}\}$, where "strict" says that the elements of $\mathcal{B}$ are not included. Since $\mathcal{B}$ is finite, $\mathcal{H}(\mathcal{B})$ also is finite.

Theorem 2 *If a fair SGGS-derivation from a satisfiable set S of clauses is in a finite basis $\mathcal{B}$, then S has a model of cardinality $|\mathcal{H}(\mathcal{B})| + 1$ that can be extracted from the limit of the derivation.*

Proof Let I be the initial interpretation. By Theorem 1 the derivation halts with some trail Γ which is its limit. Since SGGS is model complete in the limit, $I[\Gamma] \models S$. The domain of $I[\Gamma]$ is $\mathcal{H}$, which is infinite in general. However, since the derivation is in $\mathcal{B}$, all cgi's of all clauses in Γ are in $\mathcal{B}$, and therefore we can extract from $I[\Gamma]$ a model J with domain $\mathcal{H}(\mathcal{B}) \uplus \{u\}$, where u is a new constant symbol. For every constant symbol c, let $c^J = c$ if $c \in \mathcal{H}(\mathcal{B})$, and $c^J = u$ otherwise; for every n-ary ($n \geqslant 1$) function symbol f, let $f^J(t_1, \ldots, t_n) = f(t_1^J, \ldots, t_n^J)$ if $f(t_1, \ldots, t_n) \in \mathcal{H}(\mathcal{B})$, and $f^J(t_1, \ldots, t_n) = u$ otherwise; for every predicate symbol P, $(t_1, \ldots, t_n) \in P^J$ if and only if $I[\Gamma] \models P(t_1, \ldots, t_n)$. Note that J is well-defined because if $f(t_1, \ldots, t_n) \in \mathcal{H}(\mathcal{B})$ then $t_1, \ldots, t_n$ are also, hence all terms are interpreted in $\mathcal{H}(\mathcal{B}) \uplus \{u\}$. As J agrees with $I[\Gamma]$ on all atoms, $J \models S$, and its cardinality is $|\mathcal{H}(\mathcal{B})|+1$ by construction.

In summary, the finite basis approach for SGGS yields *termination*, *decidability*, and the *small model property*.

4.2 SGGS Decides Datalog, EPR, and the Stratified Fragment

A set of Datalog clauses, or *Datalog program* (e.g., [26]), is a set of Horn clauses where (i) there are no functions, so that all terms are either constants or variables, (ii) every fact is ground, and (iii) every variable that occurs in the positive literal of a rule C also occurs in at least one negative literal of C. Since the Herbrand universe and the Herbrand base $\mathcal{A}$ of a Datalog program are *finite*, $\mathcal{A}$ itself is the finite basis, and Theorem 1, together with the completeness theorems for SGGS, yields the following.

Theorem 3 *Given a Datalog program S, every fair SGGS-derivation halts, is a refutation if S is unsatisfiable, and constructs a model of S if S is satisfiable.*

The *Bernays-Schönfinkel* (BS) class [14,68] includes the sentences of the form $\exists^*\forall^*\varphi$, where φ is a formula with no occurrences of either quantifiers or functions, while constants are allowed. The reduction of BS formulae to clausal form yields *Effectively PRopositional logic* [65,3,37], abbreviated EPR. The *stratified* fragment generalizes EPR to many-sorted logic [1,47,64].

A signature is *stratified* [1,47,64], if there is a well-founded ordering $<_s$ on the set Σ of sorts, and for all functions $f\colon s_1 \times \cdots \times s_n \to s$, it holds that $s_i >_s s$ for all i, $1 \leqslant i \leqslant n$. The *sort-dependency graph* displays dependencies between sorts: it is a directed graph such that the set of vertices is Σ and there is an arc from s to s' if and only if there is a function symbol $f\colon s_1 \times \cdots \times s_n \rightarrow s'$ such that $s_i = s$ for some i, $1 \leqslant i \leqslant n$. A sort s is *cyclic*, if there exists a non-trivial path (i.e., a path of length greater or equal than 1) from s to s in the graph, and *acyclic* otherwise. In a stratified signature *all sorts are acyclic.* If a sentence over a stratified signature belongs to the $\exists^*\forall^*$ fragment, Skolemization only introduces constants and preserves stratification. If there is only one sort, this fragment reduces to EPR, because stratification over a single sort implies that there are no function symbols. However, also stratified sentences with a prefix other than $\exists^*\forall^*$ can yield stratified clauses [57].

Example 4 Assume the signature from Example 2, which is stratified with ordering $s_1 >_s s_2$. The Skolemization of $\forall x \exists y.\, \mathsf{P}(\mathsf{f}(x), y)$ preserves stratification, as clause $\mathsf{P}(\mathsf{f}(x), \mathsf{g}(x))$ with Skolem symbol $\mathtt{g}\colon s_1 \rightarrow s_2$ is still stratified. On the other hand, the Skolemization of $\forall x \exists y.\, \mathsf{P}(\mathsf{f}(y), x)$ yields $\mathsf{P}(\mathsf{f}(\mathsf{g}(x)), x)$ with Skolem symbol $\mathtt{g}\colon s_2 \rightarrow s_1$, so that stratification is lost.

A set of clauses whose signature is stratified is also called stratified. Since stratification prevents building terms of unbounded depth, the Herbrand universe and the Herbrand base are again *finite*, and we have the next theorem.

Theorem 4 *Given a stratified input set S, every fair SGGS-derivation halts, is a refutation if S is unsatisfiable, and constructs a model of S if S is satisfiable.*

However, SGGS-derivations in EPR can be exponentially long, as in the following example with a clause set S_k that describes a k-digits binary counter. Let Q be a predicate symbol of arity k, and for all i, $1 \leqslant i \leqslant k$, let $\overline{\mathsf{0}}_i$, $\overline{\mathsf{1}}_i$, and $\overline{x}_i$ be i-tuples of $\mathsf{0}$'s, $\mathsf{1}$'s, and distinct variables $x_1, \ldots, x_i$, respectively. Then S_k consists of the following $k+2$ clauses, for $0 \leqslant m \leqslant k-1$:

$$C_0\colon \mathsf{Q}(\overline{\mathsf{0}}_k) \quad C_{m+1}\colon \neg\mathsf{Q}(\overline{x}_m, \mathsf{0}, \overline{\mathsf{1}}_{k-m-1}) \vee \mathsf{Q}(\overline{x}_m, \mathsf{1}, \overline{\mathsf{0}}_{k-m-1}) \quad C_{k+1}\colon \neg\mathsf{Q}(\overline{\mathsf{1}}_k).$$

This set was used in the context of an analysis of first-order theorem-proving strategies [66, Def. 2.4.10] to show that resolution can do better than hyperresolution or positive/negative resolution. Indeed, resolution offers a refutation in $2k{+}1$ steps [66, Thm. 2.4.11], whereas positive resolution and positive hyperresolution simulate the binary counter, and negative resolution and negative hyperresolution do the same counting in reverse, so that all four strategies generate exponentially long derivations [66, Thm. 2.4.12]. As these sign-based refinements of resolution only generate ground clauses from S_k, this set was also used to show that generating ground instances and applying a propositional proof system can do exponentially worse than resolution in EPR [59, Sect. 2.1]. A recent model-based clause-learning decision procedure for EPR named SCL and evolved from NRCL [3] also behaves exponentially on S_k [37, Sect. 4]. Unlike in Example 3, SGGS behaves like hyperresolution on S_k.

Example 5 Given as input the k-digits binary counter clause set S_k, and I^- as initial interpretation, SGGS generates a derivation that simulates binary counting with a series of $2^k{+}1$ SGGS-extension steps, each adding a clause:

$$\begin{array}{lll}
\Gamma_0 \colon \varepsilon & \vdash \Gamma_1 \colon [\mathsf{Q}(\overline{0}_k)] & \mathsf{extend}\ (C_0)\\
& \vdash \Gamma_2 \colon [\mathsf{Q}(\overline{0}_k)],\ \neg\mathsf{Q}(\overline{0}_k) \vee [\mathsf{Q}(\overline{0}_{k-1},1)] & \mathsf{extend}\ (C_{k-1})\\
& \vdash \Gamma_3 \colon \ldots, \neg\mathsf{Q}(\overline{0}_{k-1},1) \vee [\mathsf{Q}(\overline{0}_{k-2},1,0)] & \mathsf{extend}\ (C_{k-2})\\
& \vdash \Gamma_4 \colon \ldots, \neg\mathsf{Q}(\overline{0}_{k-2},1,0) \vee [\mathsf{Q}(\overline{0}_{k-2},1,1)] & \mathsf{extend}\ (C_{k-1})\\
& \vdash \Gamma_5 \colon \ldots, \neg\mathsf{Q}(\overline{0}_{k-2},1,1) \vee [\mathsf{Q}(\overline{0}_{k-3},1,0,0)] & \mathsf{extend}\ (C_{k-3})\\
& \ldots & \ldots\\
& \vdash \Gamma_{2^k-1} \colon \ldots, \neg\mathsf{Q}(\overline{1}_{k-2},0,1) \vee [\mathsf{Q}(\overline{1}_{k-1},0)] & \mathsf{extend}\ (C_{k-2})\\
& \vdash \Gamma_{2^k} \colon \ldots, \neg\mathsf{Q}(\overline{1}_{k-1},0) \vee [\mathsf{Q}(\overline{1}_k)] & \mathsf{extend}\ (C_{k-1})\\
& \vdash \Gamma_{2^k+1} \colon \ldots, \neg\mathsf{Q}(\overline{1}_{k-1},0) \vee [\mathsf{Q}(\overline{1}_k)],\ [\neg\mathsf{Q}(\overline{1}_k)] & \mathsf{extend}\ (C_{k+1}).
\end{array}$$

At this stage a conflict emerges with I^--all-true conflict clause $[\neg\mathsf{Q}(\overline{1}_k)]$. After another 2^{k+1} steps, alternating SGGS-move (abbreviated move) and SGGS-resolution (abbreviated resolve), unsatisfiability is detected:

$$\begin{array}{ll}
\vdash \Gamma_{2^k+2} \colon \ldots, [\neg\mathsf{Q}(\overline{1}_k)],\ \neg\mathsf{Q}(\overline{1}_{k-1},0) \vee [\mathsf{Q}(\overline{1}_k)] & \mathsf{move}\\
\vdash \Gamma_{2^k+3} \colon \ldots, [\neg\mathsf{Q}(\overline{1}_k)],\ [\neg\mathsf{Q}(\overline{1}_{k-1},0)] & \mathsf{resolve}\\
\vdash \Gamma_{2^k+4} \colon \ldots, [\neg\mathsf{Q}(\overline{1}_{k-1},0)],\ \neg\mathsf{Q}(\overline{1}_{k-2},0,1) \vee [\mathsf{Q}(\overline{1}_{k-1},0)],\ [\neg\mathsf{Q}(\overline{1}_k)] & \mathsf{move}\\
\vdash \Gamma_{2^k+5} \colon \ldots, [\neg\mathsf{Q}(\overline{1}_{k-1},0)],\ [\neg\mathsf{Q}(\overline{1}_{k-2},0,1)],\ [\neg\mathsf{Q}(\overline{1}_k)] & \mathsf{resolve}\\
\ldots & \ldots\\
\vdash \Gamma_{2^{k+2}} \colon [\neg\mathsf{Q}(\overline{0}_k)],\ [\mathsf{Q}(\overline{0}_k)],\ \ldots & \mathsf{move}\\
\vdash \Gamma_{2^{k+2}+1} \colon \bot, \ldots & \mathsf{resolve}
\end{array}$$

SGGS with I^+ also behaves exponentially operating the counter in reverse.

In essence, this set of clauses appears to defeat simultaneously sign-based semantic guidance, instance generation, and conflict-driven clause learning.

4.3 SGGS Does Not Decide the Ackermann, Monadic, and FO^2 Classes

In this section we show that SGGS with sign-based semantic guidance does not decide the Ackermann, monadic, and FO^2 fragments. Let φ be a formula with no occurrences of either quantifiers or functions, while constants are allowed. The *Ackermann* class comprises the sentences of the form $\exists^*\forall\exists^*\varphi$ [2,44,36]. The *monadic*, or *Löwenheim*, class contains the sentences where there are no functions and predicates are unary [2,44,38]. In the *two-variable* fragment, denoted FO^2, there are only two variables and no functions [41,38].

We consider three sets of clauses to illustrate how SGGS works. Two of them are well-known in the literature. As these clause sets admit *finite* models, the nontermination of SGGS in these examples shows that SGGS is *not* guaranteed to terminate whenever the input set has a finite model. We conclude the section with an example that shows that it is *not* the case that whenever SGGS terminates there is a finite Herbrand model. Indeed, a finite SGGS trail can represent an infinite Herbrand model by using non-ground selected literals (SGGS is not restricted to generate ground instances of clauses) and by borrowing infinitely many literals from the initial interpretation.

The first set is $S_0 = \{\mathsf{P}(x) \vee \mathsf{P}(\mathsf{f}(x)),\ \neg\mathsf{P}(x') \vee \neg\mathsf{P}(\mathsf{f}(x'))\}$ [44, Sect. 5]. Set S_0 is in FO^2, because there are only two variables; and it is in the Ackermann and monadic classes because it is obtained from the Skolemization of the sentence $\forall x \exists y.(\mathsf{P}(x) \vee \mathsf{P}(y)) \wedge (\neg\mathsf{P}(x) \vee \neg\mathsf{P}(y))$. S_0 has a finite model $\mathcal{I}$ with domain $\{0, 1\}$, interpreting P as $\mathsf{P}^{\mathcal{I}} = \{0\}$, and f as $\mathsf{f}^{\mathcal{I}}(0) = 1$ and $\mathsf{f}^{\mathcal{I}}(1) = 0$. Adding to the signature a constant a to form Herbrand universe and Herbrand base, S_0 has two infinite Herbrand models: $J_1 = \{\mathsf{P}(\mathsf{f}^{2k}(\mathsf{a})), \neg\mathsf{P}(\mathsf{f}^{2k+1}(\mathsf{a})) : k \geqslant 0\}$ and $J_2 = \{\neg\mathsf{P}(\mathsf{f}^{2k}(\mathsf{a})), \mathsf{P}(\mathsf{f}^{2k+1}(\mathsf{a})) : k \geqslant 0\}$.

The addition of $\mathsf{P}(\mathsf{a})$ selects only J_1 as Herbrand model, yielding a simpler problem $S_1 = \{\mathsf{P}(\mathsf{a}),\ \mathsf{P}(x) \vee \mathsf{P}(\mathsf{f}(x)),\ \neg\mathsf{P}(x') \vee \neg\mathsf{P}(\mathsf{f}(x'))\}$. S_1 is in the same classes (membership in the Ackermann and monadic classes stems from the Skolemization of $\exists w \forall x \exists y.\mathsf{P}(w) \wedge (\mathsf{P}(x) \vee \mathsf{P}(y)) \wedge (\neg\mathsf{P}(x) \vee \neg\mathsf{P}(y)))$, and it is satisfied by finite model $\mathcal{I}$ extended with $\mathsf{a}^{\mathcal{I}} = 0$.

By enriching the signature so that binary clauses are mixed one gets $S_2 = \{\mathsf{P}(\mathsf{a}),\ \neg\mathsf{P}(\mathsf{b}),\ \neg\mathsf{P}(x) \vee \mathsf{P}(\mathsf{f}(x)),\ \mathsf{P}(x') \vee \neg\mathsf{P}(\mathsf{g}(x'))\}$, which is in the same classes (membership in the Ackermann and monadic classes descends from the Skolemization of $\exists v \exists w \forall x \exists y \exists z.\mathsf{P}(v) \wedge \neg\mathsf{P}(w) \wedge (\neg\mathsf{P}(x) \vee \mathsf{P}(y)) \wedge (\mathsf{P}(x) \vee \neg\mathsf{P}(z)))$, has only Herbrand model $J_3 = \{\mathsf{P}(\mathsf{a}), \neg\mathsf{P}(\mathsf{b}), \mathsf{P}(\mathsf{f}^k(\mathsf{a})), \neg\mathsf{P}(\mathsf{g}^k(\mathsf{b})) : k \geqslant 0\}$, and finite model $\mathcal{I}$ extended with $\mathsf{b}^{\mathcal{I}} = 1$ and $\mathsf{g}^{\mathcal{I}} = \mathsf{f}^{\mathcal{I}}$. Problem S_2 is even simpler, because it is made of Horn clauses, and because with binary mixed clauses every sign-guided hyperinference unifies only one pair of literals.

Resolution, even with subsumption, generates infinitely many clauses from S_0, S_1, and S_2, and so does hyperresolution [44]. From S_0 positive hyperresolution generates $\{\mathsf{P}(x) \vee \mathsf{P}(\mathsf{f}^{2k+1}(x)) : k \geqslant 1\}$ and negative hyperresolution generates $\{\neg\mathsf{P}(x) \vee \neg\mathsf{P}(\mathsf{f}^{2k+1}(x)) : k \geqslant 1\}$. From S_1 positive hyperresolution also adds $\{\mathsf{P}(\mathsf{f}^{2k}(\mathsf{a}) : k \geqslant 1\}$, while negative hyperresolution also adds $\{\neg\mathsf{P}(\mathsf{f}^{2k+1}(\mathsf{a}) : k \geqslant 0\}$. From S_2 positive hyperresolution generates $\{\mathsf{P}(\mathsf{f}^k(\mathsf{a})) : k \geqslant 1\}$ and negative hyperresolution generates $\{\neg\mathsf{P}(\mathsf{g}^k(\mathsf{b})) : k \geqslant 1\}$. Ordered resolution with an ordering $>$ on literals such that $\mathsf{P}(\mathsf{f}(x)) > \mathsf{P}(x)$, and $\mathsf{P}(\mathsf{g}(x)) > \mathsf{P}(x)$ for S_2, plus tautology elimination for S_0 and S_1, terminates right away.

The first example of this section shows how SGGS generates infinite derivations from S_2, working to modify I^- or I^+ to get model J_3 in the limit. Set S_2 is so simple that derivations made only of SGGS-extensions are possible.

Example 6 Given S_2 and $I = I^-$, the SGGS-derivation begins by putting on trail Γ the I-all-false (i.e., positive) clause $\mathsf{P}(\mathsf{a})$. Thus, $I[\Gamma] \models \mathsf{P}(\mathsf{a})$, but $I[\Gamma] \not\models \neg\mathsf{P}(x) \vee \mathsf{P}(\mathsf{f}(x))$, and an infinite series of instances gets generated:

$\varepsilon \vdash [\mathsf{P}(\mathsf{a})]$ $\qquad$ extend

$$\begin{array}{ll}
\vdash [\mathsf{P(a)}],\ \neg\mathsf{P(a)} \vee [\mathsf{P(f(a))}] & \mathsf{extend}\\
\vdash [\mathsf{P(a)}],\ \neg\mathsf{P(a)} \vee [\mathsf{P(f(a))}],\ \neg\mathsf{P(f(a))} \vee [\mathsf{P(f(f(a)))}] & \mathsf{extend}\\
\vdash \ldots &
\end{array}$$

The derivation lists as selected the positive literals of J_3, while $I[\Gamma]$ gets the negative ones from I^-. If the initial interpretation is I^+, SGGS starts by putting $[\neg\mathsf{P(b)}]$ on the trail, and then generates the instances of $\mathsf{P}(x') \vee \neg\mathsf{P}(\mathsf{g}(x'))$ by an infinite series of SGGS-extensions. The derivation lists as selected the negative literals of J_3, as $I[\Gamma]$ imports the positive ones from I^+.

The next example shows how input set S_1 induces SGGS to embark in infinite derivations[4] aiming at reaching model J_1 in the limit.

Example 7 Given $S_1 = \{\mathsf{P(a)},\ \mathsf{P}(x) \vee \mathsf{P}(\mathsf{f}(x)),\ \neg\mathsf{P}(x') \vee \neg\mathsf{P}(\mathsf{f}(x'))\}$ and $I = I^-$, the SGGS-derivation starts by placing the I-all-false input clauses on the trail:

$$\varepsilon\ \vdash [\mathsf{P(a)}],\ [\mathsf{P}(x)] \vee \mathsf{P}(\mathsf{f}(x)) \qquad \mathsf{extend}$$

where either literal can be selected in the second clause. Selecting $\mathsf{P}(\mathsf{f}(x))$ avoids an intersection with $\mathsf{P(a)}$, but suppose that $\mathsf{P}(x)$ is selected. Then the first clause splits the second one by s-splitting and SGGS-deletion removes the representative because it is disposable. We abbreviate $top(x) \neq \mathsf{a} \rhd \varphi[x]$ as $\varphi[\mathsf{f}(x)]$ where x is a new variable.

$$\begin{array}{ll}
\vdash [\mathsf{P(a)}],\ [\mathsf{P(a)}] \vee \mathsf{P(f(a))},\ [\mathsf{P}(\mathsf{f}(x))] \vee \mathsf{P}(\mathsf{f}^2(x)) & \mathsf{s\text{-}split}\\
\vdash [\mathsf{P(a)}],\ [\mathsf{P}(\mathsf{f}(x))] \vee \mathsf{P}(\mathsf{f}^2(x)) & \mathsf{delete}
\end{array}$$

At this point $I[\Gamma]$ satisfies no instance of the I-all-true input clause $\neg\mathsf{P}(x') \vee \neg\mathsf{P}(\mathsf{f}(x'))$. SGGS-extension fires unifying the two literals of this clause with the two selected literals on the trail (the mgu is $\alpha = \{x' \leftarrow \mathsf{a},\ x \leftarrow \mathsf{a}\}$):

$$\vdash [\mathsf{P(a)}],\ [\mathsf{P}(\mathsf{f}(x))] \vee \mathsf{P}(\mathsf{f}^2(x)),\ \neg\mathsf{P(a)} \vee [\neg\mathsf{P(f(a))}] \qquad \mathsf{extend}$$

The added clause is a conflict clause with $\neg\mathsf{P(a)}$ assigned to the first clause, and $\neg\mathsf{P(f(a))}$ to the second one, so that $\neg\mathsf{P(f(a))}$ is selected, because the selected literal in an I-all-true clause, if selected, must be assigned rightmost. Since $\mathsf{P(f(a))}$ is less general than $\mathsf{P}(\mathsf{f}(x))$ ($\neg Gr(\neg\mathsf{P(f(a))}) \subset pcgi(\mathsf{P}(\mathsf{f}(x)), \Gamma)$), the third clause splits the second one by left splitting (abbreviated $\mathsf{l\text{-}split}$), which enables SGGS-move followed by SGGS-resolution:

$$\begin{array}{ll}
\vdash [\mathsf{P(a)}],\ [\mathsf{P(f(a))}] \vee \mathsf{P}(\mathsf{f}^2(\mathsf{a})),\ [\mathsf{P}(\mathsf{f}^2(x))] \vee \mathsf{P}(\mathsf{f}^3(x)), & \\
\quad \neg\mathsf{P(a)} \vee [\neg\mathsf{P(f(a))}] & \mathsf{l\text{-}split}\\
\vdash [\mathsf{P(a)}],\ \neg\mathsf{P(a)} \vee [\neg\mathsf{P(f(a))}],\ [\mathsf{P(f(a))}] \vee \mathsf{P}(\mathsf{f}^2(\mathsf{a})), & \\
\quad [\mathsf{P}(\mathsf{f}^2(x))] \vee \mathsf{P}(\mathsf{f}^3(x)) & \mathsf{move}\\
\vdash [\mathsf{P(a)}],\ \neg\mathsf{P(a)} \vee [\neg\mathsf{P(f(a))}],\ \neg\mathsf{P(a)} \vee [\mathsf{P}(\mathsf{f}^2(\mathsf{a}))], &
\end{array}$$

[4] The SGGS-derivation with I^- given for this set in [24, Ex. 11] is incorrect.

$$[\mathsf{P}(\mathsf{f}^2(x))] \vee \mathsf{P}(\mathsf{f}^3(x)) \qquad \mathsf{resolve}$$
$$\vdash \dots$$

The infinite derivation lists as selected the literals $\mathsf{P}(\mathsf{a})$, $\neg\mathsf{P}(\mathsf{f}(\mathsf{a}))$, $\mathsf{P}(\mathsf{f}^2(\mathsf{a})), \dots$ of model J_1, and so do three other infinite derivations, one with $I = I^-$ and $\mathsf{P}(\mathsf{f}(x))$ selected in the second extension clause, one with $I = I^+$ and $\mathsf{P}(x)$ selected, and one with $I = I^+$ and $\mathsf{P}(\mathsf{f}(x))$ selected.

The following example illustrates how SGGS generates infinite derivations from S_0 to get either model J_1 or model J_2 depending on literal selection.

Example 8 Given $S_0 = \{\mathsf{P}(x) \vee \mathsf{P}(\mathsf{f}(x)),\ \neg\mathsf{P}(x') \vee \neg\mathsf{P}(\mathsf{f}(x'))\}$, and $I = I^-$, the first SGGS-extension adds the I-all-false input clause

$$\varepsilon\ \vdash [\mathsf{P}(x)] \vee \mathsf{P}(\mathsf{f}(x)) \qquad \mathsf{extend}$$

where either literal can be selected and $\mathsf{P}(x)$ is. If $\mathsf{P}(x)$ is selected, SGGS builds model J_1, and if $\mathsf{P}(\mathsf{f}(x))$ is selected, SGGS builds model J_2. SGGS-extension applies next with $\neg\mathsf{P}(x') \vee \neg\mathsf{P}(\mathsf{f}(x'))$ as main premise and two variants $[\mathsf{P}(x_1)] \vee \mathsf{P}(\mathsf{f}(x_1))$ and $[\mathsf{P}(x_2)] \vee \mathsf{P}(\mathsf{f}(x_2))$ of the clause in Γ as side premises. There are two mgu's, hence two possible steps with extension clause $\neg\mathsf{P}(x) \vee \neg\mathsf{P}(\mathsf{f}(x))$: $\alpha_1 = \{x_1 \leftarrow x', x_2 \leftarrow \mathsf{f}(x')\}$ and $\alpha_2 = \{x_2 \leftarrow x', x_1 \leftarrow \mathsf{f}(x')\}$. If α_1 is applied, $\neg\mathsf{P}(x)$ is assigned to the first variant and $\neg\mathsf{P}(\mathsf{f}(x))$ to the second one, so that $\neg\mathsf{P}(\mathsf{f}(x))$ is selected, because the selected literal in an I-all-true clause, if assigned, must be assigned rightmost. If α_2 is applied, $\neg\mathsf{P}(\mathsf{f}(x))$ is assigned to the first variant and $\neg\mathsf{P}(x)$ to the second one, so that $\neg\mathsf{P}(x)$ is selected. Putting both variants on the trail is useless, since SGGS-deletion removes the second one, and both literals of the extension clause can be assigned to the first clause on the trail, but distinguishing the two mgu's is useful to see which literal gets selected in the extension clause. If α_2 is applied, the result is:

$$\vdash [\mathsf{P}(x)] \vee \mathsf{P}(\mathsf{f}(x)),\ [\neg\mathsf{P}(x)] \vee \neg\mathsf{P}(\mathsf{f}(x)) \qquad \mathsf{extend}$$

However at this point the derivation is stuck, because neither SGGS-move nor SGGS-factoring nor left splitting apply to the I-all-true conflict clause. SGGS-move does not apply because the second clause has two literals assigned to the first one. SGGS-factoring does not apply because the two literals do not unify. Left splitting does not apply because $\neg Gr(\neg\mathsf{P}(x)) = pcgi(\mathsf{P}(x), \Gamma)$. Since a stuck derivation is not fair, a stuck state must be avoided by looking ahead or undoing. If α_1 is applied, the derivation proceeds with left splitting:

$$\vdash [\mathsf{P}(x)] \vee \mathsf{P}(\mathsf{f}(x)),\ \neg\mathsf{P}(x) \vee [\neg\mathsf{P}(\mathsf{f}(x))] \qquad \mathsf{extend}$$
$$\vdash top(x) \neq \mathsf{f} \rhd [\mathsf{P}(x)] \vee \mathsf{P}(\mathsf{f}(x)),\ [\mathsf{P}(\mathsf{f}(x))] \vee \mathsf{P}(\mathsf{f}^2(x)) \qquad \mathsf{l\text{-}split}$$

where $\neg\mathsf{P}(x) \vee [\neg\mathsf{P}(\mathsf{f}(x))]$ is removed, because it has literals assigned to the split clause. SGGS-extension applies again with $\neg\mathsf{P}(x') \vee \neg\mathsf{P}(\mathsf{f}(x'))$ as main premise, the two clauses in Γ as side premises, and mgu $\alpha = \{x' \leftarrow x\}$:

$$\vdash top(x) \neq \mathsf{f} \rhd [\mathsf{P}(x)] \vee \mathsf{P}(\mathsf{f}(x)),\ [\mathsf{P}(\mathsf{f}(x))] \vee \mathsf{P}(\mathsf{f}^2(x)),$$

$$top(x) \neq \mathsf{f} \rhd \neg\mathsf{P}(x) \vee [\neg\mathsf{P}(\mathsf{f}(x))] \qquad \mathsf{extend}$$

The first and the second literal of the extension clause are assigned to the first and second clause, respectively, so that the second literal is selected. Then left splitting applies:

$$\begin{aligned}&\vdash top(x) \neq \mathsf{f} \rhd [\mathsf{P}(x)] \vee \mathsf{P}(\mathsf{f}(x)),\ top(x) \neq \mathsf{f} \rhd [\mathsf{P}(\mathsf{f}(x))] \vee \mathsf{P}(\mathsf{f}^2(x)),\\ &\quad [\mathsf{P}(\mathsf{f}^2(x))] \vee \mathsf{P}(\mathsf{f}^3(x)),\ top(x) \neq \mathsf{f} \rhd \neg\mathsf{P}(x) \vee [\neg\mathsf{P}(\mathsf{f}(x))] && \mathsf{l\text{-}split}\end{aligned}$$

so that SGGS-move and SGGS-resolution can solve the conflict:

$$\begin{aligned}&\vdash top(x) \neq \mathsf{f} \rhd [\mathsf{P}(x)] \vee \mathsf{P}(\mathsf{f}(x)),\ top(x) \neq \mathsf{f} \rhd \neg\mathsf{P}(x) \vee [\neg\mathsf{P}(\mathsf{f}(x))],\\ &\quad top(x) \neq \mathsf{f} \rhd [\mathsf{P}(\mathsf{f}(x))] \vee \mathsf{P}(\mathsf{f}^2(x)),\ [\mathsf{P}(\mathsf{f}^2(x))] \vee \mathsf{P}(\mathsf{f}^3(x)) && \mathsf{move}\\ &\vdash top(x) \neq \mathsf{f} \rhd [\mathsf{P}(x)] \vee \mathsf{P}(\mathsf{f}(x)),\ top(x) \neq \mathsf{f} \rhd \neg\mathsf{P}(x) \vee [\neg\mathsf{P}(\mathsf{f}(x))],\\ &\quad top(x) \neq \mathsf{f} \rhd \neg\mathsf{P}(x) \vee [\mathsf{P}(\mathsf{f}^2(x))],\ [\mathsf{P}(\mathsf{f}^2(x))] \vee \mathsf{P}(\mathsf{f}^3(x)) && \mathsf{resolve}\end{aligned}$$

As the selected literals of the third and fourth clauses intersect, s-splitting applies, followed by the deletion of the representative:

$$\begin{aligned}&\vdash top(x) \neq \mathsf{f} \rhd [\mathsf{P}(x)] \vee \mathsf{P}(\mathsf{f}(x)),\ top(x) \neq \mathsf{f} \rhd \neg\mathsf{P}(x) \vee [\neg\mathsf{P}(\mathsf{f}(x))],\\ &\quad top(x) \neq \mathsf{f} \rhd \neg\mathsf{P}(x) \vee [\mathsf{P}(\mathsf{f}^2(x))],\\ &\quad top(x) \neq \mathsf{f} \rhd [\mathsf{P}(\mathsf{f}^2(x))] \vee \mathsf{P}(\mathsf{f}^3(x)),\ [\mathsf{P}(\mathsf{f}^3(x))] \vee \mathsf{P}(\mathsf{f}^4(x)) && \mathsf{s\text{-}split}\\ &\vdash top(x) \neq \mathsf{f} \rhd [\mathsf{P}(x)] \vee \mathsf{P}(\mathsf{f}(x)),\ top(x) \neq \mathsf{f} \rhd \neg\mathsf{P}(x) \vee [\neg\mathsf{P}(\mathsf{f}(x))],\\ &\quad top(x) \neq \mathsf{f} \rhd \neg\mathsf{P}(x) \vee [\mathsf{P}(\mathsf{f}^2(x))],\ [\mathsf{P}(\mathsf{f}^3(x))] \vee \mathsf{P}(\mathsf{f}^4(x)) && \mathsf{delete}\\ &\vdash \ldots\end{aligned}$$

Since $top(x) \neq \mathsf{f}$ is satisfied by $\{x \leftarrow \mathsf{a}\}$ in the Herbrand universe, the infinite derivation is listing $J_1 = \{\mathsf{P}(\mathsf{a}),\ \neg\mathsf{P}(\mathsf{f}(\mathsf{a})),\ \mathsf{P}(\mathsf{f}^2(\mathsf{a})), \ldots\}$. If $\mathsf{P}(\mathsf{f}(x))$ is selected in the first clause, the same sequence of inference rules is applied:

$$\begin{aligned}\varepsilon\ &\vdash \mathsf{P}(x) \vee [\mathsf{P}(\mathsf{f}(x))] && \mathsf{extend}\\ &\vdash \mathsf{P}(x) \vee [\mathsf{P}(\mathsf{f}(x))],\ \neg\mathsf{P}(\mathsf{f}(x)) \vee [\neg\mathsf{P}(\mathsf{f}^2(x))] && \mathsf{extend}\\ &\vdash top(x) \neq \mathsf{f} \rhd \mathsf{P}(x) \vee [\mathsf{P}(\mathsf{f}(x))],\ \mathsf{P}(\mathsf{f}(x)) \vee [\mathsf{P}(\mathsf{f}^2(x))] && \mathsf{l\text{-}split}\\ &\vdash top(x) \neq \mathsf{f} \rhd \mathsf{P}(x) \vee [\mathsf{P}(\mathsf{f}(x))],\ \mathsf{P}(\mathsf{f}(x)) \vee [\mathsf{P}(\mathsf{f}^2(x))],\\ &\quad top(x) \neq \mathsf{f} \rhd \neg\mathsf{P}(\mathsf{f}(x)) \vee [\neg\mathsf{P}(\mathsf{f}^2(x))] && \mathsf{extend}\\ &\vdash top(x) \neq \mathsf{f} \rhd \mathsf{P}(x) \vee [\mathsf{P}(\mathsf{f}(x))],\ top(x) \neq \mathsf{f} \rhd \mathsf{P}(\mathsf{f}(x)) \vee [\mathsf{P}(\mathsf{f}^2(x))],\\ &\quad \mathsf{P}(\mathsf{f}^2(x)) \vee [\mathsf{P}(\mathsf{f}^3(x))],\ top(x) \neq \mathsf{f} \rhd \neg\mathsf{P}(\mathsf{f}(x)) \vee [\neg\mathsf{P}(\mathsf{f}^2(x))] && \mathsf{l\text{-}split}\\ &\vdash top(x) \neq \mathsf{f} \rhd \mathsf{P}(x) \vee [\mathsf{P}(\mathsf{f}(x))],\\ &\quad top(x) \neq \mathsf{f} \rhd \neg\mathsf{P}(\mathsf{f}(x)) \vee [\neg\mathsf{P}(\mathsf{f}^2(x))],\\ &\quad top(x) \neq \mathsf{f} \rhd \mathsf{P}(\mathsf{f}(x)) \vee [\mathsf{P}(\mathsf{f}^2(x))],\ \mathsf{P}(\mathsf{f}^2(x)) \vee [\mathsf{P}(\mathsf{f}^3(x))] && \mathsf{move}\\ &\vdash top(x) \neq \mathsf{f} \rhd \mathsf{P}(x) \vee [\mathsf{P}(\mathsf{f}(x))],\\ &\quad top(x) \neq \mathsf{f} \rhd \neg\mathsf{P}(\mathsf{f}(x)) \vee [\neg\mathsf{P}(\mathsf{f}^2(x))],\end{aligned}$$

$$\mathit{top}(x) \neq \mathsf{f} \rhd \neg\mathsf{P}(\mathsf{f}(x)) \vee [\mathsf{P}(\mathsf{f}(x))],\ \mathsf{P}(\mathsf{f}^2(x)) \vee [\mathsf{P}(\mathsf{f}^3(x))] \qquad \textsf{resolve}$$

Unlike in the first derivation, the resolvent is disposable and gets deleted:

$$\begin{aligned}&\vdash \mathit{top}(x) \neq \mathsf{f} \rhd \mathsf{P}(x) \vee [\mathsf{P}(\mathsf{f}(x))],\\ &\qquad \mathit{top}(x) \neq \mathsf{f} \rhd \neg\mathsf{P}(\mathsf{f}(x)) \vee [\neg\mathsf{P}(\mathsf{f}^2(x))],\ \mathsf{P}(\mathsf{f}^2(x)) \vee [\mathsf{P}(\mathsf{f}^3(x))] \qquad \textsf{delete}\end{aligned}$$

The derivation continues with SGGS-extension with $\neg\mathsf{P}(x') \vee \neg\mathsf{P}(\mathsf{f}(x'))$ as main premise, the third and first clauses in Γ as side premises, and mgu $\alpha = \{x' \leftarrow \mathsf{f}^3(y),\ x \leftarrow \mathsf{f}(y)\}$, renaming as y the x in the third clause of Γ:

$$\begin{aligned}&\vdash \mathit{top}(x) \neq \mathsf{f} \rhd \mathsf{P}(x) \vee [\mathsf{P}(\mathsf{f}(x))],\\ &\qquad \mathit{top}(x) \neq \mathsf{f} \rhd \neg\mathsf{P}(\mathsf{f}(x)) \vee [\neg\mathsf{P}(\mathsf{f}^2(x))],\ \mathsf{P}(\mathsf{f}^2(x)) \vee [\mathsf{P}(\mathsf{f}^3(x))],\\ &\qquad \mathit{top}(x) \neq \mathsf{f} \rhd \neg\mathsf{P}(\mathsf{f}^3(x)) \vee [\neg\mathsf{P}(\mathsf{f}^4(x))] \qquad \textsf{extend}\\ &\vdash \mathit{top}(x) \neq \mathsf{f} \rhd \mathsf{P}(x) \vee [\mathsf{P}(\mathsf{f}(x))],\\ &\qquad \mathit{top}(x) \neq \mathsf{f} \rhd \neg\mathsf{P}(\mathsf{f}(x)) \vee [\neg\mathsf{P}(\mathsf{f}^2(x))],\\ &\qquad \mathit{top}(x) \neq \mathsf{f} \rhd \mathsf{P}(\mathsf{f}^2(x)) \vee [\mathsf{P}(\mathsf{f}^3(x))],\ \mathsf{P}(\mathsf{f}^3(x)) \vee [\mathsf{P}(\mathsf{f}^4(x))] \qquad \textsf{l-split}\\ &\vdash \dots\end{aligned}$$

The first three selected literals are $\mathsf{P}(\mathsf{f}(\mathsf{a}))$, $\neg\mathsf{P}(\mathsf{f}^2(\mathsf{a}))$, and $\mathsf{P}(\mathsf{f}^3(\mathsf{a}))$, and since $I[\Gamma]$ gets $\neg\mathsf{P}(\mathsf{a})$ from I^-, model $J_2 = \{\neg\mathsf{P}(\mathsf{a}),\ \mathsf{P}(\mathsf{f}(\mathsf{a})),\ \neg\mathsf{P}(\mathsf{f}^2(\mathsf{a})),\ \mathsf{P}(\mathsf{f}^3(\mathsf{a}))\dots\}$ emerges. Since S_0 is symmetric with respect to sign, with $I = I^+$ one gets two derivations identical to those above, except that all signs of all literals on the trail are flipped: the first derivation yields J_2 and the second one yields J_1.

Terminating SGGS-derivations can capture infinite Herbrand models, as a finite SGGS trail can represent an infinite Herbrand model by using non-ground selected literals and by borrowing infinitely many literals from I.

Example 9 Let the input set of clauses be $S = \{(i)\ \mathsf{P}(x,\mathsf{a}),\ (ii)\ \neg\mathsf{P}(x,y) \vee \mathsf{R}(y) \vee \mathsf{P}(x,\mathsf{f}(y)),\ (iii)\ \neg\mathsf{R}(\mathsf{f}(x)) \vee \neg\mathsf{P}(x,\mathsf{f}(x))\}$. SGGS with I^+ halts after putting clause (iii) on the trail. If the first literal is selected, we have

$$\Gamma_0\colon \varepsilon\ \vdash \Gamma_1\colon\ [\neg\mathsf{R}(\mathsf{f}(x))] \vee \neg\mathsf{P}(x,\mathsf{f}(x)),$$

where $I^p(\Gamma_1) = \{\neg\mathsf{R}(\mathsf{f}^k(\mathsf{a}))\ :\ k \geqslant 1\}$ and $I[\Gamma_1]$ is the infinite Herbrand model given by $I^p(\Gamma_1)$ plus all the positive literals whose complement is not in $I^p(\Gamma_1)$. If the second literal is selected,

$$\Gamma_0\colon \varepsilon\ \vdash \Gamma_1\colon\ \neg\mathsf{R}(\mathsf{f}(x)) \vee [\neg\mathsf{P}(x,\mathsf{f}(x))],$$

we have $I^p(\Gamma_1) = \{\neg\mathsf{P}(\mathsf{f}^k(\mathsf{a}),\mathsf{f}^{k+1}(\mathsf{a}))\ :\ k \geqslant 0\}$ and $I[\Gamma_1]$ is the infinite Herbrand model given by $I^p(\Gamma_1)$ plus all the positive literals whose complement is not in $I^p(\Gamma_1)$. If I is I^-, the termination of SGGS depends on literal selection. The following derivation halts:

$$\Gamma_0\colon \varepsilon\ \vdash \Gamma_1\colon\ [\mathsf{P}(x,\mathsf{a})]\ \vdash\ \Gamma_2\colon [\mathsf{P}(x,\mathsf{a})],\ \neg\mathsf{P}(x,\mathsf{a}) \vee [\mathsf{R}(\mathsf{a})] \vee \mathsf{P}(x,\mathsf{f}(\mathsf{a})),$$

with $I^p(\Gamma_2) = \{\mathsf{P}(\mathsf{f}^k(\mathsf{a}),\mathsf{a}) : k \geqslant 0\} \cup \{\mathsf{R}(\mathsf{a})\}$ and $I[\Gamma_2]$ given by $I^p(\Gamma_2)$ plus all the negative literals whose atom is not in $I^p(\Gamma_2)$. If the last literal in the instances of $\neg\mathsf{P}(x,y) \vee \mathsf{R}(y) \vee \mathsf{P}(x,\mathsf{f}(y))$ is systematically selected, SGGS with I^- diverges:

$$\begin{aligned}\varepsilon \vdash\ & [\mathsf{P}(x,\mathsf{a})]\ \vdash\ [\mathsf{P}(x,\mathsf{a})],\ \neg\mathsf{P}(x,\mathsf{a}) \vee \mathsf{R}(\mathsf{a}) \vee [\mathsf{P}(x,\mathsf{f}(\mathsf{a}))]\\ \vdash\ & [\mathsf{P}(x,\mathsf{a})],\ \neg\mathsf{P}(x,\mathsf{a}) \vee \mathsf{R}(\mathsf{a}) \vee [\mathsf{P}(x,\mathsf{f}(\mathsf{a}))],\\ & \neg\mathsf{P}(x,\mathsf{f}(\mathsf{a})) \vee \mathsf{R}(\mathsf{f}(\mathsf{a})) \vee [\mathsf{P}(x,\mathsf{f}^2(\mathsf{a}))]\ \vdash\ \ldots.\end{aligned}$$

Hyperresolution generates infinitely many clauses from this set. For example, using (*ii*) as nucleus, (*i*) as initial satellite, and then each resulting hyperresolvent as next satellite, positive hyperresolution produces $\mathsf{R}(\mathsf{a}) \vee \mathsf{P}(x,\mathsf{f}(\mathsf{a}))$, $\mathsf{R}(\mathsf{a}) \vee \mathsf{R}(\mathsf{f}(\mathsf{a})) \vee \mathsf{P}(x,\mathsf{f}^2(\mathsf{a}))$, $\mathsf{R}(\mathsf{a}) \vee \mathsf{R}(\mathsf{f}(\mathsf{a})) \vee \mathsf{R}(\mathsf{f}^2(\mathsf{a})) \vee \mathsf{P}(x,\mathsf{f}^3(\mathsf{a}))$, and so on.

4.4 SGGS Does Not Decide the Guarded Fragment

In this section we show by counterexamples that SGGS with sign-based semantic guidance does not decide the *guarded fragment* [4,29]. The guarded fragment admits no function symbols and restricts quantification to the following schemes: $\forall\bar{y}.(R(\bar{x},\bar{y}) \supset \psi[\bar{x},\bar{y}])$ and $\exists\bar{y}.(R(\bar{x},\bar{y}) \wedge \psi[\bar{x},\bar{y}])$, where ψ is also a guarded formula, and all the variables that occur in ψ must appear in the atomic guard $R(\bar{x},\bar{y})$. For the fragments considered in the previous sections, the clausal version of a fragment contains the sets of clauses generated by transforming into clausal form the formulae of the fragment. For the guarded fragment this is not the case: we adopt the existing notion of *guarded clauses* [29] and we refer to [29] for a discussion of reduction to clausal form in the guarded fragment.

A clause C is *guarded*, if (i) for all non-ground compound subterms t of C, $\mathcal{V}ar(t) = \mathcal{V}ar(C)$, and (ii) if $\mathcal{V}ar(C) \neq \emptyset$, there exists a literal $L \in C^-$, called a *guard*, such that $\mathcal{V}ar(L) = \mathcal{V}ar(C)$ and every compound subterm of L is ground [29]. Although formulæ in the guarded fragment have no function symbols, guarded clauses may contain function symbols introduced by Skolemization. A *guarded set* is a set of guarded clauses. For example, $G_0 = \{\mathsf{R}(\mathsf{f}(\mathsf{a})),\ \neg\mathsf{P}(x) \vee \mathsf{R}(x) \vee \mathsf{Q}(\mathsf{f}(x))\}$ is a guarded set. Given G_0, SGGS with I^+ halts right away, and SGGS with I^- halts after placing $\mathsf{R}(\mathsf{f}(\mathsf{a}))$ on the trail. However, it is simple to give a guarded set where the termination of SGGS depends on the initial interpretation.

Example 10 Given the guarded set $G_1 = \{\mathsf{P}(\mathsf{a}),\ \neg\mathsf{P}(x) \vee \mathsf{P}(\mathsf{f}(x))\}$, SGGS with I^+ halts right away, but the SGGS-derivation with I^- is infinite:

$$\begin{aligned}\varepsilon \vdash\ & [\mathsf{P}(\mathsf{a})] && \mathsf{extend}\\ \vdash\ & [\mathsf{P}(\mathsf{a})],\ \neg\mathsf{P}(\mathsf{a}) \vee [\mathsf{P}(\mathsf{f}(\mathsf{a}))] && \mathsf{extend}\\ \vdash\ & [\mathsf{P}(\mathsf{a})],\ \neg\mathsf{P}(\mathsf{a}) \vee [\mathsf{P}(\mathsf{f}(\mathsf{a}))],\ \neg\mathsf{P}(\mathsf{f}(\mathsf{a})) \vee [\mathsf{P}(\mathsf{f}^2(\mathsf{a}))] && \mathsf{extend}\\ \vdash\ & \ldots\end{aligned}$$

In the next example the termination of SGGS depends on both initial interpretation and literal selection.

Example 11 Given the guarded set $G_2 = \{\neg\mathsf{P}(\mathsf{a}),\ \neg\mathsf{Q}(x) \vee \mathsf{P}(x) \vee \neg\mathsf{P}(\mathsf{f}(x))\}$, SGGS halts right away with I^-, but goes on forever with I^+:

$$\begin{aligned}
\varepsilon &\vdash [\neg\mathsf{P}(\mathsf{a})]\\
&\vdash [\neg\mathsf{P}(\mathsf{a})],\ \neg\mathsf{Q}(\mathsf{a}) \vee \mathsf{P}(\mathsf{a}) \vee [\neg\mathsf{P}(\mathsf{f}(\mathsf{a}))]\\
&\vdash [\neg\mathsf{P}(\mathsf{a})],\ \neg\mathsf{Q}(\mathsf{a}) \vee \mathsf{P}(\mathsf{a}) \vee [\neg\mathsf{P}(\mathsf{f}(\mathsf{a}))],\ \neg\mathsf{Q}(\mathsf{f}(\mathsf{a})) \vee \mathsf{P}(\mathsf{f}(\mathsf{a})) \vee [\neg\mathsf{P}(\mathsf{f}^2(\mathsf{a}))]\\
&\vdash \dots
\end{aligned}$$

where SGGS-extension is applied at every step. However, it suffices to select $\neg\mathsf{Q}(\mathsf{f}^n(\mathsf{a}))$ in place of $\neg\mathsf{P}(\mathsf{f}^{n+1}(\mathsf{a}))$, for some $n \geqslant 0$, that the derivation halts.

In the following example SGGS does not terminate regardless of literal selection and choice between I^- and I^+.

Example 12 The set $G_3 = \{\mathsf{P}(\mathsf{a}),\ \neg\mathsf{P}(x) \vee \mathsf{P}(\mathsf{f}(\mathsf{f}(x))),\ \neg\mathsf{P}(x) \vee \neg\mathsf{P}(\mathsf{f}(x))\}$ is guarded and is satisfied by the same finite model $\mathcal{I}$ and infinite Herbrand model J_1 given for S_1 from the previous section. With I^- SGGS generates an infinite derivation that lists as selected the positive literals of model J_1:

$$\begin{aligned}
\varepsilon &\vdash [\mathsf{P}(\mathsf{a})] && \mathsf{extend}\\
&\vdash [\mathsf{P}(\mathsf{a})],\ \neg\mathsf{P}(\mathsf{a}) \vee [\mathsf{P}(\mathsf{f}^2(\mathsf{a}))] && \mathsf{extend}\\
&\vdash [\mathsf{P}(\mathsf{a})],\ \neg\mathsf{P}(\mathsf{a}) \vee [\mathsf{P}(\mathsf{f}^2(\mathsf{a}))],\ \neg\mathsf{P}(\mathsf{f}^2(\mathsf{a})) \vee [\mathsf{P}(\mathsf{f}^4(\mathsf{a}))] && \mathsf{extend}\\
&\vdash \dots
\end{aligned}$$

The SGGS-derivation with I^+ is infinite even if literals of lower depth are preferred for selection, as the literals of model J_1 are listed as selected:

$$\begin{aligned}
\varepsilon &\vdash [\neg\mathsf{P}(x)] \vee \neg\mathsf{P}(\mathsf{f}(x)) && \mathsf{extend}\\
&\vdash [\neg\mathsf{P}(x)] \vee \neg\mathsf{P}(\mathsf{f}(x)),\ [\mathsf{P}(\mathsf{a})] && \mathsf{extend}\\
&\vdash [\neg\mathsf{P}(\mathsf{a})] \vee \neg\mathsf{P}(\mathsf{f}(\mathsf{a})),\ [\neg\mathsf{P}(\mathsf{f}(x))] \vee \neg\mathsf{P}(\mathsf{f}^2(x)),\ [\mathsf{P}(\mathsf{a})] && \mathsf{l\text{-}split}\\
&\vdash [\mathsf{P}(\mathsf{a})],\ [\neg\mathsf{P}(\mathsf{a})] \vee \neg\mathsf{P}(\mathsf{f}(\mathsf{a})),\ [\neg\mathsf{P}(\mathsf{f}(x))] \vee \neg\mathsf{P}(\mathsf{f}^2(x)) && \mathsf{move}\\
&\vdash [\mathsf{P}(\mathsf{a})],\ [\neg\mathsf{P}(\mathsf{f}(\mathsf{a}))],\ [\neg\mathsf{P}(\mathsf{f}(x))] \vee \neg\mathsf{P}(\mathsf{f}^2(x)) && \mathsf{resolve}\\
&\vdash [\mathsf{P}(\mathsf{a})],\ [\neg\mathsf{P}(\mathsf{f}(\mathsf{a}))],\ [\neg\mathsf{P}(\mathsf{f}(\mathsf{a}))] \vee \neg\mathsf{P}(\mathsf{f}^2(\mathsf{a})),\\
&\qquad [\neg\mathsf{P}(\mathsf{f}^2(x))] \vee \neg\mathsf{P}(\mathsf{f}^3(x)) && \mathsf{s\text{-}split}\\
&\vdash [\mathsf{P}(\mathsf{a})],\ [\neg\mathsf{P}(\mathsf{f}(\mathsf{a}))],\ [\neg\mathsf{P}(\mathsf{f}^2(x))] \vee \neg\mathsf{P}(\mathsf{f}^3(x)) && \mathsf{delete}\\
&\vdash [\mathsf{P}(\mathsf{a})],\ [\neg\mathsf{P}(\mathsf{f}(\mathsf{a}))],\ [\neg\mathsf{P}(\mathsf{f}^2(x))] \vee \neg\mathsf{P}(\mathsf{f}^3(x)),\ [\neg\mathsf{P}(\mathsf{a})] \vee \mathsf{P}(\mathsf{f}^2(\mathsf{a})) && \mathsf{extend}\\
&\vdash [\mathsf{P}(\mathsf{a})],\ [\neg\mathsf{P}(\mathsf{f}(\mathsf{a}))],\ [\neg\mathsf{P}(\mathsf{f}^2(x))] \vee \neg\mathsf{P}(\mathsf{f}^3(x)),\ [\mathsf{P}(\mathsf{f}^2(\mathsf{a}))] && \mathsf{resolve}\\
&\vdash [\mathsf{P}(\mathsf{a})],\ [\neg\mathsf{P}(\mathsf{f}(\mathsf{a}))],\ [\neg\mathsf{P}(\mathsf{f}^2(\mathsf{a}))] \vee \neg\mathsf{P}(\mathsf{f}^3(\mathsf{a})),\\
&\qquad [\neg\mathsf{P}(\mathsf{f}^3(x))] \vee \neg\mathsf{P}(\mathsf{f}^4(x)),\ [\mathsf{P}(\mathsf{f}^2(\mathsf{a}))] && \mathsf{l\text{-}split}\\
&\vdash [\mathsf{P}(\mathsf{a})],\ [\neg\mathsf{P}(\mathsf{f}(\mathsf{a}))],\ [\mathsf{P}(\mathsf{f}^2(\mathsf{a}))],\ [\neg\mathsf{P}(\mathsf{f}^2(\mathsf{a}))] \vee \neg\mathsf{P}(\mathsf{f}^3(\mathsf{a})),
\end{aligned}$$

$$\begin{array}{ll} \quad [\neg\mathsf{P}(\mathsf{f}^3(x))] \vee \neg\mathsf{P}(\mathsf{f}^4(x)) & \textsf{move} \\ \vdash [\mathsf{P}(\mathsf{a})],\ [\neg\mathsf{P}(\mathsf{f}(\mathsf{a}))],\ [\mathsf{P}(\mathsf{f}^2(\mathsf{a}))],\ [\neg\mathsf{P}(\mathsf{f}^3(\mathsf{a}))], & \\ \quad [\neg\mathsf{P}(\mathsf{f}^3(x))] \vee \neg\mathsf{P}(\mathsf{f}^4(x)) & \textsf{resolve} \\ \vdash \ldots & \end{array}$$

Resolution generates infinitely many clauses from these sets, while hyperresolution behaves similarly to SGGS. From G_0 both positive and negative hyperresolution generate nothing. From G_1 negative hyperresolution does not generate anything, whereas positive hyperresolution yields the infinite series $\{\mathsf{P}(\mathsf{f}^k(\mathsf{a})) : k \geqslant 1\}$. From G_2 positive hyperresolution does not generate anything, whereas negative hyperresolution yields the infinite series $\{\bigvee_{i=0}^{k} \neg\mathsf{Q}(\mathsf{f}^i(\mathsf{a})) \vee \neg\mathsf{P}(\mathsf{f}^{k+1}(\mathsf{a})) : k \geqslant 0\}$. From G_3 positive hyperresolution yields the infinite series $\{\mathsf{P}(\mathsf{f}^{2k}(a))\}_{k\geqslant 0}$, while negative hyperresolution generates $\neg\mathsf{P}(\mathsf{f}(\mathsf{a}))$ from nucleus $\mathsf{P}(\mathsf{a})$ and satellite $\neg\mathsf{P}(x) \vee \neg\mathsf{P}(\mathsf{f}(x))$, and then the infinite series $\{\neg\mathsf{P}(x) \vee \neg\mathsf{P}(\mathsf{f}^{2k+1}(x)) : k \geqslant 1\}$ from nucleus $\neg\mathsf{P}(x) \vee \mathsf{P}(\mathsf{f}(\mathsf{f}(x)))$ using $\neg\mathsf{P}(x) \vee \neg\mathsf{P}(\mathsf{f}(x))$ as initial satellite and then each resulting hyperresolvent as next satellite. Given an ordering $>$ on literals such that $\mathsf{P}(\mathsf{f}(x)) > \mathsf{P}(x)$ for G_1 and G_2, and such that $\mathsf{P}(\mathsf{f}(x)) > \mathsf{P}(x)$ and $\mathsf{P}(\mathsf{f}^2(x)) > \mathsf{P}(x)$ for G_3, ordered resolution halts right away.

4.5 SGGS Decides the Ground-Preserving Sets Decided by Hyperresolution

SGGS with I^- or I^+ as initial interpretation and hyperresolution share sign-based semantic guidance. We show that if the input clauses are positively ground-preserving, SGGS with I^- terminates whenever positive hyperresolution does. The result for SGGS with I^+ and negative hyperresolution is dual.

Definition 4 (Ground-Preserving) A clause C is *positively ground-preserving* if $\mathcal{V}ar(C) \subseteq \mathcal{V}ar(C^-)$, and *negatively ground-preserving* if $\mathcal{V}ar(C) \subseteq \mathcal{V}ar(C^+)$. A set of clauses is *positively/negatively ground-preserving* if all its clauses are, and *ground-preserving* if it is one or the other.

For example, $\neg\mathsf{P}(x, y, z) \vee \mathsf{Q}(y) \vee \mathsf{Q}(\mathsf{f}(z))$ and $\neg\mathsf{Q}(x) \vee \neg\mathsf{Q}(y)$ are positively ground-preserving. Datalog clauses are positively ground-preserving. The binary counter clauses of Example 5 are both positively and negatively ground-preserving. Guarded clauses are positively ground-preserving, since the guard is negative and contains all variables.

If a set S is positively ground-preserving, the positive clauses in S are ground, hence the initial satellites are ground, and positive hyperresolution generates only ground clauses, as at every step all variables in the nucleus get instantiated with ground terms by the simultaneous mgu with literals in ground satellites. The dual properties hold for the negative variant.

We begin by showing that SGGS also has this property, which implies that SGGS-splitting inferences do not apply and hence SGGS-constraints do not appear. Let I be *suitable for a ground-preserving set S* if either I is I^- and S is positively ground-preserving, or I is I^+ and S is negatively ground-preserving.

Lemma 2 *If the input set S is ground-preserving, all clauses on the trail of a fair SGGS-derivation with initial interpretation suitable for S are ground.*

Proof Assume that S is positively ground-preserving and I is I^- (the other case is dual). The proof is by induction on the stage k such that the step $\Gamma_k \vdash \Gamma_{k+1}$ adds clause C to the trail (i.e., C appears in Γ_{k+1}).
Base case: $k = 0$, and C is one of the I^--all-false (i.e., positive) input clauses added by the first SGGS-extension that yields Γ_1. (S must contain at least an I^--all-false clause, because otherwise it would be satisfied by I^- and the derivation would not even start.) Since S is ground-preserving, C is ground.
Induction hypothesis: for all j, $0 \leq j < k$, all clauses C added by the step $\Gamma_j \vdash \Gamma_{j+1}$ are ground.
Inductive case: let C be a clause added to the trail by the step $\Gamma_k \vdash \Gamma_{k+1}$. The only inferences that generate new clauses are SGGS-splitting, SGGS-resolution, and SGGS-extension. $\Gamma_k \vdash \Gamma_{k+1}$ cannot be a splitting step, because the splitting of a ground clause is trivial, hence excluded by fairness. If $\Gamma_k \vdash \Gamma_{k+1}$ is an SGGS-resolution step, a resolvent of ground clauses is also ground. If $\Gamma_k \vdash \Gamma_{k+1}$ is an SGGS-extension step, it adds an instance $C\alpha$ of a clause $C \in S$, where α is the simultaneous mgu of *all* I^--true (i.e., negative) literals $L_1, \ldots, L_n$ in C with as many I^--false (i.e., positive) selected literals $M_1, \ldots, M_n$ in clauses in Γ. By induction hypothesis, the clauses containing $M_1, \ldots, M_n$ are ground. Thus, $L_1\alpha, \ldots, L_n\alpha$ are also ground. The I^--false (i.e., positive) literals of $C\alpha$ are ground, because C is positively ground-preserving, so that all its variables appear in a negative literal and get grounded by α. Thus, $C\alpha$ is ground.

Let $Res^+_H(S)$ be the set of positive hyperresolvents generated from S; $R^0_H(S) = S$, $R^{k+1}_H(S) = R^k_H(S) \cup Res^+_H(R^k_H(S))$, and $R^*_H(S) = \bigcup_{k \geqslant 0} R^k_H(S)$. If S is positively ground-preserving, all clauses in $R^*_H(S) \setminus S$ are ground.

Lemma 3 *If the input set S is positively ground-preserving, for all fair SGGS-derivations with I^- as initial interpretation, for every clause C on the trail during the derivation, there exists a positive clause $C' \in R^*_H(S)$ such that $C^+ \subseteq C'$.*

Proof By Lemma 2 all clauses C that appear on the trail during the derivation are ground. The proof is by induction on the stage k such that the step $\Gamma_k \vdash \Gamma_{k+1}$ adds clause C to the trail (i.e., C appears in Γ_{k+1}).
Base case: $k = 0$, and C is one of the I^--all-false (i.e., positive) input clauses added by the first SGGS-extension that yields Γ_1. Then $C^+ = C$ and $C' = C \in S \subseteq R^*_H(S)$.
Induction hypothesis: for all j, $0 \leq j < k$, for all clauses C added by step $\Gamma_j \vdash \Gamma_{j+1}$ the claim holds.
Inductive case: let C be a clause added to the trail by step $\Gamma_k \vdash \Gamma_{k+1}$. By fairness, SGGS-splitting does not apply to ground clauses, and we only need to consider SGGS-resolution and SGGS-extension. If $\Gamma_k \vdash \Gamma_{k+1}$ is an SGGS-resolution step, the added clause is the SGGS-resolvent R generated from

(ground) parents $C_1[L]$ and $C_2[\neg L]$, where L is I^--false (i.e., positive) and $C_2[\neg L]$ is I^--all-true (i.e., negative), so that $R^+ \subseteq {C_1}^+$. By induction hypothesis there exists a positive clause $C' \in R^*_H(S)$ such that ${C_1}^+ \subseteq C'$, and hence $R^+ \subseteq C'$. If $\Gamma_k \vdash \Gamma_{k+1}$ is an SGGS-extension step with main premise $C \in S$ and side premises $D_1[M_1], \ldots, D_n[M_n]$ in Γ_k, the added clause is the instance $C\alpha$, for α the simultaneous mgu of all I^--true (i.e., negative) literals $L_1, \ldots, L_n$ in C with the I^--false (i.e., positive) selected literals $M_1, \ldots, M_n$. By induction hypothesis, for all i, $1 \leqslant i \leqslant n$, there exists a positive clause $\widehat{D}_i \in R^*_H(S)$ such that $D_i^+ \subseteq \widehat{D}_i$, so that M_i is a literal of $\widehat{D}_i$. Thus, positive hyperresolution applies to nucleus C and satellites $\widehat{D}_1, \ldots, \widehat{D}_n$ resolving upon all the negative literals $L_1, \ldots, L_n$ in C and the positive literals M_i in $\widehat{D}_i$ $(1 \leqslant i \leqslant n)$ with simultaneous mgu α. The generated positive hyperresolvent is $C' = (C^+ \vee \widehat{D}_1 \setminus \{M_1\} \vee \ldots \vee \widehat{D}_n \setminus \{M_n\})\alpha$. Since $C^+\alpha \subseteq C'$, the claim holds.

Given a set S of clauses, positive hyperresolution is guaranteed to halt if and only if $R^*_H(S)$ is finite. The next theorem shows that if positive hyperresolution is guaranteed to halt, so is SGGS.

Theorem 5 *If the input set S is positively ground-preserving and $R^*_H(S)$ is finite, all fair SGGS-derivations with I^- as initial interpretation are finite.*

Proof Since S is positively ground-preserving, all clauses in $R^*_H(S) \setminus S$ are ground, and all clauses on the trail during an SGGS-derivation are ground. We prove the claim by proving the contrapositive: if there exists an infinite SGGS-derivation Θ with initial interpretation I^- and input S, then $R^*_H(S)$ must be infinite. An SGGS-derivation can be infinite only if there are infinitely many SGGS-extension inferences, because the model fixing and conflict-solving activities of SGGS are inherently finite. Thus, Θ features infinitely many SGGS-extensions, adding ground clauses involving atoms of increasing depth. (If the depth of atoms were upper bounded, Θ would be in a finite basis and would be finite by Theorem 1.) Whenever an SGGS-extension adds a ground instance $C\alpha$ of a clause $C \in S$, the substitution α is the simultaneous mgu of all the negative literals in C with positive selected literals on the trail. Since S is finite, there are finitely many candidates for main premise. Therefore, infinitely many SGGS-extensions can occur only if there are infinitely many distinct sets of side premises in Θ involving atoms of increasing depth. Since the selected literals in the side premises are positive, this means that in the derivation Θ infinitely many distinct $C_j^+ \subseteq C_j$ appear on the trail. By Lemma 3, for all (ground) clauses C_j on the trail during Θ there exists a positive (ground) clause $C' \in R^*_H(S)$ such that $C_j^+ \subseteq C'$. Therefore, $R^*_H(S)$ must be infinite.

The dual variants of Lemma 3 and Theorem 5 hold for SGGS-derivations with I^+ and negative hyperresolution. Since the *positive variable dominated* (PVD) [34, 25] and *bounded depth increase* (BDI) [52] fragments are positively ground-preserving, and positive hyperresolution decides them, so does SGGS.[5]

[5] For PVD also the finite basis approach applies implying the small model property [24].

Corollary 1 *Given a PVD or BDI input set S, every fair SGGS-derivation with I^- as initial interpretation halts, is a refutation if S is unsatisfiable, and constructs a model of S if S is satisfiable.*

5 SGGS Decides Three New Fragments of First-Order Logic

In this section we introduce three new decidable fragments of first-order logic, by showing that SGGS decides them. The first one is the *restrained* fragment, which combines ground-preservigness with an ordering-based property. The other two are the *sort-restrained* fragment, which generalizes the stratified and restrained fragments, and the *sort-refined-PVD* fragment, which generalizes the stratified and PVD fragments.

5.1 SGGS Decides the Restrained Fragments

The following example gives the intuition for *restrained clauses.*

Example 13 Consider the set $S = \{(i)\ \mathsf{P}(\mathsf{s}^{10}(\mathsf{0}), \mathsf{s}^9(\mathsf{0})),\ (ii)\ \neg\mathsf{P}(\mathsf{s}(\mathsf{s}(x)), y) \vee \mathsf{P}(x, \mathsf{s}(y))\}$, which is positively ground-preserving. Let I^- be the initial interpretation. SGGS-extension puts $\mathsf{P}(10, 9)$ on the trail, abbreviating $\mathsf{s}^n(\mathsf{0})$ as n. This stirs a series of SGGS-extensions aiming at adding to $I[\Gamma]$ the positive ground literals needed to satisfy (ii) while satisfying (i). Each SGGS-extension unifies the negative literal in (ii) with a selected positive ground literal in Γ, so that new literals in added clauses are positive:

$$\begin{aligned}\Gamma_0 : \varepsilon \ &\vdash\ \Gamma_1 : [\mathsf{P}(10,9)]\\ &\vdash\ \Gamma_2 : [\mathsf{P}(10,9)],\ \neg\mathsf{P}(10,9) \vee [\mathsf{P}(8,10)]\\ &\vdash\ \Gamma_3 : [\mathsf{P}(10,9)],\ \neg\mathsf{P}(10,9) \vee [\mathsf{P}(8,10)],\ \neg\mathsf{P}(8,10) \vee [\mathsf{P}(6,11)].\end{aligned}$$

After adding $\neg\mathsf{P}(6,11) \vee [\mathsf{P}(4,12)]$, $\neg\mathsf{P}(4,12) \vee [\mathsf{P}(2,13)]$, and $\neg\mathsf{P}(2,13) \vee [\mathsf{P}(0,14)]$, SGGS halts with a model of S. The size of positive literals decreases as the derivation progresses, reflecting the fact that $\mathsf{P}(\mathsf{s}(\mathsf{s}(x)), y) \succ \mathsf{P}(x, \mathsf{s}(y))$ in clause (ii), for $\succ$ any KBO or any LPO with $\mathsf{P} \succ_p \mathsf{s}$ in the precedence.

This observation suggests to strengthen ground-preservingness with an ordering-based condition in order to get a finite basis.

Definition 5 (Restraining quasi-ordering) A quasi-ordering $\succeq$ on terms and atoms is *restraining*, if (i) it is stable, (ii) the strict ordering $\succ\ =\ \succeq \setminus \preceq$ is well-founded, and (iii) the equivalence $\approx\ =\ \succeq \cap \preceq$ has finite equivalence classes.

Condition (i) implies that $\succ$ and $\approx$ are also stable. Let $\succeq$ be a restraining quasi-ordering.

Definition 6 (Restrained) A clause C is *(strictly) positively restrained* if it is positively ground-preserving, and for all non-ground literals $L \in C^+$ there exists a literal $M \in C^-$ such that $at(M) \succeq at(L)$ ($at(M) \succ at(L)$). A set of clauses is *positively restrained* if all its clauses are.

Negatively restrained clauses and clause sets are defined dually, and a set of clauses is *restrained* if it is positively or negatively restrained. The set of Example 13 is strictly positively restrained. The next example clarifies why a *quasi*-ordering is used.

Example 14 Problem PLA030-1 in TPTP is neither stratified, nor monadic, nor guarded. It includes a clause $\mathsf{differ}(x, y) \vee \neg\mathsf{differ}(y, x)$ that cannot be *strictly* restrained. Let $\succ_{\mathsf{acrpo}}$ be an AC-compatible [73] RPO with differ as an AC-symbol, where AC abbreviates associative-commutative. The quasi-ordering $\succeq_{\mathsf{acrpo}}$, built from $\succ_{\mathsf{acrpo}}$ and the AC-equivalence $\approx_{\mathsf{AC}}$ that has finite equivalence classes, satisfies $\mathsf{differ}(x, y) \approx_{\mathsf{AC}} \mathsf{differ}(y, x)$ hence $\mathsf{differ}(x, y) \succeq_{\mathsf{acrpo}} \mathsf{differ}(y, x)$, so that PLA030-1 is negatively restrained.

Definition 7 (Basis for a restrained set) Given a restrained set S of clauses with Herbrand base $\mathcal{A}$, let $\mathcal{A}_S$ be the set of ground atoms occurring in S. Then the basis for S is $\mathcal{A}^{\preceq}_{S} = \{L : L \in \mathcal{A},\ \exists M \in \mathcal{A}_S \text{ such that } M \succeq L\}$.

In words, $\mathcal{A}^{\preceq}_{S}$ contains all the ground atoms upper bounded by those occurring in clauses in S. By Conditions (ii) and (iii) in Definition 5, $\mathcal{A}^{\preceq}_{S}$ is a finite basis. Since restrained sets are ground-preserving, the notion of suitable initial interpretation is the same as for ground-preserving sets.

Lemma 4 *If the input set S is restrained, every fair SGGS-derivation with suitable initial interpretation is in the finite basis $\mathcal{A}^{\preceq}_{S}$.*

Proof We consider S positively ground-preserving and I^- (for the dual case one exchanges the signs). Since the set is restrained hence ground-preserving, the derivation is ground by Lemma 2 (†). The proof is by induction on the length k of the derivation, and it follows the same pattern as that of Lemma 2. Let $\Gamma \vdash \Gamma'$ be the (k+1)-th step. By induction hypothesis, Γ is in $\mathcal{A}^{\preceq}_{S}$. If $\Gamma \vdash \Gamma'$ is an SGGS-resolution step, it is a ground resolution step which does not generate new atoms, and also Γ' is in $\mathcal{A}^{\preceq}_{S}$. If $\Gamma \vdash \Gamma'$ is an SGGS-extension step, it adds an instance $C\alpha$ of a clause $C \in S$, where α is the simultaneous mgu of all I^--true (i.e., negative) literals $\neg L_1, \ldots, \neg L_n$ in C with as many I^--false (i.e., positive) selected literals $M_1, \ldots, M_n$ in Γ. The literals $M_1, \ldots, M_n$ are ground by (†), and by induction hypothesis they are in $\mathcal{A}^{\preceq}_{S}$. We have to show that $at(C\alpha) \subseteq \mathcal{A}^{\preceq}_{S}$. For the negative literals $\neg L_1\alpha, \ldots, \neg L_n\alpha$ we have $L_i\alpha = M_i\alpha = M_i \in \mathcal{A}^{\preceq}_{S}$. Let L be a literal in C^+. If L is ground, then $L\alpha = L \in \mathcal{A}_S \subseteq \mathcal{A}^{\preceq}_{S}$. If L is not ground, by positive restrainedness there exists a $\neg L_i$, $1 \leqslant i \leqslant n$, such that $L_i \succeq L$. By stability, $L_i\alpha \succeq L\alpha$. Since for all i, $1 \leqslant i \leqslant n$, $M_i \in \mathcal{A}^{\preceq}_{S}$ and $M_i = M_i\alpha = L_i\alpha \succeq L\alpha$, we have $L\alpha \in \mathcal{A}^{\preceq}_{S}$.

Therefore, Theorems 1 and 2 yield decidability and the small model property.

Theorem 6 *Given a restrained input set S, every fair SGGS-derivation with suitable initial interpretation halts, is a refutation if S is unsatisfiable, and constructs a model of S if S is satisfiable.*

Corollary 2 *A restrained satisfiable set S of clauses has a model of cardinality $|\mathcal{H}(\mathcal{A}_S^{\preceq})| + 1$ that can be extracted from the limit of any fair SGGS-derivation with input S and suitable initial interpretation.*

Example 15 The clause set of Example 13 is a subset of the following satisfiable clause set S from problem PUZ054-1 in TPTP:

$$\begin{array}{lll} \mathsf{P}(\mathsf{s}^{10}(\mathsf{0}), \mathsf{s}^{9}(\mathsf{0})), & \neg\mathsf{P}(\mathsf{s}(\mathsf{s}(x)), y) \vee \mathsf{P}(x, \mathsf{s}(y)), & \neg\mathsf{P}(x, \mathsf{s}(\mathsf{s}(y))) \vee \mathsf{P}(x, \mathsf{s}(y)), \\ \neg\mathsf{P}(\mathsf{s}(\mathsf{0}), \mathsf{0}), & \neg\mathsf{P}(\mathsf{s}(x), \mathsf{s}(y)) \vee \mathsf{P}(\mathsf{s}(x), y). & \end{array}$$

This set, which is neither EPR nor FO^2 nor monadic, can be shown strictly positively restrained by any LPO with $\mathsf{P} \succ_p \mathsf{s}$ in the precedence or by any KBO. Let $\succ$ be a KBO with empty precedence, $w(\mathsf{P}) = 0$, and $w(\mathsf{s}) = w(\mathsf{0}) = w_0 = 1$. $\mathcal{A}_S$ is $\{\mathsf{P}(\mathsf{s}^{10}(\mathsf{0}), \mathsf{s}^{9}(\mathsf{0})),\ \mathsf{P}(\mathsf{s}(\mathsf{0}), \mathsf{0})\}$ and its largest atom has weight $w(\mathsf{P}(\mathsf{s}^{10}(\mathsf{0}), \mathsf{s}^{9}(\mathsf{0}))) = 21$. $\mathcal{A}_S^{\preceq}$ cannot contain an atom $L = \mathsf{P}(\mathsf{s}^{n}(\mathsf{0}), \mathsf{s}^{m}(\mathsf{0}))$, with $n \geqslant 0$ and $m \geqslant 0$, if $n > 19$ or $m > 19$, because otherwise $w(L) > w(\mathsf{P}(\mathsf{s}^{10}(\mathsf{0}), \mathsf{s}^{9}(\mathsf{0})))$. Therefore, $\mathcal{H}(\mathcal{A}_S^{\preceq}) = \{\mathsf{s}^{i}(\mathsf{0}) : 0 \leqslant i \leqslant 19\}$ and S has a model of cardinality 21 by Corollary 2.

Sign-based semantic guidance makes SGGS well suited for the restrained fragments. We see next that this holds also for sign-based resolution strategies.

5.2 Sign-Based Resolution Strategies Decide the Restrained Fragments

We consider PO-resolution and the positively restrained fragment. The results will then be extended to other positive strategies and to the dual case. Let $Res_>^+(S)$ be the set of PO-resolvents generated from clauses in S, where $>$ is the CSO on literals assumed by PO-resolution. Then, $R_>^0(S) = S$, $R_>^{k+1}(S) = R_>^k(S) \cup Res_>^+(R_>^k(S))$, and $R_>^*(S) = \bigcup_{k \geqslant 0} R_>^k(S)$.

Lemma 5 *If S is positively restrained, then for all $C \in R_>^*(S)$, for all $L \in C^+$ either $L \in \mathcal{A}_S^{\preceq}$ or $at(M) \succeq at(L)$ for some $M \in C^-$.*

Proof The proof is by induction on the stage k of the construction of $R_>^*(S)$. For $k{=}0$, the clauses in $R_>^0(S) = S$ satisfy the claim by Definitions 6 and 7. The induction hypothesis is that all clauses in $R_>^k(S)$ satisfy the claim. For the inductive case, let $(C \vee D)\sigma \in Res_>^+(R_>^k(S))$ be a PO-resolvent with mgu σ from parents $\neg L \vee C$ and $L' \vee D$ in $R_>^k(S)$, where $L' \vee D$ is a positive clause. By induction hypothesis $at(L' \vee D) \subseteq \mathcal{A}_S^{\preceq}$ (†), which means $L' \vee D$ is ground, $(L' \vee D)\sigma = L' \vee D$, and $at(D\sigma) \subseteq \mathcal{A}_S^{\preceq}$. For the positive literals in $C\sigma$, let $Q\sigma$ be one of them, so $Q \in C^+$. By induction hypothesis, either (i) $Q \in \mathcal{A}_S^{\preceq}$, or (ii) $M \succeq Q$ for some negative literal $\neg M$ in $\neg L \vee C$. In case (i),

Q is ground, $Q\sigma = Q$, and $Q\sigma \in \mathcal{A}_S^{\preceq}$. In case (ii), if $\neg M$ is one of the literals in C, then $\neg M\sigma \in C\sigma$, and $M\sigma \succeq Q\sigma$ holds by stability, so that the claim follows. Otherwise, $\neg M$ is the resolved-upon literal $\neg L$ with $L\sigma = L'\sigma$. Thus, $L = M \succeq Q$, which implies $L\sigma \succeq Q\sigma$ by stability. By (†), $L' \in \mathcal{A}_S^{\preceq}$, L' is ground, and $L'\sigma = L'$. Since $L' \in \mathcal{A}_S^{\preceq}$ and $L' = L'\sigma = L\sigma \succeq Q\sigma$, it follows that $Q\sigma \in \mathcal{A}_S^{\preceq}$ by Definition 7.

Thus, a positive clause $C \in R_>^*(S)$ is ground, as all its literals are in $\mathcal{A}_S^{\preceq}$.

Theorem 7 *Given a positively restrained input set S, every fair PO-resolution derivation terminates and is a refutation if S is unsatisfiable.*

Proof We prove that if S is positively restrained then $R_>^*(S)$ is finite, which guarantees termination. The second part of the claim follows by refutational completeness of PO-resolution [42]. Consider any PO-resolvent $(C \vee D)\sigma \in R_>^*(S)$ from parents $\neg L \vee C$ and $L' \vee D$ with mgu σ. Since $L' \vee D$ is positive, $(C \vee D)\sigma$ has strictly fewer negative literals than $\neg L \vee C$. By way of contradiction, suppose that $R_>^*(S)$ is infinite. Since the number of negative literals in PO-resolvents decreases at every resolution step, an infinite $R_>^*(S)$ must contain infinitely many positive clauses. By Lemma 5, all positive clauses in $R_>^*(S)$ are ground clauses made of atoms from $\mathcal{A}_S^{\preceq}$. Since $\mathcal{A}_S^{\preceq}$ is finite, and repeated literals in ground clauses disappear by merging, only finitely many clauses can be built from $\mathcal{A}_S^{\preceq}$, which contradicts $R_>^*(S)$ being infinite.

These results[6] extend to positive resolution, since the $>$-maximality of $L'\sigma$ in $(L' \vee D)\sigma$ is not used in the proofs, and to positive hyperresolution, for which the proof of Theorem 7 is trivial, since only positive clauses get generated.

Corollary 3 *PO-resolution, positive hyperresolution, and positive resolution decide the positively restrained fragment.*

Thus, Theorem 6 follows also from Theorem 5 and Corollary 3. Dually, negative resolution and negative hyperresolution decide the negatively restrained fragment. The next example shows that SGGS can be *exponentially more efficient* than saturation-based resolution strategies because it is model-based.

Example 16 Consider the following parametric clause set S_n consisting of $n+1$ clauses, using $i+1$-ary predicates P_i and constants c_i, for all i, $0 \leqslant i \leqslant n$:

$$\begin{array}{ll}
\mathsf{P}_0(\mathsf{c}_0) \vee \mathsf{P}_0(\mathsf{c}_1) \vee \cdots \vee \mathsf{P}_0(\mathsf{c}_n) & (C_0),\\
\neg\mathsf{P}_0(x_1) \vee \mathsf{P}_1(x_1, \mathsf{c}_0) \vee \mathsf{P}_1(x_1, \mathsf{c}_1) \vee \cdots \vee \mathsf{P}_1(x_1, \mathsf{c}_n) & (C_1),\\
\neg\mathsf{P}_1(x_1, x_2) \vee \mathsf{P}_2(x_1, x_2, \mathsf{c}_0) \vee \cdots \vee \mathsf{P}_2(x_1, x_2, \mathsf{c}_n) & (C_2),\\
\ldots & \ldots\\
\neg\mathsf{P}_{n-1}(x_1, \ldots, x_n) \vee \mathsf{P}_n(x_1, \ldots, x_n, \mathsf{c}_0) \vee \cdots \vee \mathsf{P}_n(x_1, \ldots, x_n, \mathsf{c}_n) & (C_n).
\end{array}$$

[6] Lemma 5 and Theorem 7 were proved for ordered resolution assuming $>$ ensures that $L' \vee D$ is positive [24, Lem. 5 and Thm. 6]; it is better to work with PO-resolution.

The set S_n is positively restrained by an LPO with precedence $\mathsf{P}_0 > \cdots > \mathsf{P}_n > \mathsf{c}_i$ for all i, $0 \leqslant i \leqslant n$. SGGS with I^- detects satisfiability after $n+1$ SGGS-extension steps, selecting for instance the leftmost positive literal in each extension clause, so that the model where $\mathsf{P}_0(\mathsf{c}_0), \mathsf{P}_1(\mathsf{c}_0, \mathsf{c}_0), \ldots, \mathsf{P}_n(\mathsf{c}_0, \ldots, \mathsf{c}_0)$ are true and all other positive literals are false is produced. A saturation by PO-resolution or positive hyperresolution produces exponentially many clauses, because for all i, $0 \leqslant i \leqslant n$, all n positive literals in C_i unify with the negative literal in C_{i+1}, generating n^{i+1} positive clauses, so that the total clause count is given by $\sum_{i=0}^{n} n^{i+1}$ or equivalently $\sum_{k=1}^{n+1} n^k$.

5.3 Sort-Refined Versions of the Restrained and PVD Fragments

As observed for the stratified fragment, where all sorts are acyclic, such sorts are harmless for termination. In this section we consider a signature with both cyclic and acyclic sorts. Since the key point for termination is the existence of a finite basis, we reason in terms of whether there are finitely or infinitely many ground terms of a given sort.

Definition 8 (Infinite domain) A sort has *infinite domain* if there are infinitely many ground terms of that sort, and it has *finite domain* otherwise. A variable has infinite domain if its sort does, and finite otherwise.

Clearly, a cyclic sort has infinite domain. For example, if the signature contains a constant $\mathsf{a}\colon s$ and a function $\mathsf{f}\colon s \to s$, sort s has infinite domain, as the infinitely many terms $\mathsf{f}^n(\mathsf{a})$, for all $n \geq 0$, have sort s. If the signature also contains a function $\mathsf{g}\colon s \to s'$, also s' has infinite domain, as the infinitely many terms $\mathsf{g}(f^n(\mathsf{a}))$, for all $n \geq 0$, have sort s'. In general, a sort s has infinite domain if and only if there exists a path from a cyclic sort to s in the sort dependency graph. A term, or atom, or literal has infinitely many ground instances if and only if it contains a variable with infinite domain. The idea is to apply the restrictions of the restrained, or PVD [34], fragments, respectively, only to the variables of infinite domain and the literals where such variables occur. The result will be the *sort-restrained* and *sort-refined-PVD* fragments. We begin by making ground-preservingness relative to a sort:

Definition 9 (Ground-preserving for a sort) A clause C is *positively ground-preserving for sort s* if $\mathcal{V}ar_s(C) \subseteq \mathcal{V}ar_s(C^-)$, and *negatively ground-preserving for sort s* if $\mathcal{V}ar_s(C) \subseteq \mathcal{V}ar_s(C^+)$. A set of clauses is *positively/negatively ground-preserving for sort s* if all its clauses are.

Both sort-restrained and sort-refined-PVD fragments will require that clauses are ground-preserving for all sorts of infinite domain.

5.3.1 SGGS Decides the Sort-Restrained Fragments

The next example gives the intuition for the sort-restrained fragment.

Example 17 Consider the following set S of clauses with sorts $\{s_1, s_2\}$:

$$\mathsf{P}(x, \mathsf{f}(\mathsf{b}))\ \ (i) \quad \neg\mathsf{Q}(x, \mathsf{a}) \vee \mathsf{Q}(\mathsf{a}, x)\ \ (ii) \quad \neg\mathsf{P}(x, \mathsf{f}(y)) \vee \mathsf{Q}(x, x) \vee \mathsf{P}(x, y)\ \ (iii)$$

where $\mathsf{a}\colon s_1$, $\mathsf{b}\colon s_2$, $\mathsf{f}\colon s_2 \to s_2$, $\mathsf{P} \subseteq s_1 \times s_2$, and $\mathsf{Q} \subseteq s_1 \times s_1$. This clause set is not restrained because it is not ground-preserving since the positive clause (i) is not ground, and it is not stratified because function symbol f induces a cycle over sort s_2. However, S is positively ground-preserving for sort s_2: there are no variables of sort s_2 in positive clause (i), and the only variable of sort s_2 in a mixed clause, namely y in (iii), occurs in negative literal $\neg\mathsf{P}(x, \mathsf{f}(y))$. Moreover, $\mathsf{P}(x, y)$ in clause (iii) is dominated by $\neg\mathsf{P}(x, \mathsf{f}(y))$ in the sense of positive restrainedness, since $\mathsf{P}(x, \mathsf{f}(y)) \succ \mathsf{P}(x, y)$ for $\succ$ any LPO or KBO. Indeed, SGGS using I^- terminates on input S:

$$\begin{aligned} \varepsilon &\vdash [\mathsf{P}(x, \mathsf{f}(\mathsf{b}))] && \mathsf{extend}\ (i) \\ &\vdash [\mathsf{P}(x, \mathsf{f}(\mathsf{b}))],\ \neg\mathsf{P}(x, \mathsf{f}(\mathsf{b})) \vee \mathsf{Q}(x, x) \vee [\mathsf{P}(x, \mathsf{b})] && \mathsf{extend}\ (iii) \end{aligned}$$

In the second extension clause either positive literal can be selected with either choice leading to termination.

Let $\succeq$ be a restraining quasi-ordering with the subterm property.

Definition 10 (Sort-Restrained) A clause C is *positively sort-restrained* if it is positively ground-preserving for all sorts with infinite domain, and for all literals $L \in C^+$ such that $Gr(L)$ is infinite there exists a literal $M \in C^-$ such that $at(M) \succeq at(L)$. A set is positively sort-restrained if all its clauses are.

Negatively sort-restrained clauses and clause sets are defined dually, and a set of clauses is *sort-restrained* if it is positively or negatively sort-restrained. The set of Example 17 is positively sort-restrained.

Let a set of atoms $\mathcal{L}$ be (i) *closed with respect to instantiation*, or *instantiation-closed* for short, if $L\sigma \in \mathcal{L}$ whenever $L \in \mathcal{L}$; and (ii) *closed under* $\preceq$, or $\preceq$*-closed* for short, if $M \in \mathcal{L}$ whenever $M \preceq L$ for some $L \in \mathcal{L}$.

Definition 11 (Basis for a sort-restrained set) Given a sort-restrained set S of clauses with set of sorts Σ, let $\mathcal{L}^{\downarrow}_{\Sigma}$ be the set of all atoms L such that L occurs in a clause of S and $Gr(L)$ is finite. Let $\mathcal{L}^{\preceq}_{\Sigma}$ be the smallest instantiation-closed and $\preceq$-closed superset of $\mathcal{L}^{\downarrow}_{\Sigma}$. The basis for S is $\mathcal{A}^{\preceq}_{S,\Sigma} = Gr(\mathcal{L}^{\preceq}_{\Sigma})$, that is, the set of the ground instances of the atoms in $\mathcal{L}^{\preceq}_{\Sigma}$.

Note that $\mathcal{A}^{\preceq}_{S,\Sigma} \subseteq \mathcal{L}^{\preceq}_{\Sigma}$ because $\mathcal{L}^{\preceq}_{\Sigma}$ is instantiation-closed.

Example 18 For the clause set of Example 17 let $\succeq$ be the reflexive closure of an LPO with empty precedence. $\mathcal{L}^{\downarrow}_{\Sigma}$ is $\{\mathsf{P}(x, \mathsf{f}(\mathsf{b})), \mathsf{Q}(x, \mathsf{a}), \mathsf{Q}(\mathsf{a}, x), \mathsf{Q}(x, x)\}$. $\mathcal{L}^{\preceq}_{\Sigma}$ is the union of four sets: $\mathcal{L}^{\downarrow}_{\Sigma}$, the singleton set $\{\mathsf{P}(x, \mathsf{b})\}$ by $\preceq$-closure since $\mathsf{P}(x, \mathsf{b}) \prec \mathsf{P}(x, \mathsf{f}(\mathsf{b}))$, the set $\{\mathsf{P}(\mathsf{a}, \mathsf{f}(\mathsf{b})), \mathsf{Q}(\mathsf{a}, \mathsf{a}), \mathsf{P}(\mathsf{a}, \mathsf{b})\}$ by instantiation-closure, and the set of all the variants of these atoms (i.e., all the variants of $\mathsf{P}(x, \mathsf{f}(\mathsf{b}))$, $\mathsf{Q}(x, \mathsf{a})$, $\mathsf{Q}(\mathsf{a}, x)$, $\mathsf{Q}(x, x)$, and $\mathsf{P}(x, \mathsf{b})$). Then $\mathcal{A}^{\preceq}_{S,\Sigma}$ is the set of the ground instances of the atoms in $\mathcal{L}^{\preceq}_{\Sigma}$, that is, $\{\mathsf{P}(\mathsf{a}, \mathsf{f}(\mathsf{b})),\ \mathsf{Q}(\mathsf{a}, \mathsf{a}),\ \mathsf{P}(\mathsf{a}, \mathsf{b})\}$.

The above definition of closure lets instantiation introduce variables, but this is not a problem for the finiteness of $\mathcal{A}^{\preceq}_{S,\Sigma}$ for the following reason: a substitution replaces a variable with finite domain by a term of the same sort, hence with finite domain, and such a term cannot contain a variable with infinite domain, because a term of a sort with finite domain cannot have a subterm of a sort with infinite domain.

Lemma 6 *For all sort-restrained sets S of clauses, the basis $\mathcal{A}^{\preceq}_{S,\Sigma}$ is finite.*

Proof In order to show that $\mathcal{A}^{\preceq}_{S,\Sigma} = Gr(\mathcal{L}^{\preceq}_{\Sigma})$ is finite, it suffices to show that all variables occurring in atoms in $\mathcal{L}^{\preceq}_{\Sigma}$ have finite domain. The set $\mathcal{L}^{\downarrow}_{\Sigma}$ satisfies this property by definition. The closure with respect to instantiation introduces no variable with infinite domain by the above observation. The $\preceq$-closure does not introduce variables, because $L \succeq M$ implies $\mathcal{V}ar(M) \subseteq \mathcal{V}ar(L)$ by the stability and subterm properties of the restraining quasi-ordering.

Similar to the ground-preserving and restrained cases, an initial interpretation I is *suitable for a sort-restrained set S* if either I is I^- and S is positively sort-restrained, or I is I^+ and S is negatively sort-restrained.

Lemma 7 *Given a sort-restrained input set S, every fair SGGS-derivation with suitable initial interpretation is in $\mathcal{A}^{\preceq}_{S,\Sigma}$.*

Proof We consider S positively sort-restrained and I^- (for the dual case one flips the signs). We show that the derivation is in $\mathcal{A}^{\preceq}_{S,\Sigma} = Gr(\mathcal{L}^{\preceq}_{\Sigma})$ by showing that all atoms of all clauses appearing on the trail during the derivation are in $\mathcal{L}^{\preceq}_{\Sigma}$. The proof is by induction on the length k of Θ. The base case ($k = 0$) is vacuously true. The induction hypothesis is that all atoms of all clauses on a trail Γ produced by a derivation of length k are in $\mathcal{L}^{\preceq}_{\Sigma}$. Let $\Gamma \vdash \Gamma'$ be the (k+1)-th step. If $\Gamma \vdash \Gamma'$ is a splitting step, the atoms in the split clause are in $\mathcal{L}^{\preceq}_{\Sigma}$ by induction hypothesis, and so are those in the instances generated by splitting, since $\mathcal{L}^{\preceq}_{\Sigma}$ is closed with respect to instantiation. If $\Gamma \vdash \Gamma'$ is an SGGS-resolution step, the atoms in the parents are in $\mathcal{L}^{\preceq}_{\Sigma}$ by induction hypothesis, and so are those in the SGGS-resolvent, by the closure of $\mathcal{L}^{\preceq}_{\Sigma}$ with respect to instantiation. If $\Gamma \vdash \Gamma'$ is an SGGS-extension step, it adds an instance $C\alpha$ of a clause $C \in S$, where α is the simultaneous mgu of all I^--true (i.e., negative) literals $\neg L_1, \ldots, \neg L_n$ in C with as many I^--false (i.e., positive) selected literals $M_1, \ldots, M_n$ in Γ. For all i, $1 \leqslant i \leqslant n$, $M_i \in \mathcal{L}^{\preceq}_{\Sigma}$ by induction hypothesis, and $L_i\alpha = M_i\alpha \in \mathcal{L}^{\preceq}_{\Sigma}$ by instantiation closure. For an $L \in C^+$, there are two cases. If $Gr(L)$ is finite, $L \in \mathcal{L}^{\downarrow}_{\Sigma} \subseteq \mathcal{L}^{\preceq}_{\Sigma}$ by Definition 11, and $L\alpha \in \mathcal{L}^{\preceq}_{\Sigma}$ by instantiation closure. Otherwise, by positive sort-restrainedness there exists a $\neg L_i$, for some i, $1 \leqslant i \leqslant n$, such that $L \preceq L_i$. By stability of $\preceq$, $L\alpha \preceq L_i\alpha$. It follows that $L\alpha \preceq L_i\alpha = M_i\alpha \in \mathcal{L}^{\preceq}_{\Sigma}$ because $\mathcal{L}^{\preceq}_{\Sigma}$ is $\preceq$-closed.

Since the basis $\mathcal{A}^{\preceq}_{S,\Sigma}$ is finite, decidability and the small model property follow by Theorems 1 and 2.

Theorem 8 *Given a sort-restrained input set S, every fair SGGS-derivation with suitable initial interpretation halts, is a refutation if S is unsatisfiable, and constructs a model of S if S is satisfiable.*

Corollary 4 *A sort-restrained satisfiable set S of clauses has a model of cardinality at most $|\mathcal{H}(\mathcal{A}^{\preceq}_{S,\Sigma})| + 1$ that can be extracted from the limit of any fair SGGS-derivation with input S and suitable initial interpretation.*

While sign-based resolution strategies decide the restrained fragments (cf. Section 5.2), they do not decide the sort-restrained ones, because they do not decide the stratified fragment (cf. Example 3), which is contained in each sort-restrained fragment as the special case where all sorts have finite domain.

5.3.2 SGGS Decides the Sort-Refined-PVD Class

We recall the PVD property [34,25] in order to apply it to the variables of infinite domain. For a clause C, let $\mathrm{depth}_x(C)$ be the maximum occurrence depth in C of a variable $x \in \mathcal{V}ar(C)$.

Definition 12 (PVD fragment) A set S of clauses is in the PVD fragment if every clause $C \in S$ is positively ground-preserving and $\forall x \in \mathcal{V}ar(C^+)$ it holds that $\mathrm{depth}_x(C^+) \leqslant \mathrm{depth}_x(C^-)$.

The next example captures the intuition for the sort-refined-PVD fragment.

Example 19 Assume a signature with sorts $\{s_1, s_2, s_3, s_4\}$ and symbols $\mathsf{a}\colon s_1$, $\mathsf{b}\colon s_2$, $\mathsf{c}\colon s_3$, $\mathsf{f}\colon s_1 \to s_1$, $\mathsf{g}\colon s_3 \to s_2$, $\mathsf{h}\colon s_1 \times s_2 \to s_4$, $\mathsf{P} \subseteq s_1 \times s_2$, $\mathsf{Q} \subseteq s_4$, and $\mathsf{R} \subseteq s_1 \times s_1$, so that the sort-dependency graph is as follows:

$$\begin{tikzcd} s_1 \arrow[loop left] \arrow[r] & s_4 & s_2 \arrow[l] & s_3 \arrow[l] \end{tikzcd}$$

Sort s_1 is cyclic, and both s_1 and s_4 have infinite domain, while s_2 and s_3 have finite domain. Consider the set S made of the following clauses:

$$\begin{array}{llll} \mathsf{P}(\mathsf{f}(\mathsf{a}), y) & (i) & \neg\mathsf{P}(x,y) \vee \mathsf{P}(x, \mathsf{g}(z)) & (ii) \\ \neg\mathsf{P}(\mathsf{f}(x), y) \vee \mathsf{Q}(\mathsf{h}(x,y)) & (iii) & \neg\mathsf{P}(x,z) \vee \neg\mathsf{P}(y,z) \vee \mathsf{R}(x,y) & (iv) \end{array}$$

This set is neither stratified nor ground-preserving, hence neither restrained nor PVD. Neither it is sort-restrained, because the positive literal $\mathsf{R}(x,y)$ in clause *(iv)* involves sort s_1, but no negative literal in *(iv)* can dominate $\mathsf{R}(x,y)$ in a restraining quasi-ordering, since no negative literal in *(iv)* contains both x and y. However, S is positively ground-preserving for s_1 and s_4, and all variables of sorts with infinite domain that occur in a positive literal also occur in a negative literal of the same clause. Furthermore, such variables occur in the negative literals at greater or equal depth. In other words, S satisfies the PVD property restricted to sorts with infinite domain.

Definition 13 (Sort-Refined-PVD) A clause C is *sort-refined-PVD* if it is positively ground-preserving for all sorts with infinite domain, and for all variables $x \in \mathcal{V}ar(C^+)$ of infinite domain it holds that $\mathrm{depth}_x(C^+) \leqslant \mathrm{depth}_x(C^-)$. A set of clauses is *sort-refined-PVD* if all its clauses are.

The set of Example 19 is sort-refined-PVD. We apply the finite basis approach to show that SGGS decides also this fragment. While the essence of PVD is to control the depth of variable occurrences, for sort-refined-PVD the crux is to exclude variables of infinite domain and to ensure that the occurrence depth of terms whose sort has infinite domain is upper bounded. Let d be the maximum depth of an atom in a set S of clauses, or $d = max\{\mathrm{depth}(L) : \ L \text{ is an atom in clause } C \text{ and } C \in S\}$.

Definition 14 (Basis for a sort-refined-PVD set) Given a sort-refined-PVD set S of clauses with set of sorts Σ, let $\mathcal{L}^d_{S,\Sigma}$ be the set of all atoms where all variables have finite domain and all subterms of a sort with infinite domain have occurrence depth at most d. Then the basis for S is $\mathcal{A}^d_{S,\Sigma} = Gr(\mathcal{L}^d_{S,\Sigma})$.

Note that $\mathcal{L}^d_{S,\Sigma}$ is instantiation-closed, because instantiation replaces variables with finite domain with terms whose sort has finite domain, so that no subterm whose sort has infinite domain can be introduced. It follows that $\mathcal{A}^d_{S,\Sigma} \subseteq \mathcal{L}^d_{S,\Sigma}$.

Lemma 8 *For all sort-refined-PVD sets S of clauses, the basis $\mathcal{A}^d_{S,\Sigma}$ is finite.*

Proof We show that the depth of any ground atom $L\sigma \in \mathcal{A}^d_{S,\Sigma}$ for $L \in \mathcal{L}^d_{S,\Sigma}$ is upper bounded. The occurrence depth of any subterm of $L\sigma$ whose sort has finite domain is trivially upper bounded. By Definition 14 the occurrence depth of any subterm of L whose sort has infinite domain is upper bounded by d, and L does not contain variables of infinite domain. Thus, the substitution σ cannot introduce subterms of infinite domain and also the occurrence depth of any subterm of $L\sigma$ whose sort has infinite domain is upper bounded by d. As there are only finitely many ground atoms of bounded depth, $\mathcal{A}^d_{S,\Sigma}$ is finite.

Example 20 In Example 19 the maximum depth of an atom in S is $d = 2$. Thus, $\mathcal{A}^d_{S,\Sigma}$ is the set of all ground atoms where all subterms of sort s_1 or s_4 occur at depth at most 2:

$$\mathsf{P(a,b)},\ \mathsf{P(f(a),b)},\ \mathsf{P(a,g(c))},\ \mathsf{P(f(a),g(c))},\ \mathsf{Q(h(a,b))},$$
$$\mathsf{Q(h(a,g(c)))},\ \mathsf{R(a,a)},\ \mathsf{R(f(a),a)},\ \mathsf{R(a,f(a))},\ \mathsf{R(f(a),f(a))}$$

For instance, $\mathsf{Q(h(f(a),b))} \notin \mathcal{A}^d_{S,\Sigma}$ as the subterm a occurs at depth 3.

Lemma 9 *Given a sort-refined-PVD input set S, every fair SGGS-derivation with I^- is in the finite basis $\mathcal{A}^d_{S,\Sigma}$.*

Proof We show that the derivation is in $\mathcal{A}^d_{S,\Sigma} = Gr(\mathcal{L}^d_{S,\Sigma})$ by showing that all atoms of all clauses appearing on the trail during the derivation are in $\mathcal{L}^d_{S,\Sigma}$. As $\mathcal{L}^d_{S,\Sigma}$ is instantiation-closed, the proof is the same as that of Lemma 7 except for the case of SGGS-extension. Consider an SGGS-extension step that adds an instance $C\alpha$ of a clause $C \in S$, where α is the simultaneous mgu of all I^--true (i.e., negative) literals $\neg L_1, \dots, \neg L_n$ in C with as many I^--false (i.e., positive) selected literals $M_1, \dots, M_n$ in Γ. For all i, $1 \leqslant i \leqslant n$, $M_i \in \mathcal{L}^d_{S,\Sigma}$ by induction hypothesis, and $L_i\alpha = M_i\alpha \in \mathcal{L}^d_{S,\Sigma}$ by instantiation closure. For an $L \in C^+$, let t be a subterm of $L\alpha$ at position p (i.e., $t = L\alpha|_p$) whose sort has infinite domain. We show that for all such terms t, it holds that t is not a variable and that $|p| \leqslant d$. We distinguish two cases depending on whether p is a position in L or is introduced by α.

- If p is a position in L then $|p| \leqslant d$ because $L \in C$ and $C \in S$. For the other part, by way of contradiction, suppose that t is a variable. Then also $L|_p$ must be a variable x of infinite domain such that $x\alpha = t$. By Definition 13, x occurs in some L_i, $1 \leqslant i \leqslant n$, so $x\alpha$ occurs in $L_i\alpha = M_i\alpha$. This gives a contradiction, because $M_i \in \mathcal{L}^d_{S,\Sigma}$, $M_i\alpha \in \mathcal{L}^d_{S,\Sigma}$, and hence $x\alpha = t$ cannot be a variable of infinite domain by Definition 14.
- If p is introduced by α, there must be two positions q and r and a variable y such that $p = qr$, $L|_q = y$, $y\alpha|_r = t$, and also y must have infinite domain. By Definition 13, variable y occurs in some L_i, $1 \leqslant i \leqslant n$, and $\mathrm{depth}_y(L) \leqslant \mathrm{depth}_y(L_i)$. Hence there is some position o in L_i such that $L_i|_o = y$ and $|q| \leqslant |o|$. It follows that $L_i\alpha|_{or} = M_i\alpha|_{or} = t$. Since $M_i \in \mathcal{L}^d_{S,\Sigma}$ and $M_i\alpha \in \mathcal{L}^d_{S,\Sigma}$, term t cannot be a variable. Moreover, by Definition 14 terms of sort with infinite domain occur at depth at most d, so that $|or| \leqslant d$. From $|q| \leqslant |o|$ it follows that $|p| = |qr| \leqslant |or| \leqslant d$, which proves the claim.

By Lemmas 8 and 9, Theorems 1 and 2 apply yielding the following results.

Theorem 9 *Given a sort-refined-PVD input set S, every fair SGGS-derivation with I^- as initial interpretation halts, is a refutation if S is unsatisfiable, and constructs a model of S if S is satisfiable.*

Corollary 5 *A sort-refined-PVD satisfiable set S of clauses has a model of cardinality at most $|\mathcal{H}(\mathcal{A}^d_{S,\Sigma})| + 1$ that can be extracted from the limit of any fair SGGS-derivation with I^- as initial interpretation and input set S.*

Hyperresolution decides the PVD class [34,25], but it does not decide sort-refined-PVD, because it does not decide the stratified fragment (cf. Example 3), which is contained in sort-refined-PVD as the special case where all sorts have finite domain.

6 Testing for Membership, the Koala Prover, and the Experiments

This section presents first an approach to determine whether a set of clauses is restrained, and then the experiments. We show that a set S of clauses is positively restrained, if an associated rewriting relation terminates and defines a restraining quasi-ordering. The case for negatively restrained sets is dual. Thanks to this reduction, one can have a tool that extracts candidate rewrite systems from a set of clauses and invokes a termination tool to test whether the rewriting relation terminates. Our tool tries both $\mathsf{T_TT_2}$ [48] and AProVE [40] to find *restrained* and *sort-restrained* problems, and it also detects whether a problem belongs to any of the other decidable classes considered in this article.

In the experiments, we applied this tool to classify the problems in the TPTP library [77]. This allows us to assess the relevance of the new decidable classes and the power of SGGS as a decision procedure: it turns out that *SGGS can decide 65% of the decidable problems without interpreted symbols* (e.g., equality)[7] in TPTP 7.4.0. Then, we present Koala, the *first SGGS-based theorem prover.* We tested Koala on all the problems without interpreted symbols in TPTP 7.4.0. We analyze these experiments, report statistics, and compare Koala with several state-of-the-art reasoners.

6.1 Discovering Restrained Sets

In order to show that a clause set S is positively restrained, one needs to find a restraining quasi-ordering (cf. Definitions 5 and 6). Since the strict part of a restraining quasi-ordering is a well-founded ordering, the first intuition is to extract from S a rewrite system $\mathcal{R}_S$ on atoms such that the rewrite relation $\rightarrow_{\mathcal{R}_S}$ is terminating, so that its transitive closure $\rightarrow^+_{\mathcal{R}_S}$ is a well-founded ordering. Then the transitive and reflexive closure $\rightarrow^*_{\mathcal{R}_S}$ is a restraining quasi-ordering whose equivalence relation is identity.

If the problem requires a quasi-ordering whose equivalence is not identity as in Example 14, one needs to extract from S a pair $(\mathcal{R}_S, \mathcal{E}_S)$, where $\mathcal{R}_S$ is a rewrite system and $\mathcal{E}_S$ is a set of equations. Then the rewrite relation is the *rewriting modulo* relation $\rightarrow_{\mathcal{R}_S/\mathcal{E}_S}$, which is defined by $\leftrightarrow^*_{\mathcal{E}_S} \circ \rightarrow_{\mathcal{R}_S} \circ \leftrightarrow^*_{\mathcal{E}_S}$. The crucial point is that $\rightarrow_{\mathcal{R}_S/\mathcal{E}_S}$ is terminating, so that its transitive and reflexive closure $\rightarrow^*_{\mathcal{R}_S/\mathcal{E}_S}$ is a restraining quasi-ordering.

Definition 15 (Restraining system) Given a set S of clauses, a system $(\mathcal{R}_S, \mathcal{E}_S)$ is *positively restraining* for S if for all clauses $C \in S$, for all non-ground literals $L \in C^+$, there exists a literal $\neg M \in C^-$ such that $(M \rightarrow L) \in \mathcal{R}_S$ or $(M \simeq L) \in \mathcal{E}_S$.

Often $\mathcal{E}_S$ contains permutative equations, such as $\mathsf{differ}(x, y) \simeq \mathsf{differ}(y, x)$ in Example 14. For Example 15, a possible choice is $\mathcal{R}_S = \{\mathsf{P}(\mathsf{s}(\mathsf{s}(x)), y) \rightarrow \mathsf{P}(x, \mathsf{s}(y)), \mathsf{P}(x, \mathsf{s}(\mathsf{s}(y))) \rightarrow \mathsf{P}(x, \mathsf{s}(y)), \mathsf{P}(\mathsf{s}(x), \mathsf{s}(y)) \rightarrow \mathsf{P}(\mathsf{s}(x), y)\}$ and $\mathcal{E}_S = \emptyset$.

Theorem 10 *Given a set S of clauses, if there exists a positively restraining system $(\mathcal{R}_S, \mathcal{E}_S)$ for S such that (i) $\rightarrow_{\mathcal{R}_S/\mathcal{E}_S}$ is terminating, (ii) for all $t \simeq u$ in $\mathcal{E}_S$, $\mathcal{V}ar(t) = \mathcal{V}ar(u)$, and (iii) $\leftrightarrow^*_{\mathcal{E}_S}$ has finite equivalence classes, then S is positively restrained, and S is strictly positively restrained if $\mathcal{E}_S = \emptyset$.*

Proof We show that S is positively ground-preserving, that is, for all clauses $C \in S$, it holds that $\mathcal{V}ar(C) \subseteq \mathcal{V}ar(C^-)$. Suppose that C has a non-ground

[7] SGGS and Koala do not have a built-in treatment of equality.

ground: 82	EPR: 1,059	stratified: 1,260	Ackermann: 102
monadic: 750	FO^2: 932	guarded: 569	PVD: 347
restrained: 413	sort-restrained: 1,398	sort-refined-PVD: 1,304	

Table 1 Number of problems found decidable according to different criteria

literal $L \in C^+$. By Definition 15, there exists a rule $t \to u$ in $\mathcal{R}_S$ or an equation $t \simeq u$ in $\mathcal{E}_S$ where $t = M$ and $u = L$ for some literal $\neg M \in C^-$. In the first case, $\mathcal{V}ar(u) \subseteq \mathcal{V}ar(t)$ by hypothesis (i), since otherwise $\to_{\mathcal{R}_S}$, and hence $\to_{\mathcal{R}_S/\mathcal{E}_S}$, would not be terminating. In the second case, $\mathcal{V}ar(u) = \mathcal{V}ar(t)$ by hypothesis (ii). It follows that $\mathcal{V}ar(C^+) \subseteq \mathcal{V}ar(C^-)$ and hence $\mathcal{V}ar(C) \subseteq \mathcal{V}ar(C^-)$. To complete the proof, it suffices to check that $\to^*_{\mathcal{R}_S/\mathcal{E}_S}$ is a restraining quasi-ordering. Indeed, $\to^*_{\mathcal{R}_S/\mathcal{E}_S}$ is stable, its strict part $\to^+_{\mathcal{R}_S/\mathcal{E}_S}$ is well-founded by hypothesis (i), and the equivalence classes of $\leftrightarrow^*_{\mathcal{E}_S}$ are finite by hypothesis (iii). If $\mathcal{E}_S = \emptyset$, the restraining quasi-ordering is $\to^*_{\mathcal{R}_S}$. Indeed, $\to^*_{\mathcal{R}_S}$ is stable, its strict part $\to^+_{\mathcal{R}_S}$ is well-founded by hypothesis (i), and the equivalence classes of $\to^*_{\mathcal{R}_S} \cap {}_{\mathcal{R}_S}^{*}\!\leftarrow$ are finite, because $\to^*_{\mathcal{R}_S} \cap {}_{\mathcal{R}_S}^{*}\!\leftarrow$ is identity.

6.2 Classifying Decidable Problems for the Experiments

The TPTP 7.4.0 library has over 17,000 first-order problems, 4,005 of which do not have interpreted symbols (e.g., equality, arithmetic). If a problem is not in clausal form, our testing tool transforms it into clausal form. Given a set of clauses, the tool extracts all the candidates for restraining rewrite systems. Then the tool invokes T$_\mathsf{T}$T$_2$ and AProVE to determine whether at least a candidate is a restraining rewrite systems satisfying the termination conditions for a restrained set. For instance, problem HWV036-2 (cf. Example 1) is a set of axioms which is combined with sets of ground clauses in several other TPTP problems (e.g., HWV008-2.002 adds 23 ground clauses). We found a terminating positively restraining rewrite system for HWV008-2.002, so that both this problem and HWV036-2 are strictly positively restrained.

Both the number of candidate rewrite systems and their size grow exponentially with the number of literals in the clause set. Thus, 555 problems had to be excluded, because they have more than 500 clauses and the candidate rewrite systems turned out to be too large to handle. Also, for each clause set, T$_\mathsf{T}$T$_2$ and AProVE were applied to at most 100 rewrite systems, with a timeout of 10 sec each. Membership in the already known decidable classes can also be determined automatically. For example, stratified input problems are recognized by computing the sort dependency graph and testing it for acyclicity [47]. This test is applied also to identify sort-restrained and sort-refined-PVD problems.

Table 1 shows how many of the remaining 3,450 problems belong to the various (non-disjoint) decidable classes. Initially, 377 problems were found restrained. For those still undetermined, we tested whether it is sufficient to flip the sign of all literals with a certain predicate to get a restrained problem, which succeeded in 36 cases, for a total of 413 restrained problems. Overall, 2,137 of the 3,450 problems are decidable according to at least one of the criteria, and 1,399 belong to at least one of the SGGS-decidable classes (i.e., ground, EPR, stratified, PVD, restrained, sort-restrained, and sort-refined-PVD), so that *SGGS can decide 65% of the available decidable problems.*

We analyze next how many *new decidable problems* are discovered thanks to the classes introduced in this paper. Of the 413 *restrained* problems in Table 1, 332 are positively restrained, 202 negatively restrained, and 121 are both; 74 are ground, 266 are EPR, 277 are stratified, 89 are Ackermann, 169 are monadic, 204 are FO^2, 209 are guarded, and 232 are PVD, but 77 problems fall in no other decidable class, and therefore, to the best of our knowledge, they are found to be decidable for the first time.

Of the 1,398 *sort-restrained* problems in Table 1, 82 are ground, 1,059 are EPR, 1,260 are stratified, 93 are Ackermann, 406 are monadic, 534 are FO^2, 569 are guarded, 346 are PVD, 413 are restrained, and 20 belong to no other decidable class. Adding these 20 to the above 77 gives *97 new decidable problems*. The existence of problems that are sort-restrained, but neither stratified nor restrained, shows that the generalization conquers more problems.

Of the 1,304 *sort-refined-PVD* problems in Table 1, 82 are ground, 1,059 are EPR, 1,260 are stratified, 93 are Ackermann, 404 are monadic, 515 are FO^2, 569 are guarded, and 347 are PVD. Since 26 sort-refined-PVD problems are neither stratified nor PVD, also this generalization is useful. However, the sort-refined-PVD class did not unveil previously unknown decidable problems.

The average TPTP rating of the problems in the new decidable classes is low,[8] which means that most provers can solve them. Nonetheless, the group of restrained problems includes hard ones such as instances of the binary counter problem in Example 5 (MSC015-1.n), and Rubik's cube problems (e.g., PUZ052-1). For example, MSC015-1.030 is restrained and has rating 1.00, that is, no theorem prover could solve it so far in the time allotted in competitions.

6.3 The Koala Prover and the Experiments

Koala is a new prototype theorem prover written in OCaml and it is the first implementation of SGGS.[9] In Koala, the trail is implemented as a list, with constraints maintained in standard form, and selected literals stored in a *discrimination tree* to compute substitutions efficiently. Koala computes the sort dependency graph, because it facilitates testing sorted constraints for satisfiability. Thus, Koala can detect stratified problems on its own. The *search plans* in Koala are *fair*, so that all derivations are fair.

[8] The average TPTP ratings of the discovered restrained, sort-restrained, and sort-refined-PVD problems are 0.06, 0.08, and 0.08, respectively.

[9] Koala is available at https://github.com/bytekid/koala.

problem class	SAT	UNSAT	#steps	#ext	#confl	#gen	#del	max $\|\Gamma\|$	time
ground	11	68	345	117	141	245	99	8	0.74
EPR	220	538	496	250	154	399	183	106	20.41
stratified	271	667	402	204	123	323	147	89	16.27
monadic	57	223	120	43	46	85	32	9	0.32
FO^2	213	371	143	75	40	113	35	46	6.30
Ackermann	14	79	295	100	120	209	84	7	0.63
guarded	124	216	506	210	187	388	182	27	7.22
PVD	74	230	553	228	206	425	201	6	7.50
sort-refined-PVD	274	699	389	198	119	313	142	87	15.74
restrained	65	313	129	53	46	96	41	19	1.32
sort-restrained	290	772	371	189	114	299	136	84	14.91
other problems	110	288	67	48	8	56	20	46	6.73
all problems	481	1,153	270	143	77	219	96	74	12.79

Table 2 Outcomes, statistics, and average running time for the Koala derivations

In the experiments,[10] the initial interpretation was I^- by default and I^+ for positively ground-preserving problems. The time-out was 300 sec of wall-clock time. All experiments were run single-threaded on a 12-core Intel i7-5930K 3.50GHz machine with 32GB of main memory. Table 2 reports how many problems Koala showed satisfiable or unsatisfiable along with statistics and the average running time. Considering *all problems whose satisfiability status is known*, Koala succeeded on 64% of the satisfiable problems, and on 38% of the unsatisfiable problems, with an overall success rate of 43%. Considering the *problems in SGGS-decidable fragments*, Koala found 360 satisfiable sets and 726 unsatisfiable sets, solving 1,086 problems out of 1,399 (78% success rate). Koala solved 87 of the 97 problems that were discovered *decidable for the first time* (90% success rate). Specifically, it solved 69 of the 77 restrained problems, finding 61 unsatisfiable sets and 8 satisfiable sets, and 18 of the 20 sort-restrained problems, finding 16 unsatisfiable sets and 2 satisfiable sets.

In Table 2, the columns labeled *#steps*, *#ext*, and *#confl* report the *average derivation length*, *average number of SGGS-extensions*, and *average number of conflicts*, respectively. The latter two give some intuition about the time spent in model building and conflict solving, respectively, where the number of conflicts may measure the difficulty of the search. The columns labeled *#gen*, *#del*, and *max* $|\Gamma|$ report space-related statistics: *average number of generated clauses*, *average number of deleted clauses*, and *average maximum trail length* during the derivation. Deleted clauses include disposable clauses and clauses deleted because they have literals assigned to either a split clause or the parent that gets deleted in an SGGS-resolution step. Across all problem classes, SGGS-extensions represent between one third and one half of all inferences, and about half of the generated clauses are extension clauses. The number of deletions is in a similar magnitude as the number of extensions, though somewhat smaller. The number of conflicts equals about one third of the number of inference steps. A comparison of the columns labeled *#steps*

[10] The experimental data are posted at http://cl-informatik.uibk.ac.at/users/swinkler/koala/, http://profs.sci.univr.it/~bonacina/sggs.html or https://github.com/bytekid/koala.

problem class	# sets	Koala	E	Vampire	iProver	CVC5	-fm	Darwin	-fm
ground	71	68	70	**71**	**71**	**71**	**71**	**71**	70
EPR	790	538	561	756	**774**	628	685	750	595
stratified	933	667	698	900	**918**	741	823	894	618
monadic	620	223	408	560	558	343	363	**590**	195
FO^2	575	372	403	518	**531**	406	492	512	283
Ackermann	84	79	83	**84**	**84**	78	**84**	**84**	73
guarded	403	216	241	385	**387**	320	347	384	258
PVD	261	230	226	**251**	**251**	219	242	248	213
sort-refined PVD	969	699	729	932	**953**	771	855	929	622
restrained	338	313	317	**329**	328	316	310	325	216
sort-restrained	1,045	772	796	1,007	**1,029**	837	916	1,002	624
other problems	131	288	585	815	**870**	535	664	768	131
all problems	769	1,153	1,675	2,189	**2,279**	1,462	1,733	2,164	769

Table 3 Problems found unsatisfiable by Koala and some state-of-the-art tools

and *max* $|\Gamma|$ shows that the maximum trail length is much smaller than the derivation length on average. The SGGS trail grows when SGGS-extension expands the model and shrinks when the other rules fix it.

We compared Koala with E 2.4 [74], Vampire 4.4 [51], iProver 3.5 [33], CVC5 1.0.0 [6], and Darwin 1.4.4 [9]. E and Vampire are saturation-based theorem provers, and hence feature ordered resolution, which decides the Ackermann, monadic, FO^2, and guarded fragments [44,35,29].[11] iProver implements both saturation and a model-driven instance-based engine that generates instances of clauses by the Inst-Gen method [39,46], grounds them, and submits them to a SAT-solver: if a ground set is found unsatisfiable, so is the input; otherwise, the next round of instance generation gets instances that are false in the model. Darwin implements the *model evolution calculus* (MEC) [13], which lifts to FOL the DPLL procedure for propositional satisfiability [27]. Inst-Gen and MEC decide the stratified [47] but not the restrained fragments: if Inst-Gen picks an unfortunate literal selection for Example 15, it does not halt, and MEC may not halt on satisfiable negatively restrained sets (e.g., Example 14), as it starts with I^+ as candidate model. CVC5 is a CDCL($\mathcal{T}$)-based SMT solver with instance generation to handle unversally quantified variables; it also features superposition. We included the finite model (-fm) versions of CVC5 and Darwin that search for a finite model by iterative deepening on the model's cardinality, generating all the ground instances of the clauses for a given cardinality, and giving them to a SAT solver.

Table 3 reports how many problems each prover found unsatisfiable in each class, and Table 4 does the same for the problems found satisfiable. The column *#sets* gives for each category the number of problems that have status *unsatisfiable* (in Table 3) or *satisfiable* (in Table 4) in TPTP 7.4.0 (these data are not necessarily current, hence the tools may find more). The best performance is highlighted in boldface. On unsatisfiable problems, Koala trails

[11] Ordered resolution decides FO^2 via a reduction to the Gödel fragment [75,41] that is unlikely to be implemented in provers.

problem class	# sets	Koala	E	Vampire	iProver	CVC5	-fm	Darwin	-fm
ground	11	**11**	**11**	**11**	**11**	**11**	**11**	**11**	**11**
EPR	267	220	118	211	**264**	15	251	263	246
stratified	324	271	144	260	**320**	15	306	319	300
monadic	122	57	56	87	100	14	98	84	**108**
FO^2	349	213	145	240	**288**	13	271	244	287
Ackermann	18	14	**18**	**18**	**18**	13	**18**	14	**18**
guarded	164	124	85	140	**162**	15	150	161	145
PVD	84	74	44	60	**82**	13	80	81	76
sort-refined PVD	330	274	146	262	**324**	15	311	323	303
restrained	72	65	57	66	**68**	13	67	64	65
sort-restrained	348	290	154	278	**342**	15	327	337	319
other problems	199	110	52	78	178	0	**200**	146	199
all problems	713	481	288	456	681	24	676	586	**713**

Table 4 Problems found satisfiable by Koala and some state-of-the-art tools

behind most systems on most or all classes, except for Darwin -fm. Finite model search pays the price of the exponential growth of the search space by iterative deepening, and is less suitable for unsatisfiable instances. However, CVC5 -fm does better, possibly due to a different underlying CDCL-based SAT solver. On satisfiable problems, Koala solves more problems than E, CVC5, and even Vampire in most classes, but remains behind CVC5 -fm, both versions of Darwin, and iProver, which emerges as the strongest system in both tables.

7 Related Work

Several methods decide Datalog (e.g., positive hyperresolution) or the EPR or stratified fragments (e.g., [65,47,3,37]) that are popular for applications [1, 64,57]. The other SGGS-decidable fragments in this article involve *ground-preserving* clauses. Positively ground-preserving clauses are also termed *range-restricted* [61,55,25,12]. Manthey and Bry introduced "range-restricted" for positively ground-preserving clauses [55]. At the same conference Kounalis and Rusinowitch introduced "ground-preserving" for negatively ground-preserving clauses [49,50]. Ground-preserving was used for positively ground-preserving in [21] and for either positively or negatively ground-preserving in [24]. Ground-preserving captures exactly the property we are interested in, namely that only ground clauses get generated. Since the two names are equally old and ground-preserving is more expressive for our purposes, we chose to use it.

Ground-preserving clauses were introduced in Horn logic [55,49,50]. When reasoning *forward* or *bottom-up*, that is, from the facts (e.g., by positive hyperresolution) variables that appear in the positive but not in the negative literals of a rule were deemed problematic, because they get introduced in the forward chaining process. In other words, even if we start from ground facts we get non-ground facts. The restriction to positively ground-preserving clauses was introduced to prevent this phenomenon [55].

When reasoning *backward* or *top-down*, that is, from a query (e.g., by negative hyperresolution) variables that appear in the negative but not in the positive literals of a rule get introduced in the backward chaining process. Even if we start from a ground query we get non-ground queries. The restriction to negatively ground-preserving clauses was introduced to prevent this phenomenon in Horn theories with equality [49, 50]. The purpose was to obtain linear input proofs where all center clauses are ground and decreasing in the CSO used in ordered resolution and superposition (see also [16, Sect. 5.2]). Thus, positively ground-preserving clauses are convenient for positive strategies that reason forward or bottom-up, and negatively ground-preserving clauses are convenient for negative strategies that reason backward or top-down. SGGS with I^- is another forward-reasoning or bottom-up method and SGGS with I^+ is another backward-reasoning or top-down method.

CDCL($\Gamma+\mathcal{T}$), where Γ is an inference system including hyperresolution, superposition with negative selection, and simplification, decides *essentially finite* theories with positively ground-preserving axiomatizations [21]. Essentially finite means only one monadic function f with finite range. This decidability result rests on adding *speculative axioms* of the form $f^j(x) \simeq f^k(x)$ $(j > k)$ for increasing values of j and k. Simplification applies the speculative axioms to limit the depth of generated terms.

Example 21 The clause set $\{\mathsf{P}(\mathsf{a}),\ \neg\mathsf{P}(x) \vee \mathsf{P}(\mathsf{f}(\mathsf{f}(x))),\ \neg\mathsf{P}(x) \vee \neg\mathsf{P}(\mathsf{f}(x))\}$ from Example 12 is essentially finite (the range of f is finite, because the set admits a finite model). CDCL($\Gamma+\mathcal{T}$) tries $\mathsf{f}(x) \simeq x$, detects a conflict, backtracks, tries $\mathsf{f}^2(x) \simeq x$, and halts reporting satisfiability. Without speculative axioms and simplification, it is not surprising that SGGS does not halt.

Baumgartner and Schmidt offered a comprehensive treatment of *bottom-up model-generation* (BUMG) methods, with an emphasis on positive hyperresolution enhanced with *first-order splitting* [12]. First-order splitting [80, 70] generalizes to first-order clauses the splitting of disjunctions of DPLL, at the expense of introducing backtracking in saturation. The model generation and decidability results in [12] involve *range-restriction transformations* and a technique called *blocking*. A range-restriction transformation transforms a set of clauses into an equisatisfiable set of range-restricted clauses. Blocking allows the BUMG method to guess an equality on a splitting branch and its negation on another. If a guess causes a conflict it can be undone by backtracking. Thus, these guesses are speculative inferences in the sense of [21]. Since positive hyperresolution with these enhancements decides the Bernays-Schönfinkel class with equality [12], one can conjecture that a generalization to the many-sorted case could enable it to decide the sort-restrained and sort-refined-PVD classes.

8 Future work

A key open issue in automated reasoning is whether it is better to bring conflict-driven reasoning to the first-order level (e.g., SGGS) or keep it at the propositional level, as done by the instance-based approaches that perform instance generation on top of a CDCL-based SAT solver. Some of the systems that we compared with Koala include in various ways the second approach. We do not regard our experimental comparison as conclusive, because Koala is only a prototype. This fundamental problem is open also in SMT, where it is still unknown whether it is better to stick to the CDCL($\mathcal{T}$) paradigm (conflict-driven reasoning only at the propositional level) [63,21] or move to the MCSAT/CDSAT paradigm (conflict-driven reasoning in the theories) [28, 18,19]. An answer based on experiments is premature, as not enough engineering has been invested in first-order conflict-driven systems. Furthermore, comparing implementations is necessary, but it is a comparison of tools, not methods [20]. This is all the more true given that contemporary reasoners implement multiple paradigms. This is a welcome development to get more powerful and flexible provers, but it may make it harder to know to which features a certain empirical behavior should be attributed.

In addition to this broad issue, there are several directions for future work on SGGS. A main one is to add equality reasoning by building the equality axioms in both model representation and rules. A natural candidate would be an SGGS-superposition rule, focused on generating clauses needed to explain equality conflicts. For the ground case, one may integrate in SGGS a *congruence closure* algorithm (e.g., [60,32,5,62]). Congruence closure and blocking [12] were used to import some equational reasoning in tableaux-based methods [79]. SGGS could be enhanced with blocking or other speculative inferences. Speculative inferences that cause conflicts are undone by backtracking [21,79,12]. SGGS does not have backtracking in the sense of undoing inferences. Since the model in SGGS is read off the trail in left-to-right order, it suffices to move a clause by the SGGS-move inference rule to flip the truth value of a selected literal in the candidate model. One would have then to fit speculative inferences in the SGGS approach to represent and fix models.

Methods that integrate first-order theorem proving and SMT solving have gained traction (e.g., [21,69]). One can envision composing SGGS with *theory modules* in CDSAT [18,19], viewing SGGS as a CDSAT module for FOL. Such a composition would lead to study how to bridge Herbrand interpretations and models represented by assignments, including first-order (i.e., non-Boolean) assignments. It would also be a context where to consider SGGS with initial interpretations that are not based on sign, since SGGS could assume as initial interpretation some completion of a partial interpretation built by CDSAT.

An initial interpretation is *goal-sensitive*, if it satisfies all the input clauses except the goal clauses, that is, those in the clausal form of the negation of the conjecture. If the initial interpretation is goal-sensitive, SGGS is goal-sensitive, meaning that it generates only clauses connected to goal clauses [23]. It is open whether goal-sensitivity is useful to reason in large knowledge bases.

The experiments with Koala allow us to identify critical issues for the performance of an SGGS prover. For example, instance generation by SGGS-extension may be a bottleneck for problems with many input clauses, and forms of *caching* should be considered to avoid repeating computations.

Acknowledgements We thank Konstantin Korovin for the iProver (v2.8) code for basic data structures, term indexing, and type inference, imported in Koala. Parts of this work were done while the first author was visiting the Simons Institute for the Theory of Computing, the Leibniz Zentrum für Informatik at Schloss Dagstuhl, and the Computer Science Laboratory of SRI International, whose support is greatly appreciated.

References

1. Abadi, A., Rabinovich, A., Sagiv, M.: Decidable fragments of many-sorted logic. J. Symb. Comput. **45**(2), 153–172 (2010). DOI 10.1016/j.jsc.2009.03.003
2. Ackermann, W.: Solvable Cases of the Decision Problem. North Holland, Amsterdam (1954). DOI 10.1007/BFb0022557
3. Alagi, G., Weidenbach, C.: NRCL – a model building approach to the Bernays-Schönfinkel fragment. In: C. Lutz, S. Ranise (eds.) Proceedings of FroCoS-10, *Lecture Notes in Artificial Intelligence*, vol. 9322, pp. 69–84. Springer, Berlin (2015). DOI 10.1007/978-3-319-24246-0_5
4. Andréka, H., van Benthem, J., Nemeti, I.: Modal logics and bounded fragments of predicate logic. J. Phil. Log. **27**(3), 217–274 (1998). DOI 10.1023/A:1004275029985
5. Bachmair, L., Tiwari, A., Vigneron, L.: Abstract congruence closure. J. Autom. Reason. **31**(2), 129–168 (2003). DOI 10.1023/B:JARS.0000009518.26415.49
6. Barbosa, H., Barrett, C.W., Brain, M., Kremer, G., Lachnitt, H., Mann, M., Mohamed, A., Mohamed, M., Niemetz, A., Nötzli, A., Ozdemir, A., Preiner, M., Reynolds, A., Sheng, Y., Tinelli, C., Zohar, Y.: CVC5: A versatile and industrial-strength SMT solver. In: D. Fisman, G. Rosu (eds.) Proceedings of TACAS-28, *Lecture Notes in Computer Science*, vol. 13243, pp. 415–442. Springer, Berlin (2022). DOI 10.1007/978-3-030-99524-9_24
7. Barrett, C.W., Nieuwenhuis, R., Oliveras, A., Tinelli, C.: Splitting on demand in SAT modulo theories. In: M. Hermann, A. Voronkov (eds.) Proceedings of LPAR-13, *Lecture Notes in Artificial Intelligence*, vol. 4246, pp. 512–526. Springer, Berlin (2006). DOI 10.1007/11916277_35
8. Baumgartner, P.: Hyper tableaux – the next generation. In: H. de Swart (ed.) Proceedings of TABLEAUX-7, *Lecture Notes in Artificial Intelligence*, vol. 1397, pp. 60–76. Springer, Berlin (1998)
9. Baumgartner, P., Fuchs, A., Tinelli, C.: Implementing the model evolution calculus. Int. J. Artif. Intell. Tools **15**(1), 21–52 (2006). DOI 10.1142/S0218213006002552
10. Baumgartner, P., Furbach, U., Niemelä, I.: Hyper tableaux. In: J.J. Alferes, L.M. Pereira, E. Orłowska (eds.) Proceedings of JELIA-5, *Lecture Notes in Artificial Intelligence*, vol. 1126, pp. 1–17. Springer, Berlin (1996)
11. Baumgartner, P., Furbach, U., Pelzer, B.: The hyper tableaux calculus with equality and an application to finite model computation. J. Log. Comput. **20**(1), 77–109 (2008)
12. Baumgartner, P., Schmidt, R.A.: Blocking and other enhancements for bottom-up model generation methods. J. Autom. Reason. **64**, 197–251 (2020). DOI 10.1007/s10817-019-09515-1
13. Baumgartner, P., Tinelli, C.: The model evolution calculus as a first-order DPLL method. Artif. Intell. **172**(4-5), 591–632 (2008). DOI 10.1016/j.artint.2007.09.005
14. Bernays, P., Schönfinkel, M.: Zum Entscheidungsproblem der mathematischen Logik. Mathematische Annalen **99**, 342–372 (1928). DOI 10.1007/BF01459101
15. Bonacina, M.P.: On conflict-driven reasoning. In: N. Shankar, B. Dutertre (eds.) Proceedings of the 6th Workshop on Automated Formal Methods (AFM) May 2017, *Kalpa Publications*, vol. 5, pp. 31–49. EasyChair (2018). DOI 10.29007/spwm
16. Bonacina, M.P., Dershowitz, N.: Canonical ground Horn theories. In: A. Voronkov, C. Weidenbach (eds.) Programming Logics: Essays in Memory of H. Ganzinger, *Lecture Notes in Computer Science*, vol. 7797, pp. 35–71. Springer, Berlin (2013). DOI 10.1007/978-3-642-37651-1_3
17. Bonacina, M.P., Furbach, U., Sofronie-Stokkermans, V.: On first-order model-based reasoning. In: N. Martí-Oliet, P. Olveczky, C. Talcott (eds.) Logic, Rewriting, and Concurrency: Essays Dedicated to José Meseguer, *Lecture Notes in Computer Science*, vol. 9200, pp. 181–204. Springer, Berlin (2015). DOI 10.1007/978-3-319-23165-5_8

18. Bonacina, M.P., Graham-Lengrand, S., Shankar, N.: Conflict-driven satisfiability for theory combination: transition system and completeness. J. Autom. Reason. **64**(3), 579–609 (2020). DOI 10.1007/s10817-018-09510-y
19. Bonacina, M.P., Graham-Lengrand, S., Shankar, N.: Conflict-driven satisfiability for theory combination: lemmas, modules, and proofs. J. Autom. Reason. **66**(1), 43–91 (2022). DOI 10.1007/s10817-021-09606-y
20. Bonacina, M.P., Hsiang, J.: On the modelling of search in theorem proving – towards a theory of strategy analysis. Inf. Comput. **147**, 171–208 (1998). DOI 10.1006/inco.1998.2739
21. Bonacina, M.P., Lynch, C.A., de Moura, L.: On deciding satisfiability by theorem proving with speculative inferences. J. Autom. Reason. **47**(2), 161–189 (2011). DOI 10.1007/s10817-010-9213-y
22. Bonacina, M.P., Plaisted, D.A.: Semantically-guided goal-sensitive reasoning: model representation. J. Autom. Reason. **56**(2), 113–141 (2016). DOI 10.1007/s10817-015-9334-4
23. Bonacina, M.P., Plaisted, D.A.: Semantically-guided goal-sensitive reasoning: inference system and completeness. J. Autom. Reason. **59**(2), 165–218 (2017). DOI 10.1007/s10817-016-9384-2
24. Bonacina, M.P., Winkler, S.: SGGS decision procedures. In: N. Peltier, V. Sofronie-Stokkermans (eds.) Proceedings of IJCAR-10, *Lecture Notes in Artificial Intelligence*, vol. 12166, pp. 356–374. Springer, Berlin (2020). DOI 10.1007/978-3-030-51074-9_20
25. Caferra, R., Leitsch, A., Peltier, N.: Automated Model Building. Kluwer Academic Publishers (2004). DOI 10.1007/978-1-4020-2653-9
26. Ceri, S., Gottlob, G., Tanca, L.: What you always wanted to know about Datalog (and never dared to ask). IEEE Trans. Knowl. Data Eng. **1**(1), 146–166 (1989). DOI 10.1109/69.43410
27. Davis, M., Logemann, G., Loveland, D.: A machine program for theorem-proving. C. ACM **5**(7), 394–397 (1962). DOI 10.1145/368273.368557
28. de Moura, L., Jovanović, D.: A model-constructing satisfiability calculus. In: R. Giacobazzi, J. Berdine, I. Mastroeni (eds.) Proceedings of VMCAI-14, *Lecture Notes in Computer Science*, vol. 7737, pp. 1–12. Springer, Berlin (2013). DOI 10.1007/978-3-642-35873-9_1
29. de Nivelle, H., de Rijke, M.: Deciding the guarded fragments by resolution. J. Symb. Comput. **35**(1), 21–58 (2003). DOI 10.1016/S0747-7171(02)00092-5
30. Dershowitz, N.: Orderings for term-rewriting systems. Theoret. Comput. Sci. **17**(3), 279–301 (1982). DOI 10.1016/0304-3975(82)90026-3
31. Dershowitz, N., Plaisted, D.A.: Rewriting. In: J.A. Robinson, A. Voronkov (eds.) Handbook of Automated Reasoning, vol. 1, chap. 9, pp. 535–610. Elsevier (2001). DOI 10.1016/b978-044450813-3/50011-4
32. Downey, P.J., Sethi, R., Tarjan, R.E.: Variations on the common subexpression problem. J. ACM **27**(4), 758–771 (1980). DOI 10.1145/322217.322228
33. Duarte, A., Korovin, K.: Implementing superposition in iProver (system description). In: N. Peltier, V. Sofronie-Stokkermans (eds.) Proceedings of IJCAR-10, *Lecture Notes in Computer Science*, vol. 12167, pp. 388–397. Springer, Berlin (2020). DOI 10.1007/978-3-030-51054-1_24
34. Fermüller, C.G., Leitsch, A.: Model building by resolution. In: E. Börger, G. Jäger, H. Kleine Büning, S. Martini (eds.) Proceedings of CSL-6, *Lecture Notes in Computer Science*, vol. 702, pp. 134–148. Springer, Berlin (1993). DOI 10.1007/3-540-56992-8_10
35. Fermüller, C.G., Leitsch, A., Hustadt, U., Tammet, T.: Resolution decision procedures. In: Handbook of Automated Reasoning, pp. 1791–1849. Elsevier and MIT Press, Amsterdam and Cambridge (2001). DOI 10.1016/b978-044450813-3/50027-8
36. Fermüller, C.G., Salzer, G.: Ordered paramodulation and resolution as decision procedure. In: A. Voronkov (ed.) Proceedings of LPAR-4, *Lecture Notes in Artificial Intelligence*, vol. 698, pp. 122–133. Springer, Berlin (1993). DOI 10.1007/3-540-56944-8_47
37. Fiori, A., Weidenbach, C.: SCL clause learning from simple models. In: P. Fontaine (ed.) Proceedings of CADE-27, *Lecture Notes in Artificial Intelligence*, vol. 11716, pp. 233–249. Springer, Berlin (2019). DOI 10.1007/978-3-030-29436-6_14

38. Fontaine, P.: Combinations of theories for decidable fragments of first-order logic. In: S. Ghilardi, R. Sebastiani (eds.) Proceedings of FroCoS-7, *Lecture Notes in Artificial Intelligence*, vol. 5749, pp. 263–278. Springer, Berlin (2009). DOI 10.1007/978-3-642-04222-5_16
39. Ganzinger, H., Korovin, K.: New directions in instantiation-based theorem proving. In: Proceedings of LICS-18, pp. 55–64. IEEE (2003)
40. Giesl, J., Aschermann, C., Brockschmidt, M., Emmes, F., Frohn, F., Fuhs, C., Hensel, J., Otto, C., Plücker, M., Schneider-Kamp, P., Ströder, T., Swiderski, S., Thiemann, R.: Analyzing program termination and complexity automatically with AProVE. J. Autom. Reason. **58**(1), 3–31 (2017). DOI 10.1007/s10817-016-9388-y
41. Grädel, E., Kolaitis, P.G., Vardi, M.Y.: On the decision problem for two-variable first-order logic. Bull. Symb. Log. **3**, 53–69 (1997). DOI 10.2307/421196
42. Hsiang, J., Rusinowitch, M.: Proving refutational completeness of theorem proving strategies: the transfinite semantic tree method. J. ACM **38**(3), 559–587 (1991). DOI 10.1145/116825.116833
43. Hustadt, U., Schmidt, R.A., Georgieva, L.: A survey of decidable first-order fragments and description logics. J. of Relational Methods in Computer Science **1**, 251–276 (2004). DOI 10.1007/978-3-642-37651-1_15
44. Joyner Jr., W.H.: Resolution strategies as decision procedures. J. ACM **23**(3), 398–417 (1976). DOI 10.1145/321958.321960
45. Knuth, D.E., Bendix, P.B.: Simple word problems in universal algebras. In: J. Leech (ed.) Proceedings of the Conference on Computational Problems in Abstract Algebras, pp. 263–298. Pergamon Press (1970). DOI 10.1016/B978-0-08-012975-4
46. Korovin, K.: Inst-Gen – a modular approach to instantiation-based automated reasoning. In: A. Voronkov, C. Weidenbach (eds.) Programming Logics: Essays in Memory of H. Ganzinger, *Lecture Notes in Computer Science*, vol. 7797, pp. 239–270. Springer, Berlin (2013). DOI 10.1007/978-3-642-37651-1_10
47. Korovin, K.: Non-cyclic sorts for first-order satisfiability. In: P. Fontaine, C. Ringeissen, R.A. Schmidt (eds.) Proceedings of FroCoS-9, *Lecture Notes in Artificial Intelligence*, vol. 8152, pp. 214–228. Springer, Berlin (2013). DOI 10.1007/978-3-642-40885-4_15
48. Korp, M., Sternagel, C., Zankl, H., Middeldorp, A.: Tyrolean Termination Tool 2. In: R. Treinen (ed.) Proceedings of RTA-20, *Lecture Notes in Computer Science*, vol. 5595, pp. 295–304. Springer, Berlin (2009). DOI 10.1007/978-3-642-02348-4_21
49. Kounalis, E., Rusinowitch, M.: On word problems in Horn theories. In: E. Lusk, R. Overbeek (eds.) Proceedings of CADE-9, *Lecture Notes in Computer Science*, vol. 310, pp. 527–537. Springer, Berlin (1988). DOI 10.1007/BFb0012854
50. Kounalis, E., Rusinowitch, M.: On word problems in Horn theories. J. Symb. Comput. **11**(1–2), 113–128 (1991). DOI 10.1016/S0747-7171(08)80134-4
51. Kovács, L., Voronkov, A.: First-order theorem proving and Vampire. In: N. Sharygina, H. Veith (eds.) Proceedings of CAV-25, *Lecture Notes in Computer Science*, vol. 8044, pp. 1–35. Springer, Berlin (2013). DOI 10.1007/978-3-642-39799-8_1
52. Lamotte-Schubert, M., Weidenbach, C.: BDI: a new decidable clause class. J. Log. Comput. **27**(2), 441–468 (2017). DOI 10.1093/logcom/exu074
53. Lee, S.J., Plaisted, D.A.: Eliminating duplication with the hyperlinking strategy. J. Autom. Reason. **9**, 25–42 (1992)
54. Ludwig, M., Waldmann, U.: An extension of the Knuth-Bendix ordering with LPO-like properties. In: N. Dershowitz, A. Voronkov (eds.) Proceedings of LPAR-14, *Lecture Notes in Artificial Intelligence*, vol. 4790, pp. 348–362. Springer, Berlin (2007). DOI 10.1007/978-3-540-75560-9_26
55. Manthey, R., Bry, F.: SATCHMO: a theorem prover implemented in Prolog. In: E. Lusk, R. Overbeek (eds.) Proceedings of CADE-9, *Lecture Notes in Computer Science*, vol. 310, pp. 415–434. Springer, Berlin (1988). DOI 10.1007/BFb0012847
56. Marques Silva, J., Lynce, I., Malik, S.: Conflict-driven clause learning SAT solvers. In: A. Biere, M. Heule, H. Van Maaren, T. Walsh (eds.) Handbook of Satisfiability, *Frontiers in Artificial Intelligence and Applications*, vol. 185, pp. 131–153. IOS Press (2009). DOI 10.3233/978-1-58603-929-5-131
57. McMillan, K.L.: Developing distributed protocols with Ivy. Slides from http://vmcaischool19.tecnico.ulisboa.pt/ (2019)

58. Mei, H., Qin, G., Xu, M., Esiner, J.: Neural Datalog through time: informed temporal modeling via logical specification. In: H. Daumé III, A. Singh (eds.) Proceedings of ICML-37, *Proceedings of Machine Learning Research*, vol. 119, pp. 6808–6819 (2020)
59. Navarro, J.A., Voronkov, A.: Proof systems for effectively propositional logic. In: A. Armando, P. Baumgartner, G. Dowek (eds.) Proceedings of IJCAR-4, *Lecture Notes in Artificial Intelligence*, vol. 5195, pp. 426–440. Springer, Berlin (2008). DOI 10.1007/978-3-540-71070-7_36
60. Nelson, G., Oppen, D.C.: Fast decision procedures based on congruence closure. J. ACM **27**(2), 356–364 (1980). DOI 10.1145/322186.322198
61. Nicolas, J.M.: Logic for improving integrity checking in relational databases. Acta Infor. **18**(3), 227–253 (1982). DOI 10.1145/322186.322198
62. Nieuwenhuis, R., Oliveras, A.: Fast congruence closure and extensions. Inf. Comput. **205**(4), 557–580 (2007). DOI 10.1016/j.ic.2006.08.009
63. Nieuwenhuis, R., Oliveras, A., Tinelli, C.: Solving SAT and SAT modulo theories: from an abstract Davis-Putnam-Logemann-Loveland procedure to DPLL(T). J. ACM **53**(6), 937–977 (2006)
64. Padon, O., McMillan, K.L., Panda, A., Sagiv, M., Shoham, S.: Ivy: Safety verification by interactive generalization. SIGPLAN Notices **51**(6), 614–630 (2016). DOI 10.1145/2980983.2908118
65. Piskac, R., de Moura, L., Bjørner, N.: Deciding effectively propositional logic using DPLL and substitution sets. J. Autom. Reason. **44**(4), 401–424 (2010). DOI 10.1007/978-3-540-71070-7_35
66. Plaisted, D.A., Zhu, Y.: The Efficiency of Theorem Proving Strategies. Friedr. Vieweg & Sohn (1997)
67. Plaisted, D.A., Zhu, Y.: Ordered semantic hyper linking. J. Autom. Reason. **25**, 167–217 (2000)
68. Ramsey, F.P.: On a problem in formal logic. Proceedings of the London Mathematical Society **30**, 264–286 (1930). DOI 10.1112/plms/s2-30.1.264
69. Reger, G., Suda, M., Voronkov, A.: Playing with AVATAR. In: A.P. Felty, A. Middeldorp (eds.) Proceedings of CADE-25, *Lecture Notes in Artificial Intelligence*, vol. 9195, pp. 399–415. Springer, Berlin (2015). DOI 10.1007/978-3-319-21401-6_28
70. Riazanov, A.: Implementing an efficient theorem prover. Ph.D. thesis, Department of Computer Science, The University of Manchester (2003)
71. Robinson, J.A.: Automatic deduction with hyper-resolution. Int. J. of Computer Mathematics **1**, 227–234 (1965). DOI 10.2307/2272384
72. Robinson, J.A.: A machine oriented logic based on the resolution principle. J. ACM **12**(1), 23–41 (1965). DOI 10.1145/321250.321253
73. Rubio, A.: A fully syntactic AC-RPO. Inf. Comput. **178**(2), 515–533 (2002). DOI 10.1006/inco.2002.3158
74. Schulz, S., Cruanes, S., Vukmirovic, P.: Faster, higher, stronger: E 2.3. In: P. Fontaine (ed.) Proceedings of CADE-27, *Lecture Notes in Computer Science*, vol. 11716, pp. 495–507. Springer, Berlin (2019). DOI 10.1007/978-3-030-29436-6_29
75. Scott, D.: A decision method for validity of sentences in two variables. J. Symb. Log. **27**, 377–377 (1962)
76. Slagle, J.R.: Automatic theorem proving with renamable and semantic resolution. J. ACM **14**(4), 687–697 (1967). DOI 10.1145/321420.321428
77. Sutcliffe, G.: The TPTP Problem Library and Associated Infrastructure. From CNF to TH0, TPTP v6.4.0. J. Autom. Reason. **59**(4), 483–502 (2017). DOI 10.1007/s10817-009-9143-8
78. van Gelder, A., Topor, R.W.: Safety and translation of relational calculus queries. ACM Trans. Database Syst. **16**(2), 235–278 (1991). DOI 10.1145/114325.103712
79. Waldmann, U., Schmidt, R.A.: Modal tableau systems with blocking and congruence closure. In: H. de Nivelle (ed.) Proceedings of TABLEAUX-24, *Lecture Notes in Artificial Intelligence*, vol. 9323, pp. 38–53. Springer, Berlin (2015). DOI 10.1007/978-3-319-24312-2_4
80. Weidenbach, C.: Combining superposition, sorts and splitting. In: A. Robinson, A. Voronkov (eds.) Handbook of Automated Reasoning, vol. 2, pp. 1965–2012. Elsevier, Amsterdam (2001). DOI 10.1016/b978-044450813-3/50029-1