Journal of Automated Reasoning manuscript No.
(will be inserted by the editor)

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

Maria Paola Bonacina and Sarah Winkler

the date of receipt and acceptance should be inserted later

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 theorem-proving method for decision procedures, because it generalizes the Conflict-Driven Clause Learning (CDCL) procedure for propositional satisfiability, and it is both refutationally complete and model-complete in the limit, so that SGGS decision procedures are *model-constructing*. We investigate the termination of SGGS with both positive and negative results: for example, SGGS decides the *stratified fragment*, and hence Effectively PRopositional logic (EPR). 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, focusing on problems in SGGS-decidable classes.

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, Libera Università di Bolzano, Piazza Domenicani 3, 39100 Bolzano, 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 formulæ were shown decidable (e.g., [28,20,36] 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 article we apply this approach to SGGS, or *Semantically-Guided Goal-Sensitive reasoning* [17,18]. SGGS is *semantically guided* by a fixed initial interpretation. Most of the results in this article assume *sign-based semantic guidance* as in *hyperresolution* [59]: the initial interpretation is either *all-negative* (all negative literals are true, as in positive hyperresolution) or *all-positive* (all positive literals are true, as in negative hyperresolution). SGGS is *model-based*, as it searches for a model of the input set of clauses by building candidate models that modify the initial interpretation. SGGS is *conflict-driven*, as it applies inferences such as resolution only to explain conflicts between candidate model and clauses to be satisfied. Indeed, SGGS is a generalization to FOL of the CDCL (Conflict Driven Clause Learning) procedure for propositional satisfiability [47]. SGGS is also *model complete in the limit*, which means that given a satisfiable input, the limit of any fair derivation represents a model. It follows that model generation is guaranteed if termination is, and SGGS-based decision procedures are *model-constructing*. Surveys of related methods are available [13,11].

Since SGGS does not have a built-in treatment of equality, we consider *first-order logic without equality*. A first issue is the behavior of SGGS on known decidable classes that we recall here for clarity. Let φ be a formula with no occurrences of either quantifiers or functions, while constants are allowed. The *Bernays-Schönfinkel class* [10,56] includes the sentences of the form $\exists^*\forall^*\varphi$, whose transformation in clausal form yields the EPR (*Effectively PRopositional*) fragment [54,3,30]. The *stratified* fragment generalizes EPR to many-sorted logic [1,40,53]. The *Ackermann class* comprises the sentences of the form $\exists^*\forall\exists^*\varphi$ [2,37,29]. The *monadic* or *Löwenheim class* contains the sentences where there are no functions and predicates are unary [2,37,31]. In the *two-variable* fragment, denoted FO^2, there are only two variables and no functions [33,31]. 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 the key property is that all its variables appear in the atomic guard $R(\bar{x},\bar{y})$ [4,23]. In most cases, the clausal version of a fragment contains the sets of clauses obtained by transforming sentences into clausal form; for the guarded fragment, we adopt the existing notion of *guarded clauses* [23].

In SGGS a candidate model is represented as a sequence of clauses, called a *trail*. Previous work showed that if the length of the trail during a fair SGGS-derivation is upper bounded, the derivation is finite [18]. We introduce the concept of a *finite basis* for SGGS, meaning a finite subset of the Herbrand base. We show that if all ground instances of all clauses generated in a fair

SGGS-derivation are made of atoms coming from a finite basis, the length of the trail is upper bounded. If for all sets S of clauses in a fragment $\mathcal{F}$ it is possible to identify a finite basis, which typically depends on S, the termination of SGGS is guaranteed. Also, $\mathcal{F}$ has the *small model property*, because we show that the cardinality of the model built by SGGS for a satisfiable S in $\mathcal{F}$ is upper bounded. Since the Herbrand base of a stratified clause set is finite, *SGGS decides the stratified fragment*. These results make no assumption on the initial Herbrand interpretation guiding SGGS. Then, we give counterexamples showing that SGGS with sign-based semantic guidance *cannot* decide the Ackermann, monadic, FO^2, and guarded fragments.

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 databases [51,65], theorem proving [46,42,43], model building [27,20], and decision procedures [27,20,16,44,8]. Under the assumption that the input clauses are ground-preserving, we prove first that SGGS with sign-based semantic guidance generates only ground clauses, and then that it *terminates whenever hyperresolution does*. It follows that SGGS decides all ground-preserving fragments decided by hyperresolution, such as the *positive variable dominated* (PVD) [27,20] and *the bounded depth increase* (BDI) [44] fragments. However, many theorem-proving problems do not belong to any known decidable class.

Example 1 Problem HWV036-2 from TPTP 7.3.0 [64] specifies a full-adder in 51 clauses, including for instance:

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

This set is satisfiable, which means that termination of a theorem prover is *a priori* not guaranteed. However, it is neither EPR, nor Ackermann, nor monadic, nor FO^2. One can also check that it is neither guarded nor stratified.

Therefore, we apply SGGS to discover *new* decidable fragments. By building on top of ground-preservingness, we define the *positively/negatively restrained* fragments with an ordering-based restriction. By taking sorts into account as in the stratified fragment, we define the *sort-restrained* classes, which are a generalization of both stratified and restrained fragments, and the *sort-refined-PVD* class, which is a generalization of both stratified and PVD fragments. We show that *SGGS with sign-based semantic guidance decides all these new classes* by the finite basis approach. It follows that the new classes have the *small model property*. We prove that sign-based resolution strategies (e.g., PO-resolution and hyperresolution) decide the restrained fragments, whereas they do not decide the sort-restrained and sort-refined-PVD classes, due to the stratified part. The termination of an inference system is *modular* if termination on any input set S_1 from fragment $\mathcal{F}_1$ and S_2 from fragment $\mathcal{F}_2$ implies termination on any disjoint union $S_1 \uplus S_2$. We prove that the termination of SGGS is modular if the SGGS-derivations in $\mathcal{F}_1$ and $\mathcal{F}_2$ are in finite bases, and S_1 and S_2 do not share predicate symbols.

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 TTT2 [41] and AProVE [32].

In the experiments, we applied these tools to discover restrained and sort-restrained problems in the TPTP library [64]. This allows us to evaluate empirically the relevance of these classes and discover problems not previously known to be decidable. For instance, the axiomatization in Example 1, as well as all the TPTP problems including it, is found to be restrained. Then we describe the new SGGS-based 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 article is organized as follows. Section 2 provides the background, including an overview of SGGS with sorts and sign-based initial interpretation. Section 3 presents the finite basis approach, all the results about SGGS and already known decidable fragments, and the theorem relating the termination of SGGS to that of hyperresolution. Section 4 introduces the new decidable fragments and the results showing that they are SGGS-decidable. Section 5 treats modularity of termination. Section 6 presents the reduction of membership in the new fragments to termination of rewriting, the application of the termination tools, the Koala prover, and the experiments with Koala. Section 7 includes comparison with related work, summary of contributions, and directions for future work. A short version of this article appeared [19].

2 Background

Let S be a finite set of clauses in many-sorted logic. We use Σ for the set of sorts, $s, s_1, s_2, \ldots$ 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, and $top(t)$ for the top symbol of t. Sorts are not empty (there is a ground term for every sort), and $t\colon s$ says that t has sort s. We also use L, M, P, Q for literals, $at(L)$ for L's atom, C, D, E for clauses, $\alpha, \sigma, \vartheta, \tau$ for substitutions, I, J for Herbrand interpretations, and $\mathcal{I}$ for other models. 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. C^+ and C^- are the disjunctions of the positive and negative literals in C; clause C is *positive* if $C = C^+$, *negative* if $C = C^-$.

Terms and atoms can be viewed 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 an atom or literal L is defined as $\text{depth}(L) = \text{depth}(at(L)) = 1 + max\{\text{depth}(t_i) : 1 \leqslant i \leqslant n\}$ 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 denotes the top position, and $f(t_1, \ldots, t_n)|_{ip} = t_i|_p$ for all i, $1 \leqslant i \leqslant n$. A term t has *occurrence depth* k in atom L if $L|_p = t$ and $|p| = k$, where $|p|$ is the length of position p.

An ordering $>$ on terms and atoms 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 $c[t] > c[u]$ for all contexts c (a context is a term with a hole), and has the *subterm property*, if $c[t] > t$ for all non-empty contexts c. An ordering is a *simplification ordering*, if it is stable, monotonic, and has the subterm property. A *complete simplification ordering* (CSO) is also total on ground objects. A simplification ordering is well-founded [24]. In the absence of equality, an ordering on atoms is extended to literals by ignoring signs. *Recursive path orderings* (RPO's) [24] and *Knuth-Bendix orderings* (KBO's) [38, 45] employ a *precedence*, which is a partial ordering $\succ_p$ on the symbols in the signature. KBO's attribute a non-negative *weight* to each symbol: all variables have weight w_0 and a weight function w assigns a weight to every non-variable symbol. RPO's and KBO's are simplification orderings. If $\succ_p$ is total, KBO's and *lexicographic (recursive) path orderings* (LPO's) are CSO's. Surveys on orderings and rewriting are available (e.g., [25]).

2.1 Background About Resolution Strategies

The resolution principle [60] involves two inference rules. *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* [59,34] or *P1-deduction* [55], requires that every binary resolution step has a positive parent. *Negative resolution*, also known as *all-negative-resolution* [55], requires that every binary resolution step has a negative parent.

Semantic resolution [63] generates only resolvents that are false in a fixed guiding interpretation I. *Positive hyperresolution* is semantic resolution where I is the *all-negative interpretation* I^- that makes all negative literals true [59]. It 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 with the *all-positive interpretation* I^+ that makes all positive literals true. Hyperresolution without attribute is positive hyperresolution.

Ordered resolution [34] 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 $(L_1 \vee \ldots \vee L_k \vee C)\sigma$. *Positive ordered resolution*, abbreviated *PO-resolution*, adds the requirement that $L' \vee D$ is positive and drops the $>$-maximality constraint on $\neg L\sigma$.

2.2 Background About SGGS

SGGS [17,18] 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 syntactic identity. The negation, conjunction, and disjunction of constraints is a constraint. 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. Substitutions are *sort-preserving* (i.e., $x\sigma$ has the same sort as x) so that instantiation respects sorts. The set $Gr(A \rhd C)$ of *constrained ground instances* (cgi's) of $A \rhd C$ is the set of ground instances of C that satisfy A. 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. Constraints can be omitted if irrelevant.

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, $top(x) \neq \mathsf{a} \rhd \mathsf{P}(\mathsf{f}(x), y)$ is equivalent to $\mathit{false} \rhd \mathsf{P}(\mathsf{f}(x), y)$, while $top(y) \neq \mathsf{a} \rhd \mathsf{P}(\mathsf{f}(x), y)$ is equivalent to $\mathit{true} \rhd \mathsf{P}(\mathsf{f}(x), y)$ with cgi's $P(\mathsf{f}(\mathsf{a}), \mathsf{b})$ and $P(\mathsf{f}(\mathsf{a}), \mathsf{f}(\mathsf{a}))$.

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 default to complete them. If the empty clause $\bot$ arises in the process, unsatisfiability is reported. If I is I^-, SGGS discovers which positive literals need to be true to satisfy S, and dually if I is I^+. While I can be any Herbrand interpretation, in this article I is either I^+ or I^-.

SGGS assumes a notion of *uniform falsity*, whereby 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. The sets of I-true and I-false literals of a clause C are denoted $\mathsf{tlits}(C)$ and $\mathsf{flits}(C)$, respectively. If I is I^+ or I^-, a literal is I-true or I-false based solely on its sign. If I is I^+, $\mathsf{tlits}(C) = C^+$, $\mathsf{flits}(C) = C^-$, positive clauses are I-all-true, negative clauses are I-all-false, and mixed clauses are neither. If I is I^-, $\mathsf{tlits}(C) = C^-$, $\mathsf{flits}(C) = C^+$, negative clauses are I-all-true, positive clauses are I-all-false, and mixed clauses are neither.

SGGS works on a *trail* Γ that is either empty, denoted by ε, or given by a sequence of clauses $A_1 \rhd C_1[L_1], \ldots, A_n \rhd C_n[L_n]$ $(n \geq 1)$, where $A_i \rhd C_i[L_i]$ occurs at index i in Γ, and the notation $C_i[L_i]$ means that literal $L_i \in C_i$ is *selected*. The *length* of Γ, written $|\Gamma|$, is the number of clauses in Γ. The *prefix* of length j of Γ is denoted by $\Gamma|_j$. SGGS requires that every literal in Γ is either I-true or I-false, and an I-false literal is selected if there is one, so that an I-true literal is selected only in an I-all-true clause. If I is based on sign, the first requirement becomes trivial, whereas the second one remains important in the light of how Γ represents a partial interpretation.

A trail Γ induces a *partial interpretation* $I^p(\Gamma)$: if $\Gamma = \varepsilon$, $I^p(\Gamma) = \emptyset$; otherwise, $I^p(\Gamma) = I^p(\Gamma|_{n-1}) \cup pcgi(A_n \rhd L_n, \Gamma)$, where pcgi stands for *proper constrained ground instances*. A pcgi of $A_n \rhd C_n[L_n]$ is a cgi $C[L]$ that is *not* satisfied by $I^p(\Gamma|_{n-1})$ (i.e., $I^p(\Gamma|_{n-1}) \cap C[L] = \emptyset$) and can be satisfied by adding L since $\neg L \notin I^p(\Gamma|_{n-1})$. For the selected literal, $pcgi(A_n \rhd L_n, \Gamma) = \{L : \ C[L] \in pcgi(A_n \rhd C_n[L_n], \Gamma)\}$. $I^p(\Gamma)$ is completed into an *interpretation* $I[\Gamma]$ by consulting I for the truth value of any literal undefined in $I^p(\Gamma)$. I-false literals are preferred for selection to differentiate $I[\Gamma]$ from I as $I \not\models S$.

A clause $C[L]$ is a *conflict clause*, if all its literals are uniformly false in $I[\Gamma]$; it is the *justification* of its selected literal L, if all its literals except L are uniformly false in $I[\Gamma]$, so that L must be true to satisfy $C[L]$. SGGS ensures that every I-all-true clause in Γ is either a conflict clause or a justification, by keeping track of *dependences* among literals. Given $\Gamma = A_1 \rhd C_1[L_1], \ldots, A_n \rhd C_n[L_n]$, a literal $M \in C_j[L_j]$ *depends* on L_k if $k < j$ and the selection of L_k makes M uniformly false in $I[\Gamma]$: all literals in $Gr(A_j \rhd M)$ appear negated in $pcgi(A_k \rhd L_k, \Gamma)$. By definition of trail, either M is I-true and L_k is I-false or vice versa. In the first case, SGGS *assigns* M to $C_k[L_k]$. A non-selected I-true literal must be assigned, while a selected I-true literal is assigned if possible, and if it is, it must be assigned rightmost. An I-all-true clause $C[L]$ in Γ is either a conflict clause (all its literals are assigned and L is assigned rightmost) or the justification of L (all its literals but L are assigned). In the latter case $C[L]$ is in the *disjoint prefix* of Γ, denoted $dp(\Gamma)$, which is the longest prefix such that $pcgi(A \rhd C[L], \Gamma) = Gr(A \rhd C[L])$ for all its clauses $A \rhd C[L]$.

An *SGGS-derivation* is a series of trails $\Gamma_0 \vdash \Gamma_1 \vdash \ldots \Gamma_j \vdash \ldots$, where $\Gamma_0 = \varepsilon$, and $\forall j$, $j{>}0$, an SGGS-inference generates Γ_j from Γ_{j-1} and S. When needed, we use Θ to name an SGGS-derivation. SGGS *makes progress* in two ways. Assume $\bot \notin \Gamma$ (refutation not found) and $I[\Gamma] \not\models S$ (model not found). 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* to generate from C and Γ a clause $A \rhd E$, such that E is an instance of C and $C' \in Gr(A \rhd E)$, in order to extend $I[\Gamma]$ to try to satisfy C'. If $\Gamma \neq dp(\Gamma)$, the trail needs repair: either it contains a conflict clause, or there are other intersections between selected literals. We describe first SGGS-extension and then the inference rules that apply when $\Gamma \neq dp(\Gamma)$.

SGGS-extension unifies n distinct literals $L_1, \ldots, L_n$ in a clause $C \in S$ with as many I-false selected literals of opposite sign $M_1, \ldots, M_n$ in $dp(\Gamma)$. Let α be the simultaneous mgu such that $\forall j$, $1 \leqslant j \leqslant n$, $L_j\alpha = \neg M_j\alpha$. Since the M_j's are

I-false, their instances $M_j\alpha$ are also I-false, and the instances $L_j\alpha$ are I-true. If I is I^+ or I^-, also the L_j's are I-true. In order to ensure that all literals that enter the trail are either I-true or I-false, SGGS-extension requires that all the literals in $(C\setminus\{L_1,\ldots,L_n\})\alpha\beta$ are I-false, where β is another substitution computed for this purpose. If I is I^+ or I^-, β is unnecessary, this requirement implies that $L_1,\ldots,L_n$ are *all* the I-true literals in C, and the *SGGS-extension scheme* (cf. [18, Def. 12]) gets simplified as follows:

Definition 1 (SGGS-Extension scheme) Given input clause set S and trail Γ, if for a clause $C\in S$ and all its I-true literals $L_1,\ldots,L_n$ ($n\geq 0$) there are clauses $B_1\rhd D_1[M_1],\ldots,B_n\rhd D_n[M_n]$ in $dp(\Gamma)$, such that literals $M_1,\ldots,M_n$ are I-false, and $\forall j,\ 1\leqslant j\leqslant n,\ L_j\alpha = \neg M_j\alpha$ 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, assigning literals $L_1\alpha,\ldots,L_n\alpha$ to clauses $D_1,\ldots,D_n$, respectively.

Clause C is the *main premise* and $B_1\rhd D_1[M_1],\ldots,B_n\rhd D_n[M_n]$ are the *side premises*. In order to have side premises for all I-true literals in C, multiple variants of a clause in $dp(\Gamma)$ may be used. The *SGGS-extension rules* instantiate this scheme: they are *conflicting*, if the extension clause $A\rhd E$ is a conflict clause, and *non-conflicting* otherwise, that is, if $A\rhd E$ has *pcgi*'s and therefore extends $I[\Gamma]$. A derivation starts with a non-conflicting SGGS-extension where C is I-all-false, so that α is empty, $n=0$, and $E=C$. All such steps can be done as one and we assume they are. The definition of the SGGS inference rules involves an *SGGS-suitable ordering on ground atoms* that is total and extends the size ordering, so that it is well-founded. While SGGS-extension expands the trail, *SGGS-deletion* contracts it by removing *all* disposable clauses, where $A_n\rhd C_n[L_n]$ is *disposable*, if $I^p(\Gamma|_{n-1})\models A_n\rhd C_n[L_n]$.

We consider next the inference rules that apply when $\Gamma\neq dp(\Gamma)$. The main one is *SGGS-splitting*, which decomposes a clause into instances, possibly introducing constraints, in order to isolate intersections between selected literals. The definition of *splitting* in SGGS builds upon that of *partition*: a *partition* of a clause $A\rhd C[L]$, where A is satisfiable, is a set $\{A_i\rhd C_i[L_i]\}_{i=1}^n$ such that $Gr(A\rhd C) = \bigcup_{i=1}^n\{Gr(A_i\rhd C_i)\}$, the literals $A_i\rhd L_i$ are pairwise disjoint, all A_i's are satisfiable, and the L_i's are chosen consistently with L, that is, each L_i is an instance of L. For instance, adding predicate symbol $\mathsf{Q}\subseteq s_1\times s_2$ to Example 2, a partition of $[\mathsf{P}(\mathsf{f}(x),y)]\vee\mathsf{Q}(x,y)$ is $\{[\mathsf{P}(\mathsf{f}(x),\mathsf{b})]\vee\mathsf{Q}(x,\mathsf{b}),\ [\mathsf{P}(\mathsf{f}(x),\mathsf{f}(\mathsf{a}))]\vee\mathsf{Q}(x,\mathsf{f}(\mathsf{a}))\}$.

A *splitting* of $A\rhd C[L]$ by $B\rhd D[M]$ is a partition $\{A_i\rhd C_i[L_i]\}_{i=1}^n$ of $A\rhd C[L]$ such that there exists a j, $1\leqslant j\leqslant n$, for which $at(Gr(A_j\rhd L_j))$ is the intersection of $A\rhd L$ and $B\rhd M$, and for all i, $1\leqslant i\neq j\leqslant n$, $A_i\rhd L_i$ and $B\rhd M$ are disjoint. $A_j\rhd C_j[L_j]$ is called the *representative* of $B\rhd D[M]$ in the splitting. A partition is *trivial* if it is a singleton, and a splitting is trivial if it yields a trivial partition: for example, splitting a ground clause yields the clause itself, and splitting $A\rhd C[L]$ by $B\rhd D[M]$ when $Gr(A\rhd C[L])\subseteq Gr(B\rhd D[M])$ (i.e., $B\rhd D[M]$ is more general than $A\rhd C[L]$) yields a single clause equivalent to $A\rhd C[L]$ (they have the same cgi's). The *preferred clause* in a partition is

typically the one whose selected literal has the smallest pcgi in the SGGS-suitable ordering (cf. [18, Def. 22]). The preferred clause is listed first when replacing a clause by its splitting according to the *SGGS-splitting scheme*:

Definition 2 (SGGS-Splitting scheme) If a trail Γ contains clauses $A \rhd C[L]$ and $B \rhd D[M]$, such that L and M intersect, *SGGS-splitting* replaces the *split clause* $A \rhd C[L]$ by a splitting of $A \rhd C[L]$ by $B \rhd D[M]$ denoted $\mathit{split}(A \rhd C[L], B \rhd D[M])$ or $\mathit{split}(C, D)$ for short.

Not all clauses in $\mathit{split}(C, D)$ need to be kept for completeness, provided the representative and the preferred clause are kept. There are four *SGGS-splitting rules. Splitting by similar literals* (*s-splitting*) and *splitting by dissimilar literals* (*d-splitting*) apply when $A \rhd C[L]$ occurs at a higher index than $B \rhd D[M]$ in the trail; s-splitting applies when L and M have the same sign, and d-splitting applies otherwise. After an s-splitting, D's representative in $\mathit{split}(C, D)$ is disposable (cf. [18, Lemma 3]) and SGGS-deletion removes it, eliminating the intersection. After a d-splitting, the intersection between L and M is removed by *SGGS-resolution*, that will be presented as part of conflict solving with the other two splitting rules, *SGGS-factoring* and *left splitting*. Since a splitting step replaces the split clause, assignments of I-true literals to clauses may not be preserved. Therefore, clauses with literals assigned to a split clause can be deleted. A splitting inference is *bundled* if it is followed by such deletions.

The following example illustrates some of the inference rules introduced so far, demonstrating how SGGS halts on the input set used to show that hyperresolution cannot decide EPR ([28, Ex. 4.8] and [20, 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^+. With I^-, the SGGS-derivation is as follows:

$$\begin{array}{lll} \Gamma_0 \colon \varepsilon \vdash \Gamma_1 \colon [\mathsf{P}(x,x,\mathsf{a})] & & \mathsf{extend}\ (i) \\ \quad \vdash \Gamma_2 \colon [\mathsf{P}(x,x,\mathsf{a})],\ \mathsf{P}(x,y,\mathsf{a}) \vee [\mathsf{P}(y,x,\mathsf{a})] \vee \neg\mathsf{P}(x,x,\mathsf{a}) & & \mathsf{extend}\ (ii) \\ \quad \vdash \Gamma_3 \colon [\mathsf{P}(x,x,\mathsf{a})],\ \mathsf{P}(x,x,\mathsf{a}) \vee [\mathsf{P}(x,x,\mathsf{a})] \vee \neg\mathsf{P}(x,x,\mathsf{a}), & & \\ \qquad y \neq x \rhd \mathsf{P}(x,y,\mathsf{a}) \vee [\mathsf{P}(y,x,\mathsf{a})] \vee \neg\mathsf{P}(x,x,\mathsf{a}) & & \mathsf{s\text{-}split} \\ \quad \vdash \Gamma_4 \colon [\mathsf{P}(x,x,\mathsf{a})], & & \end{array}$$

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

Since $I^- \not\models P(x, x, \mathsf{a})$, SGGS-extension (abbreviated extend possibly with the identifier of the main premise as argument) puts it on the trail. As $I[\Gamma_1]$ satisfies $P(x, x, \mathsf{a})$, but no other positive literal, $I[\Gamma_1] \not\models (ii)$. Thus, SGGS-extension unifies the third literal of clause (ii) with $[\mathsf{P}(x, x, \mathsf{a})]$ on the trail, producing Γ_2, where an I^--false (i.e., positive) literal is selected in the added clause (choosing the other makes no difference). Since the selected literals in Γ_2 intersect and have the same sign, *s-splitting* (abbreviated s-split) splits the second clause and yields Γ_3. The second clause in Γ_3 is disposable and SGGS-deletion (abbreviated delete) removes it. As $I[\Gamma_4] \models S$, the derivation halts reporting satisfiable (with I^+ SGGS proceeds dually with (iii) and (iv)).

We describe next what happens if Γ contains a conflict clause $A \rhd C$. A conflict is *explained* by *SGGS-resolution* and then *solved*.

Definition 3 (SGGS-Resolution) If Γ contains clauses $B \rhd D[M]$ and $A \rhd C[L]$ such that $B \rhd D[M]$ is I-all-true, is in $dp(\Gamma)$, 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* generates the *SGGS-resolvent* $A \rhd R$, where R is $(C \setminus \{L\}) \cup (D \setminus \{M\})\vartheta$, replaces C by $A \rhd R$, deletes all clauses with literals assigned to C, and assigns every literal $P\vartheta \in \mathsf{tlits}(R)$ to the same clause that $P \in \mathsf{tlits}(C) \cup \mathsf{tlits}(D)$ was assigned to.

SGGS-resolution *explains* the conflict by resolving $A \rhd C[L]$ with the justification $B \rhd D[M]$ of the literal M whose selection makes L uniformly false in $I[\Gamma]$. The resolvent is still a conflict clause that replaces $A \rhd C[L]$. SGGS-extension ensures that every I-false literal in a conflict clause depends on a selected I-true literal in $dp(\Gamma)$ (cf. [18, 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 $H \rhd E[P]$.

The conflict represented by $H \rhd E[P]$ is *solved* by *moving* (inference rule *SGGS-move*) $H \rhd E[P]$ to the left of the clause $A \rhd C[L]$ in $dp(\Gamma)$ to which P is assigned, provided (i) no other literal in E is assigned to C, and (ii) the pcgi's of L are the complements of the cgi's of P (i.e., $\neg Gr(H \rhd P) = pcgi(A \rhd L, \Gamma)$). The effect of an SGGS-move is to *flip* P from being uniformly false in $I[\Gamma]$ to being an implied literal, hence true in $I[\Gamma]$, with justification $H \rhd E[P]$. Condition (ii) makes this flipping precise. If Condition (i) is violated, because E contains another literal Q assigned to clause C, SGGS-move does not apply, because otherwise Q would have nowhere to be assigned after the move.

Prior to SGGS-move, either *SGGS-factoring* or *left splitting* may apply to $H \rhd E[P]$. SGGS-factoring applies if E contains another literal Q that is assigned to clause C and furthermore unifies with P with mgu ϑ. SGGS-factoring replaces E by $split(H \rhd E, (H \rhd E)\vartheta)$, where the *SGGS-factor* $(H \rhd E)\vartheta$ is its own representative. SGGS-factors inherit literal assignments like SGGS-resolvents. If ϑ does not satisfy H, the SGGS-factor is $false \rhd E\vartheta$, $split(H \rhd E, false \rhd E\vartheta)$ is empty, and SGGS-factoring simply removes $H \rhd$

$E[P]$, solving the conflict. Left splitting applies if the pcgi's of L contain more than the complements of the cgi's of P (i.e., $\neg Gr(H \rhd P) \subset pcgi(A \rhd L, \Gamma)$), and SGGS-factoring does not apply, because E does not contain another literal Q as above. Left splitting replaces $A \rhd C[L]$ by $split(A \rhd C[L], H \rhd E[P])$ with P assigned to the representative of E in $split(C, E)$. Left splitting enables SGGS-move, which places E to the left of its representative in $split(C, E)$.

In summary, conflict solving involves an SGGS-move, possibly preceded by left splitting or one or more SGGS-factoring steps. These inferences appear in examples in the sequel. A conflicting SGGS-extension is *bundled*, if it is followed by all applicable explanation and conflict-solving inferences.

The completeness argument for SGGS employs the SGGS-suitable ordering, and a *convergence ordering* $>^c$ on trails with associated equivalence $\approx^c$ ([18, Def. 46]). The *fairness* of an SGGS-derivation involves several properties: an inference is applied whenever $\bot \notin \Gamma$ and $I[\Gamma] \not\models S$; no trivial splitting is applied; SGGS-deletion is applied eagerly; all splitting inferences and all conflicting SGGS-extensions are bundled; and inferences applying to shorter prefixes of the trail are never neglected in favor of others applying to longer prefixes (cf. [18, Defs. 32, 37, and 49]). The *limit* of a fair derivation $\Gamma_0 \vdash \Gamma_1 \vdash \ldots \Gamma_j \vdash \Gamma_{j+1} \vdash \ldots$ 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 $\Gamma_j|_i \approx^c \Gamma_\infty|_i$ ([18, Def. 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 $\Gamma_\infty = \Gamma_k$. These definitions support the following results:

1. *Finiteness of descending chains of length-bounded trails* [18, Thm. 6 and Cor. 2]: A chain $\Gamma_0 >^c \Gamma_1 >^c \ldots \Gamma_j >^c \Gamma_{j+1} \ldots$ where $\forall j$, $j \geq 0$, $|\Gamma_j| \leqslant n$, for some $n \geq 0$, is finite.
2. *Descending chain theorem* [18, Thm. 8]: A fair SGGS-derivation forms a descending chain $\Gamma_0 >^c \Gamma_1 >^c \ldots >^c \Gamma_j >^c \Gamma_{j+1} \ldots$.
3. *Completeness* [18, 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*).

Theorems (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 SGGS and Known Decidable Fragments

In this section we develop the concept of *finite basis* to ensure that the length of trails in a fair SGGS-derivation is upper bounded. It follows that the derivation is guaranteed to terminate. If the input is satisfiable, the cardinality of the finite basis offers an upper bound on the cardinality of the generated model. The decidability of the stratified fragment by SGGS is a straightforward application of this approach. On the other hand, counter-examples show that SGGS with sign-based semantic guidance cannot decide other known decidable

fragments. These counter-example derivations are very useful to understand SGGS. Then we show that if clauses are *ground-preserving*, SGGS terminates whenever hyperresolution does.

3.1 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 4 (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 $\mathcal{H}(\mathcal{B}) = \{t : t \text{ is a strict subterm of } L \text{ for } L \in \mathcal{B}\}$. 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 yields *termination*, *decidability*, and the *small model property*. In the next section we apply this approach to the stratified fragment.

3.2 SGGS Decides the Stratified Fragment

A signature is *stratified*, 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 $1 \leqslant i \leqslant n$ [1,40,53]. 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 $f\colon s_1 \times \cdots \times s_n \to s'$ 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 [48].

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 $\mathsf{g}\colon s_1 \to s_2$ 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 $\mathsf{g}\colon s_2 \to s_1$, so that stratification is lost.

Given a set S of clauses whose signature is stratified, or stratified clause set for short, the Herbrand universe $\mathcal{H}$ and the Herbrand base $\mathcal{A}$ are *finite*, because stratification prevents building terms of unbounded depth. Therefore, it suffices to pick $\mathcal{A}$ itself as finite basis, and Theorem 1, together with the completeness theorems for SGGS, yields the following.

Theorem 3 *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 $\overline{0}_i$, $\overline{1}_i$, and $\overline{x}_i$ be i-tuples of 0's, 1's, and distinct variables $x_1, \ldots, x_i$, respectively. Then S_k consists of the following $k+2$ clauses, for $0 \leqslant m \leqslant k-1$:

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

This set was used in the context of an analysis of first-order theorem-proving strategies [55, 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 [55, 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 [55, 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 [49, 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 [30, 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:

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

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 $\mathsf{resolve}$), unsatisfiability is detected:

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

...	...
$\vdash \Gamma_{2^{k+2}} : [\neg \mathsf{Q}(\overline{0}_k)],\ [\mathsf{Q}(\overline{0}_k)],\ \ldots$	move
$\vdash \Gamma_{2^{k+2}+1} : \bot, \ldots$	resolve

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.

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

In this section we show by counterexamples that SGGS with sign-based semantic guidance does not decide the Ackermann, monadic, and FO^2 fragments. While one non-terminating example would suffice, we consider three clause sets, because they illustrate how SGGS works and two of them are well-known.

The first set is $S_0 = \{\mathsf{P}(x) \vee \mathsf{P}(\mathsf{f}(x)),\ \neg\mathsf{P}(x') \vee \neg\mathsf{P}(\mathsf{f}(x'))\}$ [37, 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: $J_1 = \{\mathsf{P}(\mathsf{f}^{2k}(\mathsf{a})), \neg P(\mathsf{f}^{2k+1}(\mathsf{a})) : k \geqslant 0\}$ and $J_2 = \{\neg P(\mathsf{f}^{2k}(\mathsf{a})), P(\mathsf{f}^{2k+1}(\mathsf{a})) : k \geqslant 0\}$.

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 [37]. 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:

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

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 $[\neg\mathsf{P}(\mathsf{b})]$ 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[1] aiming at reaching model J_1 in the limit.

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

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

where either literal can be selected in the second clause. Selecting $\mathsf{P}(\mathsf{f}(x))$ avoids an intersection with $\mathsf{P}(\mathsf{a})$, 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. Here and in the sequel $top(x) \neq \mathsf{a} \rhd \varphi[x]$ is abbreviated $\varphi[\mathsf{f}(x)]$ where x is a new variable.

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

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 $\alpha = \{x' \leftarrow \mathsf{a},\ x \leftarrow \mathsf{a}\}$):

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

The added clause is a conflict clause with $\neg\mathsf{P}(\mathsf{a})$ assigned to the first clause, and $\neg\mathsf{P}(\mathsf{f}(\mathsf{a}))$ to the second one, so that $\neg\mathsf{P}(\mathsf{f}(\mathsf{a}))$ is selected, because the selected literal in an I-all-true clause, if selected, must be assigned rightmost. Since $\mathsf{P}(\mathsf{f}(\mathsf{a}))$ is less general than $\mathsf{P}(\mathsf{f}(x))$ ($\neg Gr(\neg P(\mathsf{f}(\mathsf{a}))) \subset pcgi(\mathsf{P}(\mathsf{f}(x)), \Gamma)$), the

[1] The SGGS-derivation with I^- given for this set in [19, Ex. 11] is incorrect.

third clause splits the second one by left splitting (abbreviated l-split), which enables SGGS-move followed by SGGS-resolution:

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

The infinite derivation lists as selected the literals $\mathsf{P}(\mathsf{a}),\ \neg\mathsf{P}(\mathsf{f}(\mathsf{a})),\ \mathsf{P}(\mathsf{f}^2(\mathsf{a})),\ldots$ 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

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

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:

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

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 $\neg Gr(\neg P(x)) = pcgi(\mathsf{P}(x), \Gamma)$. 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:

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

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 $\alpha = \{x' \leftarrow x\}$:

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

where the extension clause has the first literal assigned to the first clause, and the second literal assigned to the second clause and hence selected. SGGS-move solves the conflict:

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

Now the selected literals of the second and third clauses intersect and have opposite sign, so that d-splitting (abbreviated d-split) pulls out the intersection and allows SGGS-resolution to remove it:

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

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

$$\vdash \mathit{top}(x) \neq \mathsf{f} \rhd [\mathsf{P}(x)] \vee \mathsf{P}(\mathsf{f}(x)),\ \mathit{top}(x) \neq \mathsf{f} \rhd \neg\mathsf{P}(x) \vee [\neg\mathsf{P}(\mathsf{f}(x))],$$
$$\mathit{top}(x) \neq \mathsf{f} \rhd \neg\mathsf{P}(x) \vee [\mathsf{P}(\mathsf{f}^2(x))],$$
$$\mathit{top}(x) \neq \mathsf{f} \rhd [\mathsf{P}(\mathsf{f}^2(x))] \vee \mathsf{P}(\mathsf{f}^3(x)),\ [\mathsf{P}(\mathsf{f}^3(x))] \vee \mathsf{P}(\mathsf{f}^4(x)) \qquad \textsf{s-split}$$
$$\vdash \mathit{top}(x) \neq \mathsf{f} \rhd [\mathsf{P}(x)] \vee \mathsf{P}(\mathsf{f}(x)),\ \mathit{top}(x) \neq \mathsf{f} \rhd \neg\mathsf{P}(x) \vee [\neg\mathsf{P}(\mathsf{f}(x))],$$
$$\mathit{top}(x) \neq \mathsf{f} \rhd \neg\mathsf{P}(x) \vee [\mathsf{P}(\mathsf{f}^2(x))],\ [\mathsf{P}(\mathsf{f}^3(x))] \vee \mathsf{P}(\mathsf{f}^4(x)) \qquad \textsf{delete}$$
$$\vdash \ldots$$

Since $\mathit{top}(x) \neq \mathsf{f}$ 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:

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

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

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

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

$$\vdash top(x) \neq \mathsf{f} \rhd \mathsf{P}(x) \vee [\mathsf{P}(\mathsf{f}(x))],$$
$$top(x) \neq \mathsf{f} \rhd \neg\mathsf{P}(\mathsf{f}(x)) \vee [\neg\mathsf{P}(\mathsf{f}^2(x))],$$
$$top(x) \neq \mathsf{f} \rhd \mathsf{P}(\mathsf{f}(x)) \vee [\mathsf{P}(\mathsf{f}^2(x))],\ \mathsf{P}(\mathsf{f}^2(x)) \vee [\mathsf{P}(\mathsf{f}^3(x))] \qquad \text{d-split}$$

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

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

$$\vdash top(x) \neq \mathsf{f} \rhd \mathsf{P}(x) \vee [\mathsf{P}(\mathsf{f}(x))],$$
$$top(x) \neq \mathsf{f} \rhd \neg\mathsf{P}(\mathsf{f}(x)) \vee [\neg\mathsf{P}(\mathsf{f}^2(x))],\ \mathsf{P}(\mathsf{f}^2(x)) \vee [\mathsf{P}(\mathsf{f}^3(x))] \qquad \text{delete}$$

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:

$$\vdash top(x) \neq \mathsf{f} \rhd \mathsf{P}(x) \vee [\mathsf{P}(\mathsf{f}(x))],$$
$$top(x) \neq \mathsf{f} \rhd \neg\mathsf{P}(\mathsf{f}(x)) \vee [\neg\mathsf{P}(\mathsf{f}^2(x))],\ \mathsf{P}(\mathsf{f}^2(x)) \vee [\mathsf{P}(\mathsf{f}^3(x))],$$
$$top(x) \neq \mathsf{f} \rhd \neg\mathsf{P}(\mathsf{f}^3(x)) \vee [\neg\mathsf{P}(\mathsf{f}^4(x))] \qquad \text{extend}$$

$$\vdash top(x) \neq \mathsf{f} \rhd \mathsf{P}(x) \vee [\mathsf{P}(\mathsf{f}(x))],$$
$$top(x) \neq \mathsf{f} \rhd \neg\mathsf{P}(\mathsf{f}(x)) \vee [\neg\mathsf{P}(\mathsf{f}^2(x))],$$
$$top(x) \neq \mathsf{f} \rhd \mathsf{P}(\mathsf{f}^2(x)) \vee [\mathsf{P}(\mathsf{f}^3(x))],\ \mathsf{P}(\mathsf{f}^3(x)) \vee [\mathsf{P}(\mathsf{f}^4(x))] \qquad \text{l-split}$$

$$\vdash \ldots$$

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 $J_2 = \{\neg\mathsf{P}(\mathsf{a}),\ \mathsf{P}(\mathsf{f}(\mathsf{a})),\ \neg\mathsf{P}(\mathsf{f}^2(\mathsf{a})),\ \mathsf{P}(\mathsf{f}^3(\mathsf{a}))\ldots\}$ 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.

3.4 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. Let the input problem S be a set of *guarded clauses*, or *guarded set* for short. 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 [23]. Although formulæ in the guarded fragment have no function symbols, guarded clauses may contain function symbols introduced by Skolemization. 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 9 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:

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

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

Example 10 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^+:

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

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 11 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:

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

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:

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

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

$\vdash [\neg \mathsf{P}(\mathsf{a})] \vee \neg \mathsf{P}(\mathsf{f}(\mathsf{a})),\ [\neg \mathsf{P}(\mathsf{f}(x))] \vee \neg \mathsf{P}(\mathsf{f}^2(x)),\ [\mathsf{P}(\mathsf{a})]$ l-split

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

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

$\vdash [\mathsf{P}(\mathsf{a})],\ [\neg \mathsf{P}(\mathsf{f}(\mathsf{a}))],\ [\neg \mathsf{P}(\mathsf{f}(\mathsf{a}))] \vee \neg \mathsf{P}(\mathsf{f}^2(\mathsf{a})),$
$[\neg \mathsf{P}(\mathsf{f}^2(x))] \vee \neg \mathsf{P}(\mathsf{f}^3(x))$ s-split

$\vdash [\mathsf{P}(\mathsf{a})],\ [\neg \mathsf{P}(\mathsf{f}(\mathsf{a}))],\ [\neg \mathsf{P}(\mathsf{f}^2(x))] \vee \neg \mathsf{P}(\mathsf{f}^3(x))$ delete

$\vdash [\mathsf{P}(\mathsf{a})],\ [\neg \mathsf{P}(\mathsf{f}(\mathsf{a}))],\ [\neg \mathsf{P}(\mathsf{f}^2(x))] \vee \neg \mathsf{P}(\mathsf{f}^3(x)),\ [\neg \mathsf{P}(\mathsf{a})] \vee \mathsf{P}(\mathsf{f}^2(\mathsf{a}))$ extend

$\vdash [\mathsf{P}(\mathsf{a})],\ [\neg \mathsf{P}(\mathsf{f}(\mathsf{a}))],\ [\neg \mathsf{P}(\mathsf{f}^2(x))] \vee \neg \mathsf{P}(\mathsf{f}^3(x)),\ [\mathsf{P}(\mathsf{f}^2(\mathsf{a}))]$ resolve

$\vdash [\mathsf{P}(\mathsf{a})],\ [\neg \mathsf{P}(\mathsf{f}(\mathsf{a}))],\ [\neg \mathsf{P}(\mathsf{f}^2(\mathsf{a}))] \vee \neg \mathsf{P}(\mathsf{f}^3(\mathsf{a})),$
$[\neg \mathsf{P}(\mathsf{f}^3(x))] \vee \neg \mathsf{P}(\mathsf{f}^4(x)),\ [\mathsf{P}(\mathsf{f}^2(\mathsf{a}))]$ l-split

$\vdash [\mathsf{P}(\mathsf{a})],\ [\neg \mathsf{P}(\mathsf{f}(\mathsf{a}))],\ [\mathsf{P}(\mathsf{f}^2(\mathsf{a}))],\ [\neg \mathsf{P}(\mathsf{f}^2(\mathsf{a}))] \vee \neg \mathsf{P}(\mathsf{f}^3(\mathsf{a})),$
$[\neg \mathsf{P}(\mathsf{f}^3(x))] \vee \neg \mathsf{P}(\mathsf{f}^4(x))$ move

$\vdash [\mathsf{P}(\mathsf{a})],\ [\neg \mathsf{P}(\mathsf{f}(\mathsf{a}))],\ [\mathsf{P}(\mathsf{f}^2(\mathsf{a}))],\ [\neg \mathsf{P}(\mathsf{f}^3(\mathsf{a}))],$
$[\neg \mathsf{P}(\mathsf{f}^3(x))] \vee \neg \mathsf{P}(\mathsf{f}^4(x))$ resolve

$\vdash \ldots$

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.

3.5 SGGS Decides Ground-Preserving Sets Decided by Hyperresolution

SGGS with I^- or I^+ as initial interpretation and hyperresolution share sign-based semantic guidance. In this section 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. We omit SGGS-constraints because they are immaterial.

Definition 5 (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. The binary counter clauses of Example 5 are both positively and negatively ground-preserving. Guarded clauses are positively ground-preserving, because 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. 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 with all signs exchanged). 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. Induction step: 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. Induction step: 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 4 *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 4 hold for SGGS-derivations with I^+ and negative hyperresolution. Since the *positive variable dominated* (PVD) [27,20] and *bounded depth increase* (BDI) [44] fragments are positively ground-preserving, and positive hyperresolution decides them, so does SGGS.[2]

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.*

The proof argument of this section does not generalize to the case where the input is not ground-preserving. Lemma 3 can be lifted, showing that for every clause C on the trail during the derivation, there exist a positive clause $C' \in R_H^*(S)$ and a substitution σ such that $C^+ \subseteq C'\sigma$ (see Appendix A). Since a non-ground hyperresolvent C' can be in this relation with infinitely many trail clauses, Theorem 4 does not get lifted.

4 SGGS Decides Three New Fragments of First-Order Logic

In this section we discover three new decidable fragments of first-order logic, by showing that SGGS decides them. The first one is the *restrained* fragment, which combines ground-preservingness 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.

4.1 SGGS Decides the Restrained Fragments

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

Example 12 Consider the positively ground-preserving set S with clauses:

$$\mathsf{P}(\mathsf{s}^{10}(\mathsf{0}), \mathsf{s}^{9}(\mathsf{0})) \quad (i), \quad \neg\mathsf{P}(\mathsf{s}(\mathsf{s}(x)), y) \vee \mathsf{P}(x, \mathsf{s}(y)) \quad (ii), \quad \neg\mathsf{P}(\mathsf{s}(\mathsf{0}), \mathsf{0}) \quad (iii).$$

Let I^- be the initial interpretation. SGGS-extension puts $\mathsf{P}(\mathsf{10}, \mathsf{9})$ on the trail, abbreviating $\mathsf{s}^n(\mathsf{0})$ as n. This stirs a series of SGGS-extensions aiming at adding

[2] For PVD also the finite basis approach applies implying the small model property [19].

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 6 (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. In the sequel, $\succeq$ is a restraining quasi-ordering.

Definition 7 (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 12 is strictly positively restrained. The next example clarifies why a *quasi*-ordering is used.

Example 13 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 [61] 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 8 (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 $\mathcal{A}_S^{\preceq} = \{L : L \in \mathcal{A},\ \exists M \in \mathcal{A}_S \text{ such that } M \succeq L\}$.

In words, $\mathcal{A}_S^{\preceq}$ contains all the ground atoms upper bounded by those occurring in clauses in S. By Conditions (ii) and (iii) in Definition 6, $\mathcal{A}_S^{\preceq}$ 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 $\mathcal{A}_S^{\preceq}$.*

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 $\mathcal{A}_S^{\preceq}$. 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 $\mathcal{A}_S^{\preceq}$. 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 $\mathcal{A}_S^{\preceq}$. We have to show that $at(C\alpha) \subseteq \mathcal{A}_S^{\preceq}$. For the negative literals $\neg L_1\alpha, \ldots, \neg L_n\alpha$ we have $L_i\alpha = M_i\alpha = M_i \in \mathcal{A}_S^{\preceq}$. Let L be a literal in C^+. If L is ground, then $L\alpha = L \in \mathcal{A}_S \subseteq \mathcal{A}_S^{\preceq}$. 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, $1 \leqslant i \leqslant n$, $M_i \in \mathcal{A}_S^{\preceq}$ and $M_i = M_i\alpha = L_i\alpha \succeq L\alpha$, we have $L\alpha \in \mathcal{A}_S^{\preceq}$.

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

Theorem 5 *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 14 The clause set of Example 12 is a subset of the following satisfiable clause set S from problem PUZ054-1 in TPTP:

$$\mathsf{P}(\mathsf{s}^{10}(\mathsf{0}), \mathsf{s}^{9}(\mathsf{0})), \quad \neg\mathsf{P}(\mathsf{s}(\mathsf{s}(x)), y) \vee \mathsf{P}(x, \mathsf{s}(y)), \quad \neg\mathsf{P}(x, \mathsf{s}(\mathsf{s}(y))) \vee \mathsf{P}(x, \mathsf{s}(y)),$$
$$\neg\mathsf{P}(\mathsf{s}(\mathsf{0}), \mathsf{0}), \quad \neg\mathsf{P}(\mathsf{s}(x), \mathsf{s}(y)) \vee \mathsf{P}(\mathsf{s}(x), y).$$

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.

4.2 Sign-Based Resolution Strategies Decide the Restrained Fragments

We consider PO-resolution and the positively restrained fragment. The findings 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 $L \in \mathcal{A}^{\preceq}_S$ 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 7 and 8. The induction hypothesis is that all clauses in $R^k_>(S)$ satisfy the claim. For the inductive step, 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 $at(L' \vee D) \subseteq \mathcal{A}^{\preceq}_S$ (†), which means $L' \vee D$ is ground, $(L' \vee D)\sigma = L' \vee D$, and $at(D\sigma) \subseteq \mathcal{A}^{\preceq}_S$. For the positive literals in $C\sigma$, let $Q\sigma$ be one of them, so $Q \in C^+$. By induction hypothesis, either (i) $Q \in \mathcal{A}^{\preceq}_S$, 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 $Q\sigma \in \mathcal{A}^{\preceq}_S$. 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 (†), $L' \in \mathcal{A}^{\preceq}_S$, L' is ground, and $L'\sigma = L'$. Since $L' \in \mathcal{A}^{\preceq}_S$ and $L' = L'\sigma = L\sigma \succeq Q\sigma$, it follows that $Q\sigma \in \mathcal{A}^{\preceq}_S$ by Definition 8.

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

Theorem 6 *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 [34]. 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 $\mathcal{A}^{\preceq}_S$. Since $\mathcal{A}^{\preceq}_S$ is finite, and repeated literals in ground clauses disappear by merging, only finitely many clauses can be built from $\mathcal{A}^{\preceq}_S$, which contradicts $R^*_>(S)$ being infinite.

These results[3] 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 6 is trivial, since only positive clauses get generated.

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

Thus, Theorem 5 follows also from Theorem 4 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 15 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, $0 \leqslant i \leqslant n$:

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

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 $\sum_{k=1}^{n+1} n^k$.

4.3 Sort-Refined Versions of the Restrained and PVD Fragments

As observed for the stratified fragment, where all sorts are acyclic, such sorts are non-problematic for termination. In this section we consider a signature with both cyclic and acyclic sorts, and we apply the restrictions of the restrained and PVD [27] fragments only to literals involving cyclic sorts. In this manner, we obtain two new fragments named *sort-restrained* and *sort-refined-PVD*, which generalize stratified and restrained, and stratified and PVD, respectively. We show that sort-restrained and sort-refined-PVD clause sets admit finite bases, so that SGGS decides both new fragments. 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.

[3] Lemma 5 and Theorem 6 were proved for ordered resolution assuming $>$ ensures that $L' \vee D$ is positive [19, Lem. 5 and Thm. 6]; it is better to work with PO-resolution.

Definition 9 (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.

A crucial observation is that 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.

Definition 10 (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.

This property will be an ingredient of the definition of both new fragments.

4.3.1 SGGS Decides the Sort-Restrained Fragments

The next example illustrates the idea of the sort-restrained fragment.

Example 16 Consider the following set S of clauses with sorts $\{s_1, s_2\}$:

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

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:

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

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

The following definition captures this observation. Let $\succeq$ be a restraining quasi-ordering with the subterm property: $c[t] \succeq t$ for all contexts c.

Definition 11 (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 16 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 12 (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 in S such that $Gr(L)$ is finite, and $\mathcal{L}^{\preceq}_{\Sigma}$ the smallest instantiation-closed and $\preceq$-closed superset of $\mathcal{L}^{\downarrow}_{\Sigma}$. Then the basis for S is $\mathcal{A}^{\preceq}_{S,\Sigma} = Gr(\mathcal{L}^{\preceq}_{\Sigma})$.

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

Example 17 For the clause set of Example 16 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 $\mathcal{L}^{\downarrow}_{\Sigma}$, 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 all the variants of these atoms. Then $\mathcal{A}^{\preceq}_{S,\Sigma}$ 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 12, 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 7 *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.*

4.3.2 SGGS Decides the Sort-Refined-PVD Class

We recall the PVD property [27] 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 13 (PVD fragment) A set S of clauses is in the PVD fragment if every clause $C \in S$ is positively ground-preserving and $\forall x \in Var(C^+)$ 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 18 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:

$$\circlearrowright s_1 \longrightarrow s_4 \longleftarrow s_2 \longleftarrow s_3$$

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:

$$\mathsf{P}(\mathsf{f}(\mathsf{a}), y) \quad (i) \qquad \neg\mathsf{P}(x, y) \vee \mathsf{P}(x, \mathsf{g}(z)) \quad (ii)$$
$$\neg\mathsf{P}(\mathsf{f}(x), y) \vee \mathsf{Q}(\mathsf{h}(x, y)) \quad (iii) \qquad \neg\mathsf{P}(x, z) \vee \neg\mathsf{P}(y, z) \vee \mathsf{R}(x, y) \quad (iv)$$

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 14 (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 $x \in Var(C^+)$ of infinite domain it holds that $\text{depth}_x(C^+) \leqslant \text{depth}_x(C^-)$. A set of clauses is *sort-refined-PVD* if all its clauses are.

The set of Example 18 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 $d = max\{\text{depth}(L) : \ L \text{ is an atom in clause } C \text{ and } C \in S\}$.

Definition 15 (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 15 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 19 In Example 18 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 $\neg L_1, \ldots, \neg L_n$ in C with as many I^--false (i.e., positive) selected literals $M_1, \ldots, M_n$ in Γ. For all i, $1 \leqslant i \leqslant n$, $M_i \in \mathcal{L}^d_{S,\Sigma}$ 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 14, 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 15.
- 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 14, 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 15 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 8 *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.*

5 A Sufficient Condition for Modularity of SGGS Termination

In this article, modularity of termination means that if SGGS terminates in each of two fragments $\mathcal{F}_1$ and $\mathcal{F}_2$, then it terminates given a clause set $S = S_1 \uplus S_2$ for sets S_1 in $\mathcal{F}_1$ and S_2 in $\mathcal{F}_2$. Let $\mathcal{P}_j$ be the set of predicate symbols occurring in S_j and $\mathcal{A}_j$ the Herbrand base of S_j, for $j \in \{1, 2\}$.

Theorem 9 *Given a set S of clauses such that $S = S_1 \uplus S_2$, $\mathcal{P}_1 \cap \mathcal{P}_2 = \emptyset$, and all fair SGGS-derivations with initial interpretation I and input set S_j are in a finite basis $\mathcal{B}_j$, for $j \in \{1, 2\}$, every fair SGGS-derivation with initial interpretation I and input set S is in the finite basis $\mathcal{B}_1 \uplus \mathcal{B}_2$.*

Proof Since $\mathcal{P}_1 \cap \mathcal{P}_2 = \emptyset$ and $\mathcal{B}_j \subseteq \mathcal{A}_j$, it follows that $\mathcal{B}_1 \cap \mathcal{B}_2 = \emptyset$, so that we can write $\mathcal{B}_1 \uplus \mathcal{B}_2$ instead of $\mathcal{B}_1 \cup \mathcal{B}_2$. All derivations in this proof use I as initial interpretation. Let Θ be a fair SGGS-derivation from S. We will prove that for all trails Γ in Θ, there exist two derivations Θ_j from S_j, for $j \in \{1, 2\}$, ending with a trail Γ^j, such that every clause in Γ occurs in exactly one of Γ^1 or Γ^2. Since by hypothesis all derivations from S_j are in the finite basis $\mathcal{B}_j$ for $j \in \{1, 2\}$, it follows that Θ is in $\mathcal{B}_1 \uplus \mathcal{B}_2$.
The proof is by induction on the length k of derivation Θ. In the base case ($k = 0$), Θ is empty, and therefore the claim is vacuously true (e.g., Θ_1 and Θ_2 are also empty). The induction hypothesis is that for all derivations Θ of length k with last trail Γ, there are derivations Θ_j from S_j with last trails Γ^j, for $j \in \{1, 2\}$, such that every clause C in Γ appears in exactly one of Γ^1 or Γ^2. Since $\mathcal{P}_1 \cap \mathcal{P}_2 = \emptyset$ and Θ_j is in $\mathcal{B}_j$ for $j \in \{1, 2\}$, clause C shows up in Γ^1 if and only if for all its literals L it is $top(at(L)) \in \mathcal{P}_1$ and C shows up in Γ^2 if and only if for all its literals L it is $top(at(L)) \in \mathcal{P}_2$. Therefore, it is sufficient to determine that for one literal L in C it is $top(at(L)) \in \mathcal{P}_1$ to conclude that C belongs to Γ^1, and the same for $\mathcal{P}_2$ and Γ^2 (†). Let $\Gamma \vdash \Gamma'$ be the $(k{+}1)$-th step in derivation Θ. We distinguish three cases based on whether this step is an SGGS-extension, an SGGS-resolution, or an SGGS-splitting.

1. $\Gamma \vdash \Gamma'$ is an SGGS-extension with main premise $C \in S$, side premises $B_1 \rhd D_1[M_1], \ldots, B_n \rhd D_n[M_n]$, and extension clause $E = \bigwedge_{i=1}^{n} B_i\alpha \rhd C\alpha$, for α the simultaneous mgu of literals $L_1, \ldots, L_n$ in C with $M_1, \ldots, M_n$. Suppose $C \in S_1$ (the case for S_2 is symmetric). It follows that for all i, $1 \leqslant i \leqslant n$, $top(at(L_i)) \in \mathcal{P}_1$. By the unification condition of the extension step it follows that for all i, $1 \leqslant i \leqslant n$, $top(at(M_i)) \in \mathcal{P}_1$. By induction hypothesis, for all i, $1 \leqslant i \leqslant n$, $B_i \rhd D_i[M_i]$ occurs either in Γ^1 or in Γ^2. Since for all i, $1 \leqslant i \leqslant n$, $top(at(M_i)) \in \mathcal{P}_1$, it follows by (†) that for all i, $1 \leqslant i \leqslant n$, $B_i \rhd D_i[M_i]$ is in Γ^1. Thus, Θ_1 can be extended with the same SGGS-extension generating a trail containing E.
2. $\Gamma \vdash \Gamma'$ is an SGGS-resolution step with parents $A \rhd C[L]$ and $B \rhd D[M]$ such that $L = \neg M\vartheta$ and resolvent $R = (C \setminus \{L\}) \cup (D \setminus \{M\})\vartheta$. By the unification condition of the resolution step, $top(at(L)) = top(at(M))$ and either $top(at(M)) \in \mathcal{P}_1$ or $top(at(M)) \in \mathcal{P}_2$ but not both. Suppose that $top(at(M)) \in \mathcal{P}_1$ (the other case is symmetric). By induction hypothesis,

$A \rhd C[L]$ occurs either in Γ^1 or in Γ^2, and the same holds for $B \rhd D[M]$. Since $top(at(L)) = top(at(M)) \in \mathcal{P}_1$, both $A \rhd C[L]$ and $B \rhd D[M]$ are in Γ^1 by (†). Thus, the same SGGS-resolution step can be applied to Θ_1 generating a trail containing R.

3. $\Gamma \vdash \Gamma'$ is an SGGS-splitting step that replaces a clause C by $split(C, D)$, for $A \rhd C[L]$ and $B \rhd D[M]$ in Γ. This happens only if the literals L and M intersect, which implies that $top(at(L)) = top(at(M))$. Suppose that $top(at(M)) \in \mathcal{P}_1$ (again the other case is symmetric). By induction hypothesis, each premise is either in Γ^1 or in Γ^2. Since $top(at(L)) = top(at(M)) \in \mathcal{P}_1$, both premises are in Γ^1 by (†). Hence Θ_1 can proceed with the same splitting step generating a trail where C is replaced by $split(C, D)$.

The following example exhibits a set of clauses which does not belong to the fragments of Section 4, but fits in Theorem 9.

Example 20 Consider the set S consisting of the following clauses over a signature with one sort:

$$\neg\mathsf{P}(\mathsf{s}(x)) \vee \mathsf{P}(\mathsf{f}(\mathsf{f}(x))) \quad (i) \qquad \neg\mathsf{P}(\mathsf{s}^{10}(\mathsf{0})) \quad (ii)$$
$$\mathsf{Q}(\mathsf{s}(\mathsf{0}), \mathsf{0}) \vee \mathsf{Q}(\mathsf{0}, \mathsf{s}(\mathsf{0})) \quad (iii) \qquad \neg\mathsf{Q}(x, y) \vee \neg\mathsf{Q}(y, z) \vee \mathsf{Q}(x, z) \quad (iv).$$

Set S is positively ground-preserving, but it is not restrained, because clause (iv), which expresses the transitivity of the Q relation, cannot satisfy the ordering-based condition for restrainedness (see Definition 7). Also, S is not PVD because clause (i) does not satisfy the depth condition for PVD (see Definition 13). Since the signature has only one sort with infinite domain, set S is neither sort-restrained nor sort-refined-PVD. However, assuming a precedence $\succ_p$ where $\mathsf{s} \succ_p \mathsf{f}$, any LPO can be used to show that the subset $S_1 = \{(i),\ (ii)\}$ is positively restrained. The complement subset $S_2 = \{(iii),\ (iv)\}$ is PVD. Since SGGS (with I^-) is guaranteed to halt in these two fragments by the finite basis property, and the partition $S = S_1 \uplus S_2$ satisfies the condition $\mathcal{P}_1 \cap \mathcal{P}_2 = \emptyset$, SGGS (with I^-) is guaranteed to halt also given set S.

The next example shows that in the presence of a shared predicate it is not possible to form a finite basis by taking the union of finite bases.

Example 21 Consider the singletons $S_1 = \{(i)\ \neg\mathsf{P}(\mathsf{f}(x), y) \vee \mathsf{P}(x, \mathsf{f}(y))\}$ and $S_2 = \{(ii)\ \mathsf{P}(x, \mathsf{a})\}$. Set S_1 is both positively and negatively ground-preserving. It can be shown positively restrained by an LPO with precedence $\succ_p$ such that $\mathsf{P} \succ_p \mathsf{f}$. Since no ground atom occur in S_1, its finite basis $\mathcal{B}_1$ is the empty set (see Definition 8). Set S_2 is EPR with finite basis $\mathcal{B}_2 = \{\mathsf{P}(\mathsf{a}, \mathsf{a})\}$. SGGS terminates trivially if given either S_1 or S_2. However, S_1 and S_2 share the predicate symbol P. If the input set is $S_1 \uplus S_2$, SGGS with I^- yields an infinite derivation that generates the infinite series $\{\neg\mathsf{P}(\mathsf{f}(x), \mathsf{f}^n(\mathsf{a})) \vee \mathsf{P}(x, \mathsf{f}^{n+1}(\mathsf{a}))\}$:

$$\begin{aligned} \varepsilon &\vdash [\mathsf{P}(x, \mathsf{a})] && \textsf{extend } (ii) \\ &\vdash [\mathsf{P}(x, \mathsf{a})],\ \neg\mathsf{P}(\mathsf{f}(x), \mathsf{a}) \vee [\mathsf{P}(x, \mathsf{f}(\mathsf{a}))] && \textsf{extend } (i) \end{aligned}$$

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

Given two clausal fragments $\mathcal{F}_1$ and $\mathcal{F}_2$, let the *disjoint union fragment* $\mathcal{F}_1 \uplus \mathcal{F}_2$ be defined as the fragment of all clause sets $S = S_1 \uplus S_2$ for clause sets S_1 in $\mathcal{F}_1$ and S_2 in $\mathcal{F}_2$, such that $\mathcal{P}_1 \cap \mathcal{P}_2 = \emptyset$. The next corollary follows from Theorems 1 and 9.

Corollary 6 *If SGGS is guaranteed to halt in $\mathcal{F}_1$ and in $\mathcal{F}_2$ by the finite basis property, SGGS is also guaranteed to halt in $\mathcal{F}_1 \uplus \mathcal{F}_2$.*

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 TTT2 [41] and AProVE [32] 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 [64]. In this manner it is possible 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 66% of the decidable problems without equality* in TPTP 7.4.0. Then, we present Koala, the *first SGGS-based theorem prover*. We tested Koala on all the problems without equality in TPTP 7.4.0. We analyze these experiments, reporting statistics, and evaluating the performances of our prototype in the SGGS-decidable classes, in the other decidable classes, and on the semidecidable problems.

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 6 and 7). 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.

In order to have a quasi-ordering whose equivalence is not necessarily identity, one needs to allow also for equations, extracting 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 16 (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 the permutative equations, such as $\mathsf{differ}(x, y) \simeq \mathsf{differ}(y, x)$ in Example 13. For Example 14, 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 literal $L \in C^+$. By Definition 16, there exists a rule $t \rightarrow u$ 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 $\rightarrow_{\mathcal{R}_S}$, and hence $\rightarrow_{\mathcal{R}_S/\mathcal{E}_S}$, 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 $\rightarrow^*_{\mathcal{R}_S/\mathcal{E}_S}$ is a restraining quasi-ordering. Indeed, $\rightarrow^*_{\mathcal{R}_S/\mathcal{E}_S}$ is stable, its strict part $\rightarrow^+_{\mathcal{R}_S/\mathcal{E}_S}$ 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 $\rightarrow^*_{\mathcal{R}_S}$. Indeed, $\rightarrow^*_{\mathcal{R}_S}$ is stable, its strict part $\rightarrow^+_{\mathcal{R}_S}$ is well-founded by hypothesis (i), and the equivalence classes of $\rightarrow^*_{\mathcal{R}_S} \cap {}_{\mathcal{R}_S}{}^*\leftarrow$ are finite, because $\rightarrow^*_{\mathcal{R}_S} \cap {}_{\mathcal{R}_S}{}^*\leftarrow$ is identity.

Problem HWV036-2 (cf. Example 1) is a set of axioms with no ground atoms, which is combined with sets of ground clauses in several other TPTP problems. For example, HWV008-2.002 adds 23 ground clauses. As part of the experiments reported in Section 6.2, we found a terminating positively restraining rewrite system for HWV008-2.002, so that this problem, as well as HWV036-2, is strictly positively restrained.

6.2 Classifying Decidable Problems for the Experiments

The TPTP 7.4.0 library has over 17,000 first-order problems, 5,000 of which do not have equality. 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 system. Then the tool invokes TTT2 and

ground: 82	EPR: 1,059	stratified: 1,260	Ackermann: 102
monadic: 750	FO^2: 204	guarded: 569	PVD: 347
restrained: 413	sort-restrained: 1,398	sort-refined-PVD: 1,304	

Table 1 Number of problems decidable according to different criteria

AProVE to determine whether at least a candidate is a restraining rewrite systems satisfying the termination conditions for a restrained set.

However, both the number of candidate rewrite systems and their size grow exponentially with the number of literals in the clause set. Therefore, first, 1,539 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. Second, for each clause set, TT_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. For example, stratified input problems are recognized by computing the sort dependency graph and testing it for acyclicity. This test is applied also to identify sort-restrained and sort-refined-PVD problems.

Table 1 shows how many of the remaining 3,461 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 3,461 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 66% 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.

problem class	# problems	SAT	UNSAT	avg. time
ground	82	11	68	0.74
EPR	1,059	203	503	20.41
stratified	1,260	256	635	16.27
monadic	750	56	209	0.32
FO^2	204	214	360	6.30
Ackermann	102	14	79	0.63
guarded	569	117	212	7.22
PVD	347	70	228	7.50
sort-refined-PVD	1,304	259	663	15.74
restrained	413	65	311	1.32
sort-restrained	1,361	270	706	14.91
other problems	1,313	106	247	6.73
all problems	4,010	630	1,095	12.79

Table 2 Outcomes of the Koala derivations

The average TPTP rating of the problems in the new decidable classes is low,[4] which means that most provers can solve them. However, 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. 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 efficiently the substitutions required by SGGS-extension. Koala computes the sort dependency graph, because it facilitates testing sorted constraints for satisfiability. Thus, Koala can also detect stratified problems on its own. The *SGGS-suitable ordering* is a KBO with a fixed precedence, $w_0 = 1$, and weight 1 for all non-variable symbols in order to extend the size ordering. Koala sorts by this ordering the clauses in a splitting, according to the notion of preferred clause. The *search plans* in Koala are *fair*, so that all derivations are fair.

In the experiments the initial interpretation was I^- by default, and I^+ only for positively ground-preserving problems. The time-out was 300 sec of wall-clock time. Table 2 reports how many problems Koala showed satisfiable or unsatisfiable along with 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

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

problem class	# steps	# ext	# conflicts	# gen	# del	max $\|\Gamma\|$
ground	345	117	141	245	99	8
EPR	496	250	154	399	183	106
stratified	402	204	123	323	147	89
monadic	120	43	46	85	32	9
FO^2	143	75	40	113	35	46
Ackermann	295	100	120	209	84	7
guarded	506	210	187	388	182	27
PVD	553	228	206	425	201	26
restrained	129	53	46	96	41	19
sort-restrained	371	189	114	299	136	84
sort-refined-PVD	389	198	119	313	142	87
other problems	67	48	8	56	20	46
all problems	270	143	77	219	96	74

Table 3 Statistics about the Koala derivations

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.

Table 3 displays some statistics for the successful Koala derivations. The first three columns report time-related statistics: *average derivation length*, *average number of SGGS-extensions*, and *average number of conflicts*. 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 second three columns in Table 3 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, clauses deleted because they have literals assigned to split clauses, and clauses deleted as part of SGGS-resolution steps.

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. It is interesting that even though Koala performs on average several hundred inference steps, the maximum trail length remains typically well below one hundred.[5]

Table 4 compares Koala with three state-of-the-art automated theorem provers in terms of number of problems solved in some of the new decidable classes, distinguishing between satisfiable and unsatisfiable problems. At least on these classes, the SGGS prototype is not too far from these state-of-the-art provers, that are the result of decades of work.

[5] The interested reader can find more details about the experiments at http://profs.scienze.univr.it/~bonacina/sggsdp/ and Koala at https://github.com/bytekid/koala.

problem class	# sets	Koala SAT	Koala UNSAT	E 2.4 SAT	E 2.4 UNSAT	Vampire 4.4 SAT	Vampire 4.4 UNSAT	iProver 3.5 SAT	iProver 3.5 UNSAT
restrained	413	65	311	56	317	66	329	68	321
sort-restrained	1,398	270	706	150	766	278	1,007	337	977

Table 4 Koala and three state-of-the-art ATP's on some of the new decidable classes

7 Discussion

This section covers related work, summary of contributions, and directions for future research.

7.1 Related Work

The already known decidable fragments of FOL considered in this paper were the object of much research. Several methods decide EPR which is popular for applications (e.g., [54,3,30]). The stratified fragment generalizes EPR to the many-sorted setting. It was introduced as the first of three fragments of increasing expressivity towards capturing Alloy specifications [1], and it found application in verification [53,48]. Ordered resolution decides the Ackermann and monadic fragments [37,28], while FO^2 can be reduced to the Gödel fragment [62,33] which is also decided by ordered resolution [37,28]. Ordered resolution decides also the guarded fragment [23]. Hyperresolution does not decide these fragments, except for subfragments of the guarded fragment [35].

The decidable fragments discovered in this paper are new, and therefore other theorem-proving methods may not have been applied to them. For example, Inst-Gen [39] is instance-based like SGGS, and the model evolution calculus (MEC) [9] lifts to first-order logic the DPLL procedure for propositional satisfiability [21]. While both methods decide the stratified fragment [40], this is not the case for the restrained fragments. If Inst-Gen picks an unfortunate literal selection for the restrained clause set of Example 14, it does not halt. SGGS avoids this phenomenon thanks to semantic guidance. Since MEC starts with the all-positive interpretation as candidate model, it may not terminate on satisfiable negatively restrained sets such as the one in Example 13. Indeed, E-Darwin [7] does not halt on this example that Koala solves in a few seconds.

Several fragments studied in this article restrict clauses to be *ground-preserving*. Positively ground-preserving clauses are also termed *range-restricted* [51,46,20,8]. Manthey and Bry introduced the name "range-restricted" [46] and at the same conference (CADE 1988) Kounalis and Rusinowitch introduced the name "ground-preserving," meaning negatively ground-preserving [42,43]. Thus, the two names are equally old in automated deduction. We chose ground-preserving (already in [16]), because hyperresolution generates only ground clauses from a ground-preserving set.

Ground-preserving clauses were introduced in Horn logic [46,42,43]. In Horn logic, positive unit clauses are called *facts*, negative clauses are called

queries, and mixed clauses are called *rules*. When reasoning *forward* or *bottom-up*, that is, from the facts (e.g., by positive hyperresolution) variables that appear in the positive literals, 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. Positively ground-preserving clauses were introduced to prevent this phenomenon [46].

Dually, when reasoning *backward* or *top-down*, that is, from a query (e.g., by negative hyperresolution) variables that appear in the negative literals, but not in the positive literals of a rule, get introduced in the backward chaining process, so that one gets non-ground queries even if one starts from a ground query. Negatively ground-preserving clauses were introduced to prevent this phenomenon in Horn theories with equality [42,43]. The purpose was to obtain linear input proofs where all center clauses are ground, and furthermore decreasing in the complete simplification ordering used to control ordered resolution and superposition (see also [12, 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.

Positive ground-preservingness was used to show that DPLL(Γ+$\mathcal{T}$), where Γ is an inference system including hyperresolution, superposition with negative selection, and simplification, decides *essentially finite* (only one monadic function f with finite range) and positively ground-preserving axiomatizations, provided that *speculative axioms* $f^j(x) \simeq f^k(x)$ (j>k) are tried for increasing values of j and k [16]. Simplification applies the speculative axioms to limit the depth of generated terms.

Example 22 Reconsider the set of clauses in Example 11:

$$\mathsf{P}(\mathsf{a}) \qquad \neg\mathsf{P}(x) \vee \mathsf{P}(\mathsf{f}(\mathsf{f}(x))) \qquad \neg\mathsf{P}(x) \vee \neg\mathsf{P}(\mathsf{f}(x)).$$

Since it admits a finite model (see Example 11) it is essentially finite. DPLL(Γ+$\mathcal{T}$) 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, as shown in Example 11.

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* [8]. First-order splitting [67,58] 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 [8] 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 can be considered speculative inferences in the sense of

[16]. Thanks to these enhancements, positive hyperresolution can decide the Bernays-Schönfinkel class with equality [8].

The finite basis approach to termination has been applied in the context of satisfiability modulo theories (SMT) (e.g., [6,22,14]). Here we applied it to SGGS which is a theorem-proving method for first-order logic, but it is conflict-driven and can be considered a generalization of the CDCL procedure that is at the heart of SAT and SMT solvers.

7.2 Summary of Contributions

In this article we investigated the SGGS theorem-proving method as a decision procedure for satisfiability in fragments of first-order logic without equality. SGGS is an attractive candidate for this purpose, because it is *semantically-guided*, *conflict-driven*, and it builds *models*, which seems a fairly rare combination of properties for a first-order method. In the description of SGGS and in most of the results, we assumed that the initial interpretation guiding SGGS is either all-negative or all-positive. This assumption simplifies the method, and one might say that SGGS with any Herbrand interpretation as initial interpretation stands to SGGS with sign-based guidance like semantic resolution stands to hyperresolution. Since SGGS is refutationally complete and model complete in the limit, what is needed to obtain decision procedures is *termination*.

A way to prove that a reasoning method terminates is to show that it only generates terms coming from a *finite basis*. 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 the ground instances of all the clauses that appear on the trail during the derivation are made of atoms from the finite basis. Such a derivation is guaranteed to terminate, and the cardinality of the finite basis yields an upper bound on the cardinality of the generated model, whenever the input set is satisfiable. Therefore, if all input sets from a certain fragment admit a finite basis, the fragment is *SGGS-decidable* and has the *small model property*. The first application of this approach is that SGGS decides the *stratified fragment*, where the Herbrand base itself is finite.

Given the connection represented by sign-based semantic guidance, another approach is to show that if hyperresolution terminates so does SGGS. We showed that if the input is positively ground-preserving, SGGS with all-negative semantic guidance generates only ground clauses, and it is guaranteed to terminate if positive hyperresolution does. The dual results hold for the negative case. It follows that SGGS decides the ground-preserving fragments decided by hyperresolution, such as PVD [27,20] and BDI [44]. SGGS with all-negative semantic guidance can be seen as another forward-reasoning or bottom-up method, and SGGS with all-positive semantic guidance can be seen as another backward-reasoning or top-down method.

We completed our investigation of the behavior of SGGS on known decidable fragments by giving counterexamples showing that SGGS with sign-

based semantic guidance does not decide the Ackermann, monadic, FO^2, and guarded fragments. Then, we applied SGGS and the finite basis approach to discover *new* decidable fragments, by showing that SGGS decides them. The *positively/negatively restrained*, *positively/negatively sort-restrained*, and *sort-refined-PVD* fragments are introduced in this manner. The *positively/negatively restrained fragments* assume positive/negative ground-preservingness, but do not involve limits on literal depth like PVD and BDI do. Rather, they impose an ordering-based condition using a quasi-ordering, so that also clauses stating symmetries can be restrained as in Example 13.

The *positively/negatively sort-restrained* fragments combine restrainedness with stratification, imposing restrainedness only to the part of the problem that is not stratified. Similarly, the *sort-refined-PVD* fragment combines PVD with stratification, imposing the PVD restrictions only to the part of the problem that is not stratified. With these fragments we showed how to combine decidability criteria to find more decidable fragments. Since SGGS decides the basic fragments (e.g., stratified and restrained), the SGGS-decidability of the sort-restrained fragments is a combination result.

Note, however, that the combination does not happen at the clausal level, as a sort-restrained set is not given by the union of a restrained set and a stratified set. Rather, the combination happens at the finer level of literals, as the restrained conditions are applied to literals involving sorts that are cyclic and therefore violate stratification. The same consideration applies to the sort-refined-PVD fragment. Thus, we investigated also the *modularity of termination* in a more generic sense. We gave a modularity theorem identifying sufficient conditions for the termination of SGGS when the input is given by the union of two sets for which the termination of SGGS is known.

For the experimental part of this work, we showed that it is possible to determine whether a set of clauses is restrained, and hence sort-restrained, by finding an associated rewrite system whose rewrite relation is terminating. More precisely, since restrainedness involves a quasi-ordering, one has to work with a pair formed by a rewrite system and a set of equations, and the rewrite relation is rewriting modulo. Thanks to this reduction, SGGS-decidability becomes another field of application for termination tools such as $\mathsf{TTT2}$ [41] and AProVE [32], that were used in our experiments to find restrained and sort-restrained problems in the TPTP library [64]. The other decidability criteria (e.g., stratification, PVD) can also be tested automatically, resulting in a classification of TPTP problems without equality.

We concluded with a presentation of the Koala prototype prover, which is the first implementation of SGGS. We applied Koala to all the problems without equality in TPTP, and reported statistics and performances. For example, the statistics indicate that the SGGS trail never grows very long, unlike the sets of clauses generated and kept by resolution-based strategies that work by saturation. This confirms the intuition that the SGGS trail reminds one of an accordeon, that grows when SGGS-extension expands the model and shrinks when the other SGGS inference rules fix the model. Although Koala is only a prototype, the experiments show potential.

7.3 Directions for Future Work

Given the novelty of this research, there are many directions for future work. A most important objective is to endow SGGS with equality reasoning. This may require to extend the SGGS approach to model representation [17] in order to represent only Herbrand interpretations that are models of the axioms of equality. The SGGS inference system may have to be enriched with inference rules for equality, such as superposition. Ideally, SGGS-superposition should be restricted similarly to SGGS-resolution. However, it appears that SGGS-superposition may have to be applied more liberally, because equality reasoning may require more than instance generation and conflict solving. Inference rules that build equality into the inference system allows it to generate equalities from equalities, building-in the transititivity and substitutivity, or congruence, axioms. Incorporating reflexivity and symmetry is usually simpler.

Another possibility is to attack first simpler instances of the problem of extending SGGS with equality. For example, one may consider only ground equalities and study how to interface SGGS with a *congruence closure* algorithm (e.g., [50,26,5,52]). The above-mentioned *blocking* technique may be a candidate for integration in SGGS. Congruence closure and blocking were used to import some equational reasoning in tableaux-based methods [66]. An interesting point could be whether there is any relation between the constraints generated by blocking and SGGS-constraints.

Forms of speculative inferences [16] other than blocking could be considered also for SGGS. Speculative inferences that cause conflicts are undone by backtracking [16,66,8]. 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., [16,57]). One can envision integrating SGGS in the CDSAT framework for *conflict-driven satisfiability* modulo a union of theories and an initial assignment [14,15]. In this context, SGGS could be seen as a CDSAT module for first-order logic, leaving equality reasoning to CDSAT modules for other theories. As a result, SGGS would be endowed with some equality reasoning and CDSAT with some quantifier reasoning.

Another lead is to consider initial interpretations not based on sign. An initial interpretation is *goal-sensitive*, if it satisfies all the input clauses except the goal clauses, that is, those generated by transforming in clausal form 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 [18]. A challenging question is whether this form of goal-sensitivity may be useful when the problem includes a large knowledge base.

The experiments with Koala allow us to identify critical issues for the performance of an SGGS prover. For example, instance generation by SGGS-extension is a bottleneck for problems with many input clauses, and forms

of *caching* should be considered to avoid repeating computations. Since no method excels at all problems, further development of Koala and more experiments may contribute to discover classes of first-order problems where the conflict-driven style of SGGS reasoning is especially rewarding.

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 (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, *LNAI*, vol. 9322, pp. 69–84. Springer (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. 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, *LNAI*, vol. 4246, pp. 512–526. Springer (2006). DOI 10.1007/11916277_35
7. 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
8. 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
9. 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
10. Bernays, P., Schönfinkel, M.: Zum Entscheidungsproblem der mathematischen Logik. Mathematische Annalen **99**, 342–372 (1928). DOI 10.1007/BF01459101
11. Bonacina, M.P.: On conflict-driven reasoning. In: N. Shankar, B. Dutertre (eds.) Proceedings of the Sixth Workshop on Automated Formal Methods (AFM) May 2017, *Kalpa Publications*, vol. 5, pp. 31–49. EasyChair (2018). DOI 10.29007/spwm
12. Bonacina, M.P., Dershowitz, N.: Canonical ground Horn theories. In: A. Voronkov, C. Weidenbach (eds.) Programming Logics: Essays in Memory of H. Ganzinger, *LNCS*, vol. 7797, pp. 35–71. Springer (2013). DOI 10.1007/978-3-642-37651-1_3
13. 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, *LNCS*, vol. 9200, pp. 181–204. Springer (2015). DOI 10.1007/978-3-319-23165-5_8
14. 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
15. Bonacina, M.P., Graham-Lengrand, S., Shankar, N.: Conflict-driven satisfiability for theory combination: lemmas, modules, and proofs. J. Autom. Reason. **in press**, 1–49 (2021). DOI 10.1007/s10817-021-09606-y. Published online 12 September 2021
16. 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

17. 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
18. 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
19. Bonacina, M.P., Winkler, S.: SGGS decision procedures. In: N. Peltier, V. Sofronie-Stokkermans (eds.) Proceedings of IJCAR-10, *LNAI*, vol. 12166, pp. 356–374. Springer (2020). DOI 10.1007/978-3-030-51074-9_20
20. Caferra, R., Leitsch, A., Peltier, N.: Automated Model Building. Kluwer (2004). DOI 10.1007/978-1-4020-2653-9
21. Davis, M., Logemann, G., Loveland, D.: A machine program for theorem-proving. C. ACM **5**(7), 394–397 (1962). DOI 10.1145/368273.368557
22. de Moura, L., Jovanović, D.: A model-constructing satisfiability calculus. In: R. Giacobazzi, J. Berdine, I. Mastroeni (eds.) Proceedings of VMCAI-14, *LNCS*, vol. 7737, pp. 1–12. Springer (2013). DOI 10.1007/978-3-642-35873-9_1
23. 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
24. Dershowitz, N.: Orderings for term-rewriting systems. Theoret. Comput. Sci. **17**(3), 279–301 (1982). DOI 10.1016/0304-3975(82)90026-3
25. 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
26. 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
27. Fermüller, C.G., Leitsch, A.: Model building by resolution. In: E. Börger, G. Jäger, H. Kleine Büning, S. Martini (eds.) Proceedings of CSL-6, *LNCS*, vol. 702, pp. 134–148 (1993). DOI 10.1007/3-540-56992-8_10
28. Fermüller, C.G., Leitsch, A., Hustadt, U., Tammet, T.: Resolution decision procedures. In: Handbook of Automated Reasoning, pp. 1791–1849. Elsevier and MIT Press (2001). DOI 10.1016/b978-044450813-3/50027-8
29. Fermüller, C.G., Salzer, G.: Ordered paramodulation and resolution as decision procedure. In: A. Voronkov (ed.) Proceedings of LPAR-4, *LNAI*, vol. 698, pp. 122–133. Springer (1993). DOI 10.1007/3-540-56944-8_47
30. Fiori, A., Weidenbach, C.: SCL clause learning from simple models. In: P. Fontaine (ed.) Proceedings of CADE-27, *LNAI*, vol. 11716, pp. 233–249 (2019). DOI 10.1007/978-3-030-29436-6_14
31. Fontaine, P.: Combinations of theories for decidable fragments of first-order logic. In: S. Ghilardi, R. Sebastiani (eds.) Proceedings of FroCoS-7, *LNAI*, vol. 5749, pp. 263–278. Springer (2009). DOI 10.1007/978-3-642-04222-5_16
32. 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
33. 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
34. 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
35. Hustadt, U., Schmidt, R.A., Georgieva, L.: Hyperresolution for guarded formulae. J. Symb. Comput. **36**(1-2), 163–192 (2003). DOI 10.1016/S0747-7171(03)00034-8
36. 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
37. Joyner Jr., W.H.: Resolution strategies as decision procedures. J. ACM **23**(3), 398–417 (1976). DOI 10.1145/321958.321960
38. 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

39. 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, *LNCS*, vol. 7797, pp. 239–270. Springer (2013). DOI 10.1007/978-3-642-37651-1_10
40. Korovin, K.: Non-cyclic sorts for first-order satisfiability. In: P. Fontaine, C. Ringeissen, R.A. Schmidt (eds.) Proceedings of FroCoS-9, *LNAI*, vol. 8152, pp. 214–228 (2013). DOI 10.1007/978-3-642-40885-4_15
41. Korp, M., Sternagel, C., Zankl, H., Middeldorp, A.: Tyrolean Termination Tool 2. In: R. Treinen (ed.) Proceedings of RTA-20, *LNCS*, vol. 5595, pp. 295–304 (2009). DOI 10.1007/978-3-642-02348-4_21
42. Kounalis, E., Rusinowitch, M.: On word problems in Horn theories. In: E. Lusk, R. Overbeek (eds.) Proceedings of CADE-9, *LNCS*, vol. 310, pp. 527–537. Springer (1988). DOI 10.1007/BFb0012854
43. 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
44. Lamotte-Schubert, M., Weidenbach, C.: BDI: a new decidable clause class. J. Log. Comput. **27**(2), 441–468 (2017). DOI 10.1093/logcom/exu074
45. 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, *LNAI*, vol. 4790, pp. 348–362. Springer (2007). DOI 10.1007/978-3-540-75560-9_26
46. Manthey, R., Bry, F.: SATCHMO: a theorem prover implemented in Prolog. In: E. Lusk, R. Overbeek (eds.) Proceedings of CADE-9, *LNCS*, vol. 310, pp. 415–434. Springer (1988). DOI 10.1007/BFb0012847
47. 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
48. McMillan, K.L.: Developing distributed protocols with Ivy. Slides from http://vmcaischool19.tecnico.ulisboa.pt/ (2019)
49. Navarro, J.A., Voronkov, A.: Proof systems for effectively propositional logic. In: A. Armando, P. Baumgartner, G. Dowek (eds.) Proceedings of IJCAR-4, *LNAI*, vol. 5195, pp. 426–440 (2008). DOI 10.1007/978-3-540-71070-7_36
50. Nelson, G., Oppen, D.C.: Fast decision procedures based on congruence closure. J. ACM **27**(2), 356–364 (1980). DOI 10.1145/322186.322198
51. Nicolas, J.M.: Logic for improving integrity checking in relational databases. Acta Infor. **18**(3), 227–253 (1982). DOI 10.1145/322186.322198
52. Nieuwenhuis, R., Oliveras, A.: Fast congruence closure and extensions. Inf. Comput. **205**(4), 557–580 (2007). DOI 10.1016/j.ic.2006.08.009
53. 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
54. 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
55. Plaisted, D.A., Zhu, Y.: The Efficiency of Theorem Proving Strategies. Friedr. Vieweg & Sohn (1997)
56. 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
57. Reger, G., Suda, M., Voronkov, A.: Playing with AVATAR. In: A.P. Felty, A. Middeldorp (eds.) Proceedings of CADE-25, *LNAI*, vol. 9195, pp. 399–415. Springer (2015). DOI 10.1007/978-3-319-21401-6_28
58. Riazanov, A.: Implementing an efficient theorem prover. Ph.D. thesis, Department of Computer Science, The University of Manchester (2003)
59. Robinson, J.A.: Automatic deduction with hyper-resolution. Int. J. of Computer Mathematics **1**, 227–234 (1965). DOI 10.2307/2272384
60. Robinson, J.A.: A machine oriented logic based on the resolution principle. J. ACM **12**(1), 23–41 (1965). DOI 10.1145/321250.321253
61. Rubio, A.: A fully syntactic AC-RPO. Inf. Comput. **178**(2), 515–533 (2002). DOI 10.1006/inco.2002.3158

62. Scott, D.: A decision method for validity of sentences in two variables. J. Symb. Log. **27**, 377–377 (1962)
63. Slagle, J.R.: Automatic theorem proving with renamable and semantic resolution. J. ACM **14**(4), 687–697 (1967). DOI 10.1145/321420.321428
64. 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
65. 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
66. Waldmann, U., Schmidt, R.A.: Modal tableau systems with blocking and congruence closure. In: H. de Nivelle (ed.) Proceedings of TABLEAUX-24, *LNAI*, vol. 9323, pp. 38–53. Springer (2015). DOI 10.1007/978-3-319-24312-2_4
67. 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

A The Lifting of Lemma 3

In this appendix we show that Lemma 3 can be lifted to the general case, without the hypothesis that input clauses are ground-preserving. As in the main text we consider positive hyperresolution and SGGS with I^- as initial interpretation.

We begin with a simple observation about simultaneous mgu's: if there exists a simultaneous mgu using as side premises, or satellites, instances of clauses, then there exists also a simultaneous mgu using as side premises, or satellites, the clauses themselves.

Lemma 10 *Given a clause C containing negative literals $\neg L_1, \ldots \neg L_n$, clauses $D_1, \ldots, D_n$ each containing a positive literal M_i, and instances $D_1\sigma_1, \ldots, D_n\sigma_n$, if there exists a simultaneous mgu α such that for all i, $1 \leq i \leq n$, $L_i\alpha = M_i\sigma_i\alpha$, then there exists also a simultaneous mgu τ such that for all i, $1 \leq i \leq n$, $L_i\tau = M_i\tau$.*

Proof For all i and j, $1 \leq i \neq j \leq n$, $\mathcal{V}ar(M_i) \cap \mathcal{V}ar(M_j) = \emptyset$, because M_i and M_j come from different clauses and each clause has its own variables. Then the simultaneous mgu τ is defined as follows: for all $x \in \mathcal{V}ar(C)$, $x\tau = x\alpha$, and for all i, $1 \leq i \leq n$, for all $x \in \mathcal{V}ar(M_i)$, $x\tau = x\sigma_i\alpha$.

Thus, if SGGS-extension applies to main premise C and side premises $D_1\sigma_1, \ldots, D_n\sigma_n$, it applies also to main premise C and side premises $D_1, \ldots, D_n$. Similarly, if positive hyperresolution applies to nucleus C and satellites $D_1\sigma_1, \ldots, D_n\sigma_n$, it applies also to nucleus C and satelllites $D_1, \ldots, D_n$. Note that one or more substitutions σ_i may be empty.

Let D be a trail clause such that $\mathsf{flits}(D) \neq \emptyset$ (i.e., D^+ is not empty). Clause D is a *positive descendant* of an extension clause E in an SGGS-derivation with I^-, if E is added to the trail by SGGS-extension at some stage k, $k > 0$, clause D is added to the trail at some stage $k + j$, $j \geqslant 0$, and there exists a substitution σ such that $D^+ \subseteq E^+\sigma$. Clearly, since $\mathsf{flits}(D) \neq \emptyset$, also $\mathsf{flits}(E) \neq \emptyset$. Note that an E such that $\mathsf{flits}(E) \neq \emptyset$ is a positive descendant of itself, because σ may be empty and j may be 0.

Lemma 11 *For all clauses D that appear on the trail during a fair SGGS-derivation with I^- as initial interpretation, if $\mathsf{flits}(D) \neq \emptyset$ (i.e., D^+ is not empty), D is a positive descendant of an extension clause.*

Proof The proof is by induction on the stage k such that the step $\Gamma_k \vdash \Gamma_{k+1}$ adds D to the trail (i.e., D appears in Γ_{k+1}). Base case: $k = 0$, and D is an I^--all-false (i.e., positive) input clause added by the first SGGS-extension that yields Γ_1. Then D is a positive descendant of itself. Induction hypothesis: for all j, $0 \leq j < k$, for all clauses D added by the step $\Gamma_j \vdash \Gamma_{j+1}$, the claim holds. Induction step: let D be a clause added to the trail by the step $\Gamma_k \vdash \Gamma_{k+1}$. If D is produced by an SGGS-extension, then D is a positive descendant of itself. If D is produced by an SGGS-splitting rule applied to split clause C, then D is

an instance of C, that is, $D = C\tau$ for some substitution τ. It follows that $D^+ = C^+\tau$. If $\mathsf{flits}(D) \neq \emptyset$ then $\mathsf{flits}(C) \neq \emptyset$. Since C was added to the trail by a step $\Gamma_j \vdash \Gamma_{j+1}$ for some j, $0 \leq j < k$, by induction hypothesis C is a positive descendant of some extension clause E: that is, $C^+ \subseteq E^+\sigma$ for some substitution σ. It follows that D also is a positive descendant of E, as $D^+ = C^+\tau \subseteq E^+\sigma\tau$. Otherwise, D is produced by an SGGS-resolution step with a non-I^--all-true (i.e., nonnegative) parent $C_1[L_1]$, where L_1 is I^--false (i.e., positive), an I^--all-true (i.e., negative) parent $C_2[\neg L_2]$, and an mgu ϑ such that $L_1 = L_2\vartheta$. That is, the mgu is a matching substitution instantiating only the variables in the I^--all-true parent. This means that D is the SGGS-resolvent $(C_1 \setminus \{L_1\}) \cup (C_2 \setminus \{\neg L_2\})\vartheta$. Since C_2 is negative and L_1 is positive, it follows that $D^+ = {C_1}^+ \setminus \{L_1\}$, and hence $D^+ \subset {C_1}^+$. Since L_1 is I^--false, $\mathsf{flits}(C_1) \neq \emptyset$. Since C_1 was added to the trail by a step $\Gamma_j \vdash \Gamma_{j+1}$ for some j, $0 \leq j < k$, by induction hypothesis C_1 is a positive descendant of some extension clause E: that is, ${C_1}^+ \subseteq E^+\sigma$ for some substitution σ. Then we have $D^+ \subset {C_1}^+ \subseteq E^+\sigma$, and D also is a positive descendant of E.

Lemma 12 *For all fair SGGS-derivations with I^- from input set S, for all clauses $C \in S$, for all instances $C\alpha$ added to the trail by SGGS-extension, if $\mathsf{flits}(C\alpha) \neq \emptyset$, there exists a positive clause $C' \in R_H^*(S)$ such that $C^+\alpha \subseteq C'$.*

Proof The proof is by induction on the stage k such that the step $\Gamma_k \vdash \Gamma_{k+1}$ adds $C\alpha$ to the trail (i.e., $C\alpha$ appears in Γ_{k+1}). Base case: $k = 0$, and $C\alpha = C$ is an I^--all-false (i.e., positive) input clause added by the first SGGS-extension with empty mgu α. Then $C' = C \in S \subseteq R_H^*(S)$. Induction hypothesis: for all j, $0 \leq j < k$, if $\Gamma_j \vdash \Gamma_{j+1}$ is an SGGS-extension with extension clause $C\alpha$ and $\mathsf{flits}(C\alpha) \neq \emptyset$, there exists a positive clause $C' \in R_H^*(S)$ such that $C^+\alpha \subseteq C'$. Induction step: let $\Gamma_k \vdash \Gamma_{k+1}$ be an SGGS-extension step with extension clause $C\alpha$ such that $\mathsf{flits}(C\alpha) \neq \emptyset$, and side premises $D_1[M_1], \ldots D_n[M_n]$. The substitution α is the simultaneous mgu of all the I^--true (i.e., negative) literals $\neg L_1, \ldots, \neg L_n$ of C with the selected literals $M_1, \ldots, M_n$ (i.e., for all i, $1 \leq i \leq n$, $L_i\alpha = M_i\alpha$). Since the literals $M_1, \ldots, M_n$ are I^--false (i.e., positive), $\mathsf{flits}(D_i) \neq \emptyset$ for all i, $1 \leq i \leq n$. Thus, by Lemma 11, the side premises are positive descendants of extension clauses: for all i, $1 \leq i \leq n$, there exist extension clause E_i and substitution σ_i such that ${D_i}^+ \subseteq {E_i}^+\sigma_i$. This implies that M_i is a literal in ${E_i}^+\sigma_i$. In other words, there exists a positive literal Q_i in E_i such that $M_i = Q_i\sigma_i$. This also means that $\mathsf{flits}(E_i) \neq \emptyset$. Since for all i, $1 \leq i \leq n$, E_i was the extension clause of an SGGS-extension step $\Gamma_j \vdash \Gamma_{j+1}$ for some j, $0 \leq j < k$, and $\mathsf{flits}(E_i) \neq \emptyset$, by induction hypothesis, for all i, $1 \leq i \leq n$, there exists a positive clause ${C_i}' \in R_H^*(S)$ such that ${E_i}^+ \subseteq {C_i}'$. It follows that Q_i is a literal of ${C_i}'$. By Lemma 10, the existence of the simultaneous mgu α such that for all i, $1 \leq i \leq n$, $L_i\alpha = M_i\alpha = Q_i\sigma_i\alpha$ implies the existence of a simultaneous mgu τ such that for all i, $1 \leq i \leq n$, $L_i\tau = Q_i\tau$. This means that positive hyperresolution applies to nucleus C and satellites ${C_1}', \ldots, {C_n}'$. Note that the conditions ${D_i}^+ \subseteq {E_i}^+\sigma_i$ and ${E_i}^+ \subseteq {C_i}'$ imply ${D_i}^+ \subseteq {C_i}'\sigma_i$. In other words, the positive literals of the side premises of the SGGS-extension step are included in instances of the satellites of the positive hyperresolution step. We show that this hyperresolution step generates a positive hyperresolvent C' such that $C^+\alpha \subseteq C'$. Indeed, $C = \neg L_1 \vee \ldots \vee \neg L_n \vee C^+$, and for all i, $1 \leq i \leq n$, ${C_i}' = Q_i \vee {C_i}''$, where ${C_i}''$ is the disjunction of all other literals in ${C_i}'$. The hyperresolvent is $C' = (C^+ \vee {C_1}'' \vee \ldots \vee {C_n}'')\tau$. Since for all $x \in \mathcal{V}ar(C)$, it is $x\tau = x\alpha$ (see the proof of Lemma 10), $C' = C^+\alpha \vee ({C_1}'' \vee \ldots \vee {C_n}'')\tau$ and $C^+\alpha \subseteq C'$ holds.

The generalization of Lemma 3 follows from Lemmas 11 and 12.

Lemma 13 *For all clauses D that appear on the trail during a fair SGGS-derivation with I^-, if $\mathsf{flits}(D) \neq \emptyset$ (i.e., D^+ is not empty), there exist a positive clause $C' \in R_H^*(S)$ and a substitution σ such that $D^+ \subseteq C'\sigma$.*

Proof By Lemma 11 there exists an extension clause E such that $D^+ \subseteq E^+\sigma$ and $\mathsf{flits}(E) \neq \emptyset$. By Lemma 12 there exists a positive clause $C' \in R_H^*(S)$ such that $E^+ \subseteq C'$. It follows that $D^+ \subseteq E^+\sigma \subseteq C'\sigma$.