Generic theorem proving for decision procedures (Technical Report RR 41/2006)

Maria Paola Bonacina Mnacho Echenim

March 2007

Introduction

Most state-of-the-art verification tools rely on built-in satisfiability procedures for specific theories. These satisfiability procedures can be quite complicated to design and combine, and significant effort is devoted to proving them correct and complete, and implementing them. A new approach to defining satisfiability procedures was introduced in [ARR03], where the authors showed that a sound and complete first-order theorem-proving strategy can be used to solve satisfiability problems for several theories of data structures. The idea behind this approach is that since such a strategy is a semi-decision procedure for first-order validity, if one proves that it *terminates* on a presentation of the theory of interest $\mathrm{Th}(\mathcal{T})$ and any set of ground literals, then it is a decision procedure for $\mathrm{Th}(\mathcal{T})$-*satisfiability*. In [ARR03], this idea was applied to a standard inference system, the superposition calculus $\mathcal{SP}$, and several theories, including those of *arrays* and *possibly cyclic lists*.

Since most verification problems involve more than one theory, a significant advantage of an approach based on generic reasoning is that it makes it conceptually simple to combine theories, by considering the union of their presentations. Along with several experimental results that showed the practicality of the rewrite-based approach, the authors of [ABRS05] defined the notion of *variable-inactive theories*. The variable-inactivity condition guarantees that $\mathcal{SP}$ terminates on a combination of theories, provided it terminates on each individual theory. The authors showed that an $\mathcal{SP}$-based strategy is a satisfiability procedure for any combination of the theories of [ARR03] and those they considered. Furthermore, contrary to the common expectation that a generic theorem-prover would be outperformed by more specialized systems such as CVC ([BDS00, SBD04]) or CVC Lite ([BB04]), the experimental results of [ABRS05] showed that this is not the case, and that such procedures are very efficient on several problems.

Several of the theories for which $\mathcal{SP}$ has been shown to yield satisfiability procedures involve *lists*. The superposition calculus yields satisfiability procedures for the theories of lists *à la* Shostak and *à la* Nelson and Oppen (see [ARR03]), and for the theory of lists with nil (see [ABRS06]). A theory of lists that was not yet considered is that of *acyclic lists*, where formulae such as $\mathrm{car}(x) \simeq x$ are unsatisfiable. This theory, along with that of *integer offsets* considered in [NO03, ABRS05], belong to the general class of *theories of recursive data structures*, that we denote $\mathcal{RDS}$. Each member of this class is denoted $\mathcal{RDS}_k$, where k represents the number of *selectors* in the theory. We shall see that the theory of integer offsets is $\mathcal{RDS}_1$, and the theory of acyclic lists is $\mathcal{RDS}_2$. In this paper, we investigate how a rewrite-based inference system can be used to solve any $\mathcal{RDS}_k$-satisfiability problem, for any k.

We also investigate how to generalize the rewrite-based approach to $\mathcal{T}$*-decision problems*, or deciding $\mathcal{T}$-satisfiability of quantifier-free formulae. Of course, a $\mathcal{T}$-satisfiability procedure could be applied after reduction to disjunctive normal form, but this approach is not practical. Another method would be to investigate how to integrate rewrite-based $\mathcal{T}$-satisfiability procedures with a SAT solver, as done for example in [dMRS02, BDS02] for $\mathcal{T}$-satisfiability procedures based on congruence closure. Here, we choose instead to

study the problem of whether rewrite-based theorem-proving strategies can be themselves $\mathcal{T}$-decision procedures.

The contributions of the paper are the following.

- Concerning Recursive Data Structures:
  - Every theory in the class $\mathcal{RDS}$ is presented by an infinite set of axioms, which cannot be given as an input to a theorem prover. Here, we present a reduction that conquers this infinite presentation problem.
  - We prove that for any fair search plan, the inference system terminates on any reduced $\mathcal{RDS}_k$-satisfiability problem.
  - We show that for every k, the theory $\mathcal{RDS}_k$ can be combined with all the theories considered in [ARR03, ABRS05, ABRS06], namely those of lists *à la* Shostak and *à la* Nelson and Oppen, arrays and records with or without extensionality, sets with extensionality, possibly empty lists and integer offsets modulo.
- Concerning $\mathcal{T}$-decision procedures:
  - We introduce the notion of subterm-inactivity and prove that if a theory $\mathcal{T}$ is subterm-inactive, a fair $\mathcal{SP}$-based strategy is a decision procedure for the $\mathcal{T}$-decision problem.
  - The conditions that have to be met for a theory to be subterm-inactive are easy to test, and all but one can be *automatically tested*. This is a significant advantage, compared to the termination proofs of [ARR03, ABRS05] where one has to analyze the inferences that can be carried out starting from the presentation $\mathcal{T}$ and a set of ground unit clauses. Furthermore, the only requirement we impose on the complete simplification ordering $\succ$ assumed by $\mathcal{SP}$ is that $t \succ c$ for every compound term t and constant c.
  - We prove that every subterm-inactive theory is also variable-inactive.
  - We show that several of the theories considered in [ARR03, ABRS05], as well as two extensions of the theory of arrays, are subterm-inactive.

Related work on Recursive Data Structures. Theories of recursive data structures were studied by Oppen in [Opp80], where he described a linear satisfiability procedure for the case where uninterpreted function symbols are excluded. In [ZSM04], Zhang et al. investigated quantifier-elimination problems for an extension of the theory considered by Oppen: their setting includes atoms (constants) and several different constructors. However, their setting also excludes uninterpreted function symbols. They provided a satisfiability procedure for this theory that "guesses" a so-called *type completion*, to determine which constructor was used on each term, or whether the term is an atom, and then calls Oppen's algorithm.

In this paper, we consider the recursive data structures as defined in [Opp80], since our aim was to investigate how to apply the rewrite-based methodology to theories

defined by *infinite sets of axioms*. Similar to any other theory for which the superposition calculus can be used as a satisfiability procedure, all these theories can be combined with the theory of equality with uninterpreted functions. For instance, it can be used to prove the $\mathcal{RDS}_k$-unsatisfiability of a set such as

$$S \;=\; \{\mathsf{cons}(c_1,\ldots,c_k) \simeq c,\ \mathsf{cons}(c_1,\ldots,c_k) \simeq c',\ f(c) \not\simeq f(c')\},$$

where f is an uninterpreted function symbol.

1 Preliminaries

1.1 Terms, literals and clauses

Given a signature Σ, Σ^n denotes the set of functions in Σ with arity n. Thus, Σ^0 denotes the set of constants in Σ. We consider the standard definitions of Σ-terms, Σ-predicates, Σ-literals and Σ-clauses. As usual, clauses are assumed to be variable-disjoint. In the following, $\simeq$ is unordered equality, $\bowtie$ is either $\simeq$ or $\not\simeq$. The letters l, r, u, v and t will denote terms, w, x, y, z variables, and all other lower-case letters will denote constants or function symbols. Given a term t, $\mathrm{top}(t)$ is the symbol appearing as t's top symbol, and $\mathrm{Var}(t)$ denotes the set of variables appearing in t. We will also consider the natural extension of Var to literals and clauses: for example, if C is a clause, then $\mathrm{Var}(C)$ is the set of variables appearing in C. Given the presentation $\mathcal{T}$ of a theory, a function symbol is *interpreted* if it appears in $\mathcal{T}$, and *uninterpreted* otherwise.

Given a presentation $\mathcal{T}$, the *$\mathcal{T}$-satisfiability problem* is the problem of deciding whether a set of ground unit clauses is satisfiable in $\mathcal{T}$. The more general *$\mathcal{T}$-decision problem* is the problem of deciding the satisfiability of any ground formula in $\mathcal{T}$. Without loss of generality, we can assume that the considered ground formulae are conjunctions of clauses.

1.2 Flattening

If a term t is a constant or a variable, then the *depth* of t is $\mathrm{depth}(t) = 0$, and $\mathrm{depth}(f(t_1,\ldots,t_n)) = 1 + \max\{\mathrm{depth}(t_i) \mid i = 1,\ldots,n\}$. The depth of the literal $l \bowtie r$ is $\max(\mathrm{depth}(l), \mathrm{depth}(r))$. A positive literal is *flat* if $\mathrm{depth}(l) + \mathrm{depth}(r) \leq 1$, and a negative literal is *flat* if its depth is 0.

Definition 1.1 A literal is *strictly flat* if its depth is 0. For a clause C, let $\mathrm{Maxd}(C) = \max\{\mathrm{depth}(t) \mid t \text{ is a term appearing in } C\}$. The clause C is *flat*, respectively, *strictly flat*, if all its literals are. A substitution is *strictly flat* if its range only contains variables or constants. $\diamond$

Proposition 1.2 *Let σ be a substitution, t a term and C a clause.*

1. *If σ is the mgu of two terms of depth 1, then σ is strictly flat.*
2. *If σ is strictly flat, then* $\mathrm{depth}(t\sigma) = \mathrm{depth}(t)$ *and* $\mathrm{Maxd}(C\sigma) = \mathrm{Maxd}(C)$.

Superposition	$\dfrac{C \vee l[u'] \simeq r \quad D \vee u \simeq t}{(C \vee D \vee l[t] \simeq r)\sigma}$	$(i), (ii), (iii), (iv)$
Paramodulation	$\dfrac{C \vee l[u'] \not\simeq r \quad D \vee u \simeq t}{(C \vee D \vee l[t] \not\simeq r)\sigma}$	$(i), (ii), (iii), (iv)$
Reflection	$\dfrac{C \vee u' \not\simeq u}{C\sigma}$	(v)
Equational Factoring	$\dfrac{C \vee u \simeq t \vee u' \simeq t'}{(C \vee t \not\simeq t' \vee u \simeq t')\sigma}$	$(i), (vi)$

where the notation $l[u']$ means that u' appears as a subterm in l, σ is the most general unifier (mgu) of u and u', u' is not a variable in *Superposition* and *Paramodulation*, and the following abbreviations hold:

(i) is $u\sigma \not\preceq t\sigma$,

(ii) is $\forall L \in D : (u \simeq t)\sigma \not\preceq L\sigma$,

(iii) is $l[u']\sigma \not\preceq r\sigma$, and

(iv) is $\forall L \in C : (l[u'] \bowtie r)\sigma \not\preceq L\sigma$.

(v) is $\forall L \in C : (u' \simeq u)\sigma \not\prec L\sigma$.

(vi) is $\forall L \in \{u' \simeq t'\} \cup C : (u \simeq t)\sigma \not\prec L\sigma$.

Figure 1: Expansion inference rules of $\mathcal{SP}$: in expansion rules, what is below the inference line is added to the clause set that contains what is above the inference line.

We will make an intensive use of *flattening*. The operation of flattening consists of transforming a finite set of ground clauses S over a signature Σ, into an equisatisfiable finite set of ground clauses S' over a signature Σ', such that:

- Σ' is obtained by adding a finite number of constants to Σ,
- every non-unit clause in S' is strictly flat,
- every unit clause in S' is flat.

This flattening operation is fairly straightforward, and it is more general than the one in [ARR03], where only unit clauses are considered.

Example 1.3 Suppose that $S = \{f(f(a)) \simeq b \vee f(c) \not\simeq d\}$, then by flattening S we obtain the set

$$S' \;=\; \{f(a) \simeq c_1,\ f(c_1) \simeq c_2,\ f(c) \simeq c_3,\ c_2 \simeq b \vee c_3 \not\simeq d\}.$$

1.3 Rewrite-based inference systems

A *simplification ordering* $\succ$ is an ordering that is *stable*, *monotonic* and contains the *subterm ordering*: if $s \succ t$, then $c[s]\sigma \succ c[t]\sigma$ for any context c and substitution σ,

Strict Subsumption	$\dfrac{C \quad D}{C}$	$D \succcurlyeq C$
Simplification	$\dfrac{C[u] \quad l \simeq r}{C[r\sigma] \quad l \simeq r}$	$u = l\sigma,\ l\sigma \succ r\sigma,\ C[u] \succ (l \simeq r)\sigma$
Deletion	$\dfrac{C \vee t \simeq t}{}$	

where $D \succcurlyeq C$ if $D \trianglerighteq C$ and $C \not\trianglerighteq D$; and $D \trianglerighteq C$ if $C\sigma \subseteq D$ (as multisets) for some substitution σ. In practice, theorem provers such as E apply also subsumption of variants: if $D \trianglerighteq C$ and $C \trianglerighteq D$, the oldest clause is retained.

Figure 2: Contraction inference rules of $\mathcal{SP}$: in contraction rules, what is above the double inference line is removed from the clause set and what is below the double inference line is added to the clause set.

and if t is a subterm of s then $s \succ t$. A *complete simplification ordering*, or CSO, is a simplification ordering that is total on ground terms. We write $t \prec s$ if $s \succ t$. More details on orderings can be found, e.g., in [DP01].

In the sequel, except stated otherwise, we will assume that for the considered CSO, if t is a compound term and c a constant, then $t \succ c$. This condition is part of the *$\mathcal{T}$-goodness requirement* for all the theories considered in [ABRS05]. We will refer to this requirement simply as the *goodness requirement*.

The *superposition calculus*, or $\mathcal{SP}$ (see [NR01]), is a *rewrite-based inference system* which is refutationally complete for first-order logic with equality. It consists of *expansion rules* (see Figure 1) and *contraction rules* (see Figure 2), and is based on a CSO on terms which is extended to literals and clauses in the standard way. Given a CSO $\succ$, we write $\mathcal{SP}_\succ$ for $\mathcal{SP}$ equipped with $\succ$. A clause C is *redundant* with respect to $\mathcal{SP}$ in a set of clauses S, if S can be derived from $S \cup \{C\}$ by application of a contraction rule in $\mathcal{SP}$. Since $\mathcal{SP}$ is the only inference system in this article, we write redundant for redundant with respect to $\mathcal{SP}$. An inference is *redundant* in S, if either its conclusion or one of its premises is redundant in S. An *$\mathcal{SP}$-strategy* is given by $\mathcal{SP}$ together with a search plan that controls the application of the inference rules. An *$\mathcal{SP}_\succ$-derivation* is a sequence

$$S_0 \vdash_{\mathcal{SP}_\succ} S_1 \vdash_{\mathcal{SP}_\succ} \ldots S_i \vdash_{\mathcal{SP}_\succ} \ldots,$$

where each S_i is a set of clauses, obtained by applying an expansion or a contraction rule to clauses in S_{i-1}. The *limit* of such a derivation is the set of *persistent clauses*:

$$S_\infty \;=\; \bigcup_{j\geq 0} \bigcap_{i\geq j} S_i.$$

A derivation $S_0 \vdash_{\mathcal{SP}_\succ} \ldots S_n \vdash_{\mathcal{SP}_\succ} \ldots$ is *fair* with respect to $\mathcal{SP}_\succ$ if all expansion inferences in $\mathcal{SP}_\succ$ with premises in S_∞ are redundant in some S_j for $j \geq 0$. A search

plan is *fair* if all the derivations it controls are fair, and an $\mathcal{SP}_\succ$-strategy is fair if its search plan is. A set of clauses S is *saturated* if every clause generated from clauses in S by an $\mathcal{SP}$-inference is redundant.

A clause C is *variable-inactive for* $\succ$ ([ABRS05]) if no maximal literal in C is an equation $t \simeq x$, where $x \notin \mathrm{Var}(t)$. A set of clauses is *variable-inactive for* $\succ$ if all its clauses are variable-inactive for $\succ$. A presentation $\mathcal{T}$ is *variable-inactive for* $\succ$ if the limit S_∞ of any fair $\mathcal{SP}_\succ$-derivation from $S_0 = \mathcal{T} \cup S$ is variable-inactive. When no confusion is possible, we will say that a clause (resp. a set of clauses or a theory presentation) is variable-inactive, without any mention of $\succ$.

Definition 1.4 Let C, C' and D be clauses, and suppose D is generated from C by a unary inference: this inference is *depth-preserving* if $\mathrm{Maxd}(D) \leq \mathrm{Maxd}(C)$. Suppose D is generated from C and C' by a binary inference: this inference is *depth-preserving* if $\mathrm{Maxd}(D) \leq \max\{\mathrm{Maxd}(C), \mathrm{Maxd}(C')\}$. $\diamond$

In general, there is not much we can say about the depth of a clause generated by a unary or binary expansion rule: this depth depends on the mgu used in the inference. When this mgu is strictly flat, we have the following result:

Proposition 1.5 *Let C and C' be two clauses, and suppose a unary (resp. binary) expansion rule can be applied to C (resp. C and C') with a strictly flat mgu. If D is the clause we obtain, then* $\mathrm{Maxd}(D) \leq \mathrm{Maxd}(C)$ *(resp.* $\mathrm{Maxd}(D) \leq \max\{\mathrm{Maxd}(C), \mathrm{Maxd}(C')\}$*)*

Proof. This is an immediate consequence of Proposition 1.2. ∎

2 The theory of recursive data structures

The theory $\mathcal{RDS}_k$ of recursive data structures is based on the following signature:

$$\begin{aligned} \Sigma_{\mathcal{RDS}_k} &= \{\mathsf{cons}\} \cup \Sigma_{sel}, \\ \Sigma_{sel} &= \{\mathsf{sel}_1, \ldots, \mathsf{sel}_k\}, \end{aligned}$$

where cons has arity k, and the sel_i's all have arity 1. The function symbols $\mathsf{sel}_1, \ldots, \mathsf{sel}_k$ stand for the *selectors*, and cons stands for the *constructor*. This theory is axiomatized by the following (infinite) set of axioms, denoted $Ax(\mathcal{RDS}_k)$:

$$\begin{aligned} \mathsf{sel}_i(\mathsf{cons}(x_1, \ldots, x_i, \ldots, x_k)) &\simeq x_i \quad \text{for } i = 1, \ldots, k \\ \mathsf{cons}(\mathsf{sel}_1(x), \ldots, \mathsf{sel}_k(x)) &\simeq x, \\ t[x] &\not\simeq x, \end{aligned}$$

where x and the x_i's are (implicitly) universally quantified variables and $t[x]$ is any compound Σ_{sel}-term where the variable x occurs. The axioms $t[x] \not\simeq x$ are *acyclicity* axioms that prevent the theory from entailing equations such as $\mathsf{sel}_1(\mathsf{sel}_2(\mathsf{sel}_3(x))) \simeq x$.

For the sake of clarity, we also define

$$\begin{aligned} Ac &= \{t[x] \not\simeq x \mid t[x] \text{ is a } \Sigma_{sel}\text{-term}\}, \\ Ac(n) &= \{t[x] \not\simeq x \mid t[x] \text{ is a } \Sigma_{sel}\text{-term and } \mathrm{depth}(t[x]) \leq n\}. \end{aligned}$$

Example 2.1 Consider the case where $k = 2$. If we write $\mathrm{car}(x)$ instead of $\mathsf{sel}_1(x)$ and $\mathrm{cdr}(x)$ instead of $\mathsf{sel}_2(x)$, then our axioms become:

$$\begin{aligned} \mathrm{car}(\mathsf{cons}(x, y)) &\simeq x, \\ \mathrm{cdr}(\mathsf{cons}(x, y)) &\simeq y, \\ \mathsf{cons}(\mathrm{car}(x), \mathrm{cdr}(x)) &\simeq x, \\ t[x] &\not\simeq x, \end{aligned}$$

and for example, we have:

$$\begin{aligned} Ac(2) = \{&\mathrm{car}(\mathrm{car}(x)) \not\simeq x, \quad \mathrm{cdr}(\mathrm{cdr}(x)) \not\simeq x, \\ &\mathrm{car}(\mathrm{cdr}(x)) \not\simeq x, \quad \mathrm{cdr}(\mathrm{car}(x)) \not\simeq x\}. \end{aligned}$$

We consider the problem of checking the $\mathcal{RDS}_k$-satisfiability of a set S of ground (equational) literals built out of the symbols in $\Sigma_{\mathcal{RDS}_k}$ and a set of finitely many constant symbols. This is done by checking the satisfiability of the following set of clauses:

$$Ax(\mathcal{RDS}_k) \quad \cup \quad S.$$

According to the methodology of [ARR03, ABRS05, ABRS06], this problem is solved in three phases:

Flattening: flatten all ground literals in the original problem, thus obtaining an equisatisfiable set of flat literals,

$\mathcal{RDS}_k$-reduction: transform the flattened problem into an equisatisfiable $\mathcal{RDS}_k$-*reduced* problem consisting of a *finite* set of clauses,

Termination: prove that any fair $\mathcal{SP}_\succ$-strategy terminates on the $\mathcal{RDS}_k$-reduced problems.

The flattening step is straightforward, and we now focus on the $\mathcal{RDS}_k$-reduction step.

3 $\mathcal{RDS}_k$-reduction

The aim of a reduction is to transform a formula into another one which is equisatisfiable and easier to work on. Here, given a formula S, we want to transform it into a formula which is equisatisfiable in a theory that does not axiomatize the relationship between the constructor and the selectors. We begin by observing that S can be transformed by suppressing either every occurrence of cons, or every occurrence of the sel_i's.

Example 3.1 Consider the case where $k = 2$, and let

$$S \;=\; \{\mathsf{cons}(c_1, c_2) \simeq c,\ \mathsf{sel}_1(c) \simeq c_1'\}.$$

If we remove the occurrence of cons, S would become

$$S_1 \;=\; \{\mathsf{sel}_1(c) \simeq c_1,\ \mathsf{sel}_2(c) \simeq c_2,\ \mathsf{sel}_1(c) \simeq c_1'\}.$$

If we remove the occurrence of sel_1, S would become

$$S_2 \;=\; \{\mathsf{cons}(c_1, c_2) \simeq c,\ c_1 \simeq c_1'\}.$$

We choose to remove every occurrence of cons because it is easier to work with function symbols of arity 1:

Definition 3.2 A set of ground flat literals is $\mathcal{RDS}_k$*-reduced* if it contains no occurrence of cons. ◇

Given a set S of ground flat literals, the symbol cons may appear only in literals of the form $\mathsf{cons}(c_1, \ldots, c_k) \simeq c$ for constants $c, c_1, \ldots, c_k$. Negative ground flat literals are of the form $c \not\simeq c'$ and therefore do not contain any occurrence of cons. The $\mathcal{RDS}_k$-reduction of S is obtained by replacing every literal $\mathsf{cons}(c_1, \ldots, c_k) \simeq c$ appearing in S by the literals $\mathsf{sel}_1(c) \simeq c_1, \ldots, \mathsf{sel}_k(c) \simeq c_k$. The resulting $\mathcal{RDS}_k$-reduced form S' of S is denoted $\mathit{Red}_{\mathcal{RDS}_k}(S)$, and it is obviously unique.

It is not intuitive in which theory the $\mathcal{RDS}_k$-reduced form of S is equisatisfiable to S, and we need the following definition:

Definition 3.3 Let (**ext**) denote the following "extensionality lemma":

$$\bigwedge_{i=1}^{k} (\mathsf{sel}_k(x) \simeq \mathsf{sel}_k(y)) \quad\Rightarrow\quad x \simeq y. \qquad \diamond$$

Proposition 3.4 *The extensionality lemma is logically entailed by the axiom* $\mathsf{cons}(\mathsf{sel}_1(x), \ldots, \mathsf{sel}_k(x)) \simeq x$.

Proof. We show that the set

$$\{\mathsf{cons}(\mathsf{sel}_1(x), \ldots, \mathsf{sel}_k(x)) \simeq x\} \cup \{\mathsf{sel}_i(a) \simeq \mathsf{sel}_i(b) \mid i = 1, \ldots, k\} \cup \{a \not\simeq b\}$$

is $\mathcal{RDS}_k$-unsatisfiable. The superposition of literal $\mathsf{sel}_1(a) \simeq \mathsf{sel}_1(b)$ into $\mathsf{cons}(\mathsf{sel}_1(x), \ldots, \mathsf{sel}_k(x)) \simeq x$ yields $\mathsf{cons}(\mathsf{sel}_1(b), \mathsf{sel}_2(a), \ldots, \mathsf{sel}_k(a)) \simeq a$. Then, the respective superpositions of $\mathsf{sel}_2(a) \simeq \mathsf{sel}_2(b)$, $\mathsf{sel}_3(a) \simeq \mathsf{sel}_3(b)$, etc, yield $\mathsf{cons}(\mathsf{sel}_1(b), \ldots, \mathsf{sel}_k(b)) \simeq a$. Finally, a superposition of the latter into $\mathsf{cons}(\mathsf{sel}_1(x), \ldots, \mathsf{sel}_k(x)) \simeq x$ produces the literal $a \simeq b$, which contradicts $a \not\simeq b$. ∎

We can then show that $\mathcal{RDS}_k$-reduction reduces satisfiability w.r.t. $Ax(\mathcal{RDS}_k)$ to satisfiability w.r.t. $Ac \cup \{(\mathbf{ext})\}$.

Lemma 3.5 *Let S be a set of ground flat literals, then $Ax(\mathcal{RDS}_k) \cup S$ is satisfiable if and only if $Ac \cup \{(\mathbf{ext})\} \cup Red_{\mathcal{RDS}_k}(S)$ is.*

PROOF. ($\Rightarrow$) For $i = 1, \ldots, k$, literal $\mathsf{sel}_i(c) \simeq c_i$ is a logical consequence of $Ax(\mathcal{RDS}_k)$ and $\mathsf{cons}(c_1, \ldots, c_k) \simeq c$. Indeed, it can be generated by a superposition of the latter into the axiom $\mathsf{sel}_i(\mathsf{cons}(x_1, \ldots, x_i, \ldots, x_k)) \simeq x_i$. So we have that $Ax(\mathcal{RDS}_k) \cup S \models Ac \cup Red_{\mathcal{RDS}_k}(S)$. By Proposition 3.4, it is also the case that $Ax(\mathcal{RDS}_k) \cup S \models \{(\mathbf{ext})\}$, hence the result.
($\Leftarrow$) Let $M = (D, I)$ be a model for $Ac \cup \{(\mathbf{ext})\} \cup Red_{\mathcal{RDS}_k}(S)$. We will build a model $M' = (D', I')$ for $Ax(\mathcal{RDS}_k) \cup S$ starting from M. In particular, I' must interpret the function symbol cons in such a way that any sequence $d_1, \ldots, d_k$ of elements of D' has an image by $\mathsf{cons}^{I'}$. We inductively build a model $M' = (D', I')$ for $Ax(\mathcal{RDS}_k) \cup S$ as follows: first, I' and I both interpret the constants appearing in S the same way; second, for every $d \in D$, we let $\mathsf{sel}_i^{I'}(d) = \mathsf{sel}_i^I(d)$ for all $i = 1, \ldots, k$.

Let $D_0 = D$, and consider the k-fold Cartesian product $D_0^k = D_0 \times \ldots \times D_0$. We start by separating the elements in D_0^k that can be represented as a tuple $\langle \mathsf{sel}_1^{I'}(d), \ldots, \mathsf{sel}_k^{I'}(d) \rangle$ with $d \in D_0$, from those that cannot. Formally, we define the following partition of D_0^k:

$$\begin{aligned} E_0 &= \{\langle \mathsf{sel}_1^{I'}(d), \ldots, \mathsf{sel}_k^{I'}(d) \rangle \mid d \in D_0\}, \\ F_0 &= D_0^k \setminus E_0. \end{aligned}$$

Note that by construction, for every $\langle d_1, \ldots, d_k \rangle \in E_0$, there exists a $d \in D_0$ such that $\mathsf{sel}_i^{I'}(d) = d_i$ for all $i = 1, \ldots, k$. Furthermore, since M satisfies axiom $(\mathbf{ext})$, d is unique. Hence, we can safely define $\mathsf{cons}^{I'}(d_1, \ldots, d_k) = d$. Therefore, for every tuple $\langle d_1, \ldots, d_k \rangle$ in E_0, if $d = \mathsf{cons}^{I'}(d_1, \ldots, d_k)$, then we have

$$\mathsf{sel}_i^{I'}(d) = d_i, \text{ and } \mathsf{cons}^{I'}(\mathsf{sel}_1^{I'}(d), \ldots, \mathsf{sel}_k^{I'}(d)) = d.$$

We now extend the function $\mathsf{cons}^{I'}$ to the elements in F_0. We let D_0' be a set disjoint from $F_0 \cup D_0$, such that there exists a bijection η_0 from F_0 to D_0'. Intuitively, D_0' will provide the images $\mathsf{cons}^{I'}(d_1, \ldots, d_k)$ of all the tuples $\langle d_1, \ldots, d_k \rangle$ in F_0, and each tuple is associated to its image by η_0. Formally, for every element $t = \langle d_1, \ldots, d_k \rangle$ in F_0, we define $\mathsf{sel}_i^{I'}(\eta_0(t)) = d_i$ for $i = 1, \ldots, k$, and $\mathsf{cons}^I(d_1, \ldots, d_k) = \eta_0(t)$. Let $D_1 = D_0 \uplus D_0'$: obviously $D_0 \subseteq D_1$, and for every $\langle d_1, \ldots, d_k \rangle \in D_0^k$, the element $d = \mathsf{cons}^{I'}(d_1, \ldots, d_k)$ is well-defined and verifies

$$\forall i = 1, \ldots, k, \ \mathsf{sel}_i^{I'}(d) = d_i, \text{ and } \mathsf{cons}^{I'}(\mathsf{sel}_1^{I'}(d), \ldots, \mathsf{sel}_k^{I'}(d)) = d.$$

At this point, since I and I' interpret the constant symbols from S and the selector functions on D_0 the same way, it is clear that I' satisfies $Ac \cup S$, as well as the other axioms of $Ax(\mathcal{RDS}_k)$ on D_0. However, I' may still not be an interpretation, since the function $\mathsf{cons}^{I'}$ is not defined on the Cartesian product D_1^k. This is why we perform the following induction step.

Suppose that for $p \geq 1$, we have constructed a set D_p such that $D_{p-1} \subseteq D_p$, on which we have defined the $\mathsf{sel}_i^{I'}$'s and $\mathsf{cons}^{I'}$ in such a way that for every $\langle d_1, \ldots, d_k \rangle \in$

D_{p-1}^k, there exists a $d \in D_p$ such that for all $i = 1, \ldots, k$, $\mathsf{sel}_i^{I'}(d) = d_i$, and $\mathsf{cons}^{I'}(\mathsf{sel}_1^{I'}(d), \ldots, \mathsf{sel}_k^{I'}(d)) = d$. Then as previously, we define the sets E_p and F_p by:

$$\begin{array}{rcl} E_p & = & \{\langle \mathsf{sel}_1^{I'}(d), \ldots, \mathsf{sel}_k^{I'}(d)\rangle \mid d \in D_p\}, \\ F_p & = & D_p^k \setminus E_p. \end{array}$$

Let D'_p be a set disjoint from $F_p \cup D_p$, such that there exists a bijection η_p from F_p to D'_p. For every element $t = \langle d_1, \ldots, d_k\rangle$ in F_p, we define $\mathsf{sel}_i^{I'}(\eta_p(t)) = d_i$ for $i = 1, \ldots, k$, and $\mathsf{cons}^{I'}(d_1, \ldots, d_k) = \eta_p(t)$. Finally, we let $D_{p+1} = D_p \uplus D'_p$, and it is clear that D_{p+1} satisfies the required property.

Let $D' = \bigcup_{i\geq 0} D_i$, then I' is an interpretation on D'. By construction, for every $\langle d_1, \ldots, d_k\rangle \in D'^k$, the image $\mathsf{cons}^{I'}(d_1, \ldots, d_k)$ is well-defined. Also by construction, we have that $\mathsf{sel}_i^{I'}(\mathsf{cons}^{I'}(d_1, \ldots, d_k)) = d_i$ and for every $d \in D'$, $\mathsf{cons}^{I'}(\mathsf{sel}_1^{I'}(d), \ldots, \mathsf{sel}_k^{I'}(d)) = d$. Thus, M' is a model for $Ax(\mathcal{RDS}_k)$. Furthermore, since I and I' both interpret constants the same way and f^I and $f^{I'}$ are identical on D for every $f \in \Sigma_{sel}$, M' is also a model for S. ∎

Example 3.6 Consider the case where $k = 1$, and let $S = \{\mathsf{cons}(c') \simeq c\}$. The $\mathcal{RDS}_1$-reduced form of S is therefore $S' = \{\mathsf{sel}_1(c) \simeq c'\}$. We consider the model $M = (\mathbb{N}, I)$ of $Ac \cup \{(\mathbf{ext})\} \cup S'$, where I interprets c as 0, c' as 1, and sel_1 as the successor function on natural numbers. Then we have

$$D_0 = \mathbb{N},\ E_0 = \mathbb{N} \setminus \{0\},\ \text{and } F_0 = \{0\},$$

and for every $d \in E_0$, $\mathsf{cons}^{I'}(d)$ is the d' such that $\mathsf{sel}_1^{I'}(d') = d$, hence $\mathsf{cons}^{I'}(d)$ is the predecessor of d.

We now select a set D'_0 disjoint from $F_0 \cup D_0$ such that there exists a bijection from F_0 to D'_0. We can for example choose $D'_0 = \{-1\}$, then define $\mathsf{sel}_1^{I'}(-1) = 0$, $\mathsf{cons}^{I'}(0) = -1$, and let $D_1 = \mathbb{N} \cup \{-1\}$. Then $F_1 = \{-1\}$ and we can choose $D'_1 = \{-2\}$, etc. At the end, we obtain $M' = (D', I')$, where $D' = \mathbb{Z}$, I' interprets sel_1 as the standard successor function on integers, and cons as the standard predecessor function on integers. It is clear that M' is a model of $Ax(\mathcal{RDS}_k) \cup S$.

It is also possible to define a notion of $\mathcal{RDS}_k$-reduction where every occurrence of the sel_i's is removed. However, no additional property is gained by using this other alternative, and the corresponding reduction is less intuitive.

4 From Ac to $Ac(n)$

The set Ac being infinite, $\mathcal{SP}$ cannot be used as a satisfiability procedure on any set of the form $Ac \cup \{(\mathbf{ext})\} \cup S$, where S is an $\mathcal{RDS}_k$-reduced set of literals. Thus, the next move is to bound the number of axioms in Ac needed to solve the satisfiability problem, and try to consider an $Ac(n)$ instead of Ac. It is clear that for any n and any set S, a model of $Ac \cup \{(\mathbf{ext})\} \cup S$ is also a model of $Ac(n) \cup \{(\mathbf{ext})\} \cup S$, the difficulty is

therefore to determine an n for which a model of $Ac \cup \{(\mathbf{ext})\} \cup S$ is guaranteed to exist, provided $Ac(n) \cup \{(\mathbf{ext})\} \cup S$ is satisfiable. The following example provides the intuition that this bound depends on the number of selectors in S.

Example 4.1 Let $S = \{\mathsf{sel}_1(c_1) \simeq c_2,\ \mathsf{sel}_2(c_2) \simeq c_3,\ \mathsf{sel}_3(c_3) \simeq c_4,\ c_1 \simeq c_4\}$. Then:

$$\begin{array}{llll} S \cup \{(\mathbf{ext})\} \cup Ac(1) & \text{and} & S \cup \{(\mathbf{ext})\} \cup Ac(2) & \text{are satisfiable,} \\ S \cup \{(\mathbf{ext})\} \cup Ac(3) & \text{and} & S \cup \{(\mathbf{ext})\} \cup Ac & \text{are unsatisfiable.} \end{array}$$

The following lemma allows us to prove that having n occurrences of selectors implies that it is indeed sufficient to consider $Ac(n)$ instead of Ac. We start by introducing the notion of an *M-path*.

Definition 4.2 Let $M = (D, I)$ be a model for an $\mathcal{RDS}_k$-reduced set of literals S. For every $m \geq 2$, a tuple $p = \langle d_1, f_1, d_2, f_2, \ldots, d_m, f_m \rangle$ is called an *M-path* if for $i = 1, \ldots, m$,

1. $f_i \in \Sigma_{sel}$,
2. for all $j \in \{i+1, \ldots, m\}$, $d_j \neq d_i$,
3. if $i \leq m-1$, then $d_{i+1} = f_i^I(d_i)$.

The *length* of p is m, and we say that p is *cyclic* if $f_m^I(d_m) = d_1$. $\diamond$

Intuitively, there is an M-path of length m from d to d' if and only if we have $f_m^I(f_{m-1}^I(\ldots(f_1^I(d))\ldots)) = d'$. Thus, if $d = d'$, then I violates one of the axioms $t[x] \not\simeq x$, where t is of depth m.

Example 4.3 Consider the case where $k = 2$, and let $S = \{\mathsf{sel}_1(c) \simeq c_1, \mathsf{sel}_2(c) \simeq c_2\}$. Let $D = \{1, 2, 3\}$, and define:

$$\begin{array}{lllllllll} I(c) & = & 1, & I(c_1) & = & 2, & I(c_2) & = & 3, \\ \mathsf{sel}_1^I(1) & = & 2, & \mathsf{sel}_1^I(2) & = & 3, & \mathsf{sel}_1^I(3) & = & 1, \\ \mathsf{sel}_2^I(1) & = & 3, & \mathsf{sel}_2^I(2) & = & 3, & \mathsf{sel}_2^I(3) & = & 2. \end{array}$$

Then $M = (D, I)$ is a model for S, $\langle 1, \mathsf{sel}_1, 2, \mathsf{sel}_1 \rangle$ is an M-path of length 2, and $\langle 1, \mathsf{sel}_1, 2, \mathsf{sel}_2, 3, \mathsf{sel}_1 \rangle$ is a cyclic M-path of length 3.

We have the following obvious property:

Proposition 4.4 *Let M be a model for a set of literals and $l \in \mathbb{N}$, then $M \models Ac(l)$ if and only if the length of every cyclic M-path is strictly greater than l.*

Definition 4.5 Given a set S of $\mathcal{RDS}_k$-reduced literals and $M = (D, I)$ a model for S, we say that an element $d \in D$ is *selector-free in S* if and only if for no $f(c) \simeq c' \in S$ (where $f \in \Sigma_{sel}$), is c interpreted as d. $\diamond$

Example 4.6 In Example 4.3, element 1 is not selector-free in S, since $\mathsf{sel}_1(c) \simeq c_1 \in S$ and $I(c) = 1$. However, elements 2 and 3 both are.

Intuitively, an element is selector-free if its images by the sel_i^I's are not constrained to be the images of constants in S. This will allow us in Lemma 4.8 to define an interpretation I' that does not interpret the sel_i's as I does, but still satisfies S.

Proposition 4.7 *Let $M = (D, I)$ be a model for a set S containing l occurrences of selectors, and let p be an M-path of length at least $l + 1$. Then at least one of the elements appearing in p is selector-free in S.*

Proof. Let $m = l + k$, where $k \geq 1$, $p = \langle d_1, f_1, \ldots, d_m, f_m \rangle$, and suppose that no element appearing in p is selector-free in S. Then by definition, for every $j = 1, \ldots, m$, there must be a literal $f_j(c_j) \simeq c'_j$ appearing in S, such that $I(c_j) = d_j$. By hypothesis, since p is an M-path, the d_j's are all distinct, so there must be at least m distinct literals in S. This is impossible, since S contains $l < m$ such literals. ∎

We now state the lemma relating the number of selectors in S and the sets $Ac(n)$ that can safely replace Ac.

Lemma 4.8 *Let S be an $\mathcal{RDS}_k$-reduced set of ground flat literals and let l be the number of occurrences of selectors in S. For $n \geq l$, suppose that $Ac(n) \cup \{(\mathbf{ext})\} \cup S$ is satisfiable. Then $Ac(n+1) \cup \{(\mathbf{ext})\} \cup S$ is also satisfiable.*

Proof. Let $M = (D, I)$ be a model of $Ac(n) \cup \{(\mathbf{ext})\} \cup S$. We are going to build a model M' of $\{(\mathbf{ext})\} \cup S$, starting from M, such that there are no cyclic M'-paths of length smaller or equal to $n + 1$. Thus, M' will also be a model of $Ac(n + 1)$.

Let $P = \{p \mid p \text{ is a cyclic } M\text{-path of length } n + 1\}$. If P is empty, then there are no cyclic M-paths of length $n+1$, so that $M \models Ac(n+1) \cup \{(\mathbf{ext})\} \cup S$ by Proposition 4.4. Otherwise let $p \in P$, since there are l occurrences of selectors in S, by Proposition 4.7 we must have $p = \langle \ldots, d, f, \ldots \rangle$, where d is selector-free in S, and $f \in \Sigma_{sel}$. Consider a set $E_p = \{e_j \mid j \geq 0\}$ disjoint from D, and let I_p be the interpretation on $D \cup E_p$ which is identical to I, except that $f^{I_p}(d) = e_0$, and for $j \geq 0$ and $i = 1, \ldots, k$, $\mathsf{sel}_i^{I_p}(e_i) = e_{i+1}$ (see Figure 3). By repeating this transformation on every M-path in P, we obtain a new model $M' = (D', J)$.

We now show that $M' \models Ac(n + 1) \cup \{(\mathbf{ext})\} \cup S$.

$M' \models Ac(n + 1)$**:** By construction, there is no cyclic M'-path of length smaller or equal to $n + 1$, hence $M' \models Ac(n + 1)$ by Proposition 4.4.

$M' \models \{(\mathbf{ext})\}$**:** Consider two elements d, d' such that $f^J(d) = f^J(d')$ for all $f \in \Sigma_{sel}$. Note that if $f^J(d) \in D$, then $f^J(d) = f^I(d)$ by construction of J. Hence, if for all $f \in \Sigma_{sel}$, $f^J(d) \in D$, then since $M \models \{(\mathbf{ext})\}$, we have $d = d'$. Otherwise, there exists a selector f such that $e = f^J(d) \notin D$, and by construction, d is the unique element that is mapped to e by f^J, so that it must be $d = d'$.

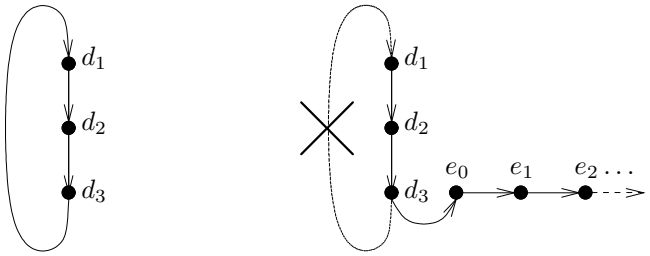

Figure 3: Breaking cyclic M-paths

$M' \models S$**:** Suppose that $f(c) \simeq c' \in S$, since both I and J interpret constants the same way, we have $I(c) = J(c)$, $I(c') = J(c')$, and $f^I(I(c)) = I(c')$. Since $I(c)$ is not selector-free in S, necessarily $f^I(I(c)) = f^J(I(c))$. We deduce that $f^J(J(c)) = f^J(I(c)) = f^I(I(c)) = I(c') = J(c')$, and that $M' \models S$. ■

Corollary 4.9 *Let S be an $\mathcal{RDS}_k$-reduced set of ground flat literals and let n be the number of occurrences of selectors in S. Then, $Ac \cup \{(\mathbf{ext})\} \cup S$ is satisfiable if and only if $Ac(n) \cup \{(\mathbf{ext})\} \cup S$ is.*

Proof. The "only if" part is trivial, and a simple induction using Lemma 4.8 shows that for every $k \geq 0$, if $Ac(n) \cup \{(\mathbf{ext})\} \cup S$ is satisfiable, then so is $Ac(n+k) \cup \{(\mathbf{ext})\} \cup S$.■

5 $\mathcal{SP}_\succ$ as a satisfiability procedure for $\mathcal{RDS}_k$

We now show that only a finite number of clauses are generated by the superposition calculus on any set $Ac(n) \cup \{(\mathbf{ext})\} \cup S$, where S is $\mathcal{RDS}_k$-reduced. This will be the case provided we use a good CSO.

Lemma 5.1 *Let $S_0 = Ac(n) \cup \{(\mathbf{ext})\} \cup S$, where S is a finite $\mathcal{RDS}_k$-reduced set of ground flat literals. Consider the limit S_∞ of the derivation $S_0 \vdash_{\mathcal{SP}_\succ} S_1 \vdash_{\mathcal{SP}_\succ} \ldots$ generated by a fair $\mathcal{RDS}_k$-good $\mathcal{SP}_\succ$-strategy; every clause in S_∞ belongs to one of the categories enumerated below:*

i) the empty clause;

ii) the clauses in $Ac(n) \cup \{(\mathbf{ext})\}$, i.e.

a) $t[x] \not\simeq x$, where t is a Σ_{sel}-term of depth at most n,

b) $x \simeq y \ \vee \left(\bigvee_{i=1}^{k} (\mathsf{sel}_i(x) \not\simeq \mathsf{sel}_i(y)) \right)$;

iii) ground clauses of the form

a) $c \simeq c' \vee (\bigvee_{j=1}^{m} d_j \not\simeq d'_j)$ *where* $m \geq 0$,

b) $f(c) \simeq c' \vee (\bigvee_{j=1}^{m} d_j \not\simeq d'_j)$ *where* $m \geq 0$,

c) $t[c] \not\simeq c' \vee (\bigvee_{j=1}^{m} d_j \not\simeq d'_j)$, *where* t *is a compound* Σ_{sel}*-term of depth at most* $n-1$ *and* $m \geq 0$,

d) $\bigvee_{j=1}^{m} d_j \not\simeq d'_j$, *where* $m \geq 1$;

iv) clauses of the form

$$c \simeq x \vee \left(\bigvee_{p=1}^{j} \mathsf{sel}_{i_p}(c) \not\simeq \mathsf{sel}_{i_p}(x)\right) \vee \left(\bigvee_{p=j+1}^{k} c_{i_p} \not\simeq \mathsf{sel}_{i_p}(x)\right) \vee \left(\bigvee_{j=1}^{m} d_j \not\simeq d'_j\right)$$

where $i_1, \ldots, i_k$ *is a permutation of* $1, \ldots, k$, $0 \leq j \leq k$ *and* $m \geq 0$;

v) clauses of the form

$$c \simeq c' \vee \left(\bigvee_{p=1}^{j_1} (\mathsf{sel}_{i_p}(c) \not\simeq \mathsf{sel}_{i_p}(c'))\right) \vee \left(\bigvee_{p=j_1+1}^{j_2} (\mathsf{sel}_{i_p}(c) \not\simeq c'_{i_p})\right) \vee \left(\bigvee_{p=j_2+1}^{j_3} (c_{i_p} \not\simeq \mathsf{sel}_{i_p}(c'))\right) \vee \left(\bigvee_{p=j_3+1}^{k} (c_{i_p} \not\simeq c'_{i_p})\right) \vee \left(\bigvee_{j=1}^{m} d_j \not\simeq d'_j\right)$$

where $i_1, \ldots, i_k$ *is a permutation of* $1, \ldots, k$, $0 \leq j_1 \leq j_2 \leq j_3 \leq k$, $j_3 > 0$ *and* $m \geq 0$.

PROOF. We prove the result by induction on the length l of the derivations. For $l = 0$, the result is trivial: the clauses in S_0 are in *(ii)* or *(iii)* with $m = 0$. Now assume the result is true for $l-1$, where $l \geq 1$, and that a new inference step is carried out. The result is obvious if the inference performed is a subsumption or a deletion, now suppose that the inference is a reflection. This reflection inference can occur on a clause in *(ii.b)*, in which case a clause containing $x \simeq x$ is generated, hence deleted, or in category *(iii)*, in which case the clause generated belongs to the same category or is the empty clause. We now suppose that the inference is either a simplification, a superposition or a paramodulation. For the sake of conciseness, we write "paramodulation" in all cases.

Paramodulation from *(ii)*, *(iv)* or *(v)*: none applies. For clauses in category *(iv)*, this is due to the fact that the considered clause contains either a literal $\mathsf{sel}_{i_p}(c) \not\simeq \mathsf{sel}_{i_p}(x)$ or $c_{i_p} \not\simeq \mathsf{sel}_{i_p}(x)$, both of which are greater than $c \simeq x$. For clauses in category *(v)*, this is due to the fact that for the considered clause we have $j_3 > 0$, hence it necessarily contains a function symbol $f \in \Sigma_{sel}$, and the literal $c \simeq c'$ cannot be maximal.

Paramodulation from *(iii)*: consider a clause in *(iii.a)*,

- a paramodulation into a clause in *(iii)* yields a clause in *(iii)*,
- a paramodulation into a clause in *(iv)* yields a clause in *(iv)*,
- a paramodulation into a clause in *(v)* yields a clause in *(v)*.

Now consider a clause in *(iii.b)*,

- a paramodulation into a clause in *(ii.a)* yields a clause in *(iii.c)*,
- a paramodulation into a clause in *(ii.b)* yields a clause in *(iv)*,
- a paramodulation into a clause in *(iii.b)* yields a clause in *(iii.a)*,
- a paramodulation into a clause in *(iii.c)* yields a clause in *(iii.c)* or *(iii.d)*,
- a paramodulation into a clause in *(iv)* yields a clause in *(iv)* or in *(v)*,
- a paramodulation into a clause in *(v)* yields a clause in *(v)* or in *(iii.a)*. ∎

We give an example of such a derivation:

Example 5.2 Consider the case where $k = 3$, and suppose we want to test the unsatisfiability of the following set:

$$\begin{array}{rlcccccccccc} S = & \{ & \mathsf{sel}_1(c) & \simeq & d_1, & \mathsf{sel}_2(c') & \simeq & d'_2, & \mathsf{sel}_2(c) & \simeq & d_2, \\ & & \mathsf{sel}_1(c') & \simeq & d'_1, & \mathsf{sel}_3(c) & \simeq & d_3, & \mathsf{sel}_3(c') & \simeq & d'_3, \\ & & d_1 & \simeq & d'_1, & d_2 & \simeq & d'_2, & d_3 & \simeq & d'_3, \\ & & & & & c & \not\simeq & c' & & & & \}. \end{array}$$

- A superposition of $\mathsf{sel}_1(c) \simeq d_1$ into $\{(\mathbf{ext})\}$ yields a clause in *(iv)* (with $m = 0$):

$$c \simeq x \vee \Big(\underline{\mathsf{sel}_2(c) \not\simeq \mathsf{sel}_2(x)} \vee \mathsf{sel}_3(c) \not\simeq \mathsf{sel}_3(x)\Big) \vee \Big(d_1 \not\simeq \mathsf{sel}_1(x)\Big);$$

- A superposition of $\mathsf{sel}_2(c') \simeq d'_2$ into the underlined literal of this clause yields a clause in *(v)*:

$$c \simeq c' \vee \Big(\mathsf{sel}_3(c) \not\simeq \mathsf{sel}_3(c')\Big) \vee \Big(\underline{\mathsf{sel}_2(c) \not\simeq d'_2}\Big) \vee \Big(d_1 \not\simeq \mathsf{sel}_1(c')\Big);$$

- A simplification of this clause by $\mathsf{sel}_2(c) \simeq d_2$ yields a clause in *(v)*:

$$c \simeq c' \vee \Big(\mathsf{sel}_3(c) \not\simeq \mathsf{sel}_3(c')\Big) \vee \Big(d_1 \not\simeq \mathsf{sel}_1(c')\Big) \vee \Big(d_2 \not\simeq d'_2\Big),$$

- Further simplifications by $\mathsf{sel}_1(c') \simeq d'_1$, $\mathsf{sel}_3(c) \simeq d_3$ and $\mathsf{sel}_3(c') \simeq d'_3$ yield the clause

$$c \simeq c' \vee \Big(\bigvee_{i=1}^{3} d_i \not\simeq d'_i\Big).$$

- The simplifications by $d_i \simeq d'_i$ for $i = 1, \ldots, 3$ yield the clause $c \simeq c'$, which together with $c \not\simeq c'$ produces the empty clause.

Since the signature is finite, there are finitely many clauses such as those enumerated in Lemma 5.1. We therefore deduce:

Corollary 5.3 *Any fair $\mathcal{RDS}_k$-good $\mathcal{SP}_\succ$-strategy terminates when applied to $Ac(n) \cup \{(\mathbf{ext})\} \cup S$, where S is a finite $\mathcal{RDS}_k$-reduced set of ground flat literals.*

We can also evaluate the complexity of this procedure by determining the number of clauses in each of the categories defined in Lemma 5.1.

Theorem 5.4 *Any fair $\mathcal{RDS}_k$-good $\mathcal{SP}_\succ$-strategy is an exponential satisfiability procedure for $\mathcal{RDS}_k$.*

PROOF. Let n be the number of literals in S, both the number of constants and the number of selectors appearing in S are therefore in $O(n)$. We examine the cardinalities of each of the categories defined in Lemma 5.1.

- Category *(ii)* contains $O(n)$ clauses if $k = 1$ and $O(k^n)$ clauses if $k \geq 2$.
- Clauses in categories *(iii)*, *(iv)* or *(v)* can contain any literal of the form $d \not\simeq d'$ where d and d' are constants, thus, these categories all contain $O(2^{n^2})$ clauses.

Hence, the total number of clauses generated is bound by a constant which is $O(2^{n^2})$, and since each inference step is polynomial, the overall procedure is in $O(2^{n^2})$. ∎

Although this complexity bound is exponential, it measures the size of the saturated set. Since a theorem prover seeks to generate a proof, as opposed to a saturated set, the relevance of this result with respect to predicting the performance of a theorem prover can therefore be quite limited.

One could actually have expected this procedure to be exponential for $k \geq 2$, since in that case $Ac(n)$ contains an exponential number of axioms. However the procedure is also exponential when $k = 1$, and a more careful analysis shows that this complexity is a consequence of the presence of (**ext**). In fact, it was shown in [ABRS06] that a fair $\mathcal{SP}_\succ$-strategy is a polynomial satisfiability procedure for the theory presented by the set of acyclicity axioms Ac when $k = 1$.

We finally address combination by proving that $\mathcal{RDS}_k$ is variable-inactive for $\mathcal{SP}_\succ$.

Theorem 5.5 *Let $S_0 = Ac(n) \cup S \cup \{(\mathbf{ext})\}$, where S is an $\mathcal{RDS}_k$-reduced set of ground flat literals, and n is the number of occurrences of selectors in S. Then S_∞ is variable-inactive.*

PROOF. The clauses in S_∞ belong to one of the classes enumerated in Lemma 5.1. Thus, the only clauses of S_∞ that may contain a literal $t \simeq x$ where $x \notin \mathrm{Var}(t)$ are in class *(iv)*. Since $\succ$ is a CSO, the literals $t \simeq x$ cannot be maximal in those clauses. ∎

This shows that the rewrite-based approach to satisfiability procedures can be applied to the combination of $\mathcal{RDS}_k$ with any number of the theories considered in [ARR03, ABRS05], including those of arrays and records with or without extensionality.

6 Subterm-inactivity

The proofs that the superposition calculus terminates on satisfiability problems for different theories are based on an enumeration of the kinds of clauses that can be generated by the inferences (see [ARR03, ABRS05, BE06]). However, the number of clauses in S_∞ can be exponentially large (it can contain for example up to $O(2^{n^2})$ clauses in the theory of arrays, see [ARR03] for details), and in general, such proofs consist of showing that all generated clauses belong to one of several categories: if each of these categories contains a finite number of clauses, so will S_∞. These proofs can be quite long, and at each new inference, a new category to deal with may arise. In this section, we introduce a set of conditions guaranteeing that a fair strategy based on $\mathcal{SP}_\succ$ is a decision procedure for the considered theory. These conditions are easy to verify and more importantly, almost all can be verified automatically.

Definition 6.1 Given a signature Σ, a selection function is a function from Σ to $\mathbb{N}$ such that for all $f \in \Sigma$, $\Gamma(f) \in \{1, \ldots, \text{arity(f)}\}$. Ω_Σ denotes the set of selection functions for Σ. $\diamond$

A selection function selects an argument in a term. For example, for a function symbol f and selection function Γ, if $\Gamma(f) = i$, then Γ selects the subterm t_i from the term $f(t_1, \ldots, t_n)$. The name "selection function" is also used for functions that select a literal in a clause: the two definitions are compatible, since such functions can be seen as selecting an argument of a disjunction operator.

We define the notion of symbol-freeness, which prevents some function or constant symbols from appearing in a clause.

Definition 6.2 (Symbol-freeness) Given a term t, $\Phi(t)$ denotes the set of function and constant symbols appearing in t. We also define $\Phi(l \bowtie r) = \Phi(l) \cup \Phi(r)$, and $\Phi(C) = \bigcup_{L \in C} \Phi(L)$.

Given a set of functions Σ', a term t is *Σ'-symbol-free* if $\Phi(t) \cap \Sigma'$ consists only of constants, and t is *strictly Σ'-symbol-free* if this intersection is empty. A literal (resp. clause) is *Σ'-symbol-free* if every term appearing in it is.

A clause is *subsymbol-free from* Σ' if every literal in C that is not strictly flat is strictly Σ'-symbol-free. $\diamond$

Informally, we will consider $\mathcal{T}$-decision problems whose clauses can be divided into three disjoint sets:

- a set T_2 of non-ground clauses representing the way two interpreted functions may interact in $\mathcal{T}$,
- a set T_1 of non-ground clauses representing properties that can be deduced by considering one interpreted function,
- a set T_g of ground clauses.

This pattern applies to $\mathcal{T}$-decision problems in several theories of interest.

Example 6.3 The theory of arrays $\mathcal{A}$ is based on the signature $\Sigma_{\mathcal{A}} = \{\text{select}, \text{store}\}$, where select has arity 2 and store has arity 3, and it is axiomatized as follows:

$$\forall x, z, v.\ \text{select}(\text{store}(x, z, v), z) \simeq v, \tag{1}$$

$$\forall x, z, w, v.\ (z \simeq w \vee \text{select}(\text{store}(x, z, v), w) \simeq \text{select}(x, w)). \tag{2}$$

The theory of arrays with extensionality $\mathcal{A}^{\text{e}}$ is defined by axioms (1) and (2), along with the following extensionality axiom:

$$\forall x, y.\ (\forall z.\ \text{select}(x, z) \simeq \text{select}(y, z) \supset x \simeq y). \tag{3}$$

A rewrite-based $\mathcal{T}$-decision procedure for the theory $\mathcal{A}^{\text{e}}$ takes as input a set T_{g} of ground clauses, together with $\{(1), (2), (3)\}$. This set can itself be decomposed into two disjoint subsets: $T_2 = \{(1), (2)\}$ which describes the way select and store interact, and $T_1 = \{(3)\}$ which describes the equality property that can be deduced from the select function.

Of course, these sets can interact with each other, and it is necessary to control these interactions as much as possible in order to guarantee termination. Before giving any formal definition, we informally enumerate the requirements that should be satisfied by these sets and which conditions are imposed to satisfy them.

General properties

1. Each clause in T_2 expresses a single property verified when combining at most two interpreted function symbols, and each clause in T_1 expresses a property that can be deduced by considering a single interpreted function symbol (**closure**),
2. Any $\mathcal{SP}_{\succ}$-inference generating a persistent clause is depth-preserving (**flatness**).

Binary inferences

1. There is no binary $\mathcal{SP}_{\succ}$-inference between a clause in T_2 and T_1 (**interaction-freeness**),
2. A binary $\mathcal{SP}_{\succ}$-inference between a clause in $T_1 \cup T_2$ and a clause in T_{g} generates a clause in T_1 or in T_{g} (**closure + negative disconnection**),
3. A binary $\mathcal{SP}_{\succ}$-inference between two clauses in T_2 generates a clause which is deleted eventually (**saturation**),
4. A binary $\mathcal{SP}_{\succ}$-inference between two clauses in T_1 generates a clause in T_1 or in T_{g} (**closure**).

Unary inferences

1. A unary inference within T_2 generates a clause that is deleted eventually (**saturation**);

2. A unary inference within T_1 generates a clause that is in T_1, in T_g, or is deleted eventually (**variable-inactivity preservation**).

In the following subsections we define formally these notions.

6.1 Restrictions on T_2

The conditions we impose on T_2 are termed collectively *saturation closure*. Informally, these conditions ensure that T_2 is saturated, and that every clause generated by a binary inference involving a clause in T_2 is in T_1 or in T_g.

Definition 6.4 (Ordered flatness) A clause C is *ordered flat* if it only contains strictly flat literals except for one, say $l \bowtie r$. Furthermore, it must be $r \prec l$, and r must contain only function symbols appearing in l. ◇

Example 6.5 Consider the following clauses:

$$\begin{aligned} C &= f(a) \simeq b \vee c \not\simeq d, \\ C' &= f(g(a)) \simeq g(a) \vee c \not\simeq d. \end{aligned}$$

These two clauses are ordered flat.

Definition 6.6 (Internal closure) Let S be a set of clauses and Γ be a selection function. S is Γ-*internally closed* if for every clause $C \in S$ and every non-strictly flat literal $L = l \bowtie r$ in C:

- If L is negative, then:

  icn.1: for every subterm u of l of depth 1, $\mathrm{Var}(C) \subseteq \mathrm{Var}(u)$,

  icn.2: every positive literal in a clause of S is $\Phi(L)$-symbol-free.

- If L is positive, then we must have $r \prec l$ and:

  icp.1: $\mathrm{depth}(l) = 2$, and l contains a unique subterm u of depth 1,

  icp.2: $\mathrm{Var}(C) = \mathrm{Var}(l)$,

  icp.3: if $\mathrm{top}(r) \neq \mathrm{top}(l)$, then $\mathrm{depth}(r) = 0$ and $\mathrm{Var}(C) \subseteq \mathrm{Var}(u)$,

  icp.4: if $\mathrm{top}(r) = \mathrm{top}(l)$, then $\mathrm{depth}(r) = 1$, $r|q_f = l|q_f$, $l|q_f$ appears nowhere else in l or r, and $\mathrm{Var}(l) \setminus \mathrm{Var}(u) = \{l|q_f\}$. ◇

By also imposing that T_2 is saturated and that every literal appearing in T_2 that contains a constant is strictly flat (formally, that every clause is subsymbol-free from Σ^0), we obtain the following definition of saturation closure:

Definition 6.7 (Saturation-closure) Let $\Gamma \in \Omega_\Sigma$, a set of clauses S is Γ-*saturation-closed* if

- it is saturated,
- every clause in S is subsymbol-free from Σ^0,
- every clause in S is ordered flat,
- S is Γ-internally closed. $\diamond$

6.2 Restrictions on T_1

The restrictions imposed to T_1 prevent its clauses from interacting with T_2, and control the clauses generated by inferences involving these clauses.

Definition 6.8 (Weak flatness) A clause C is *weakly flat* if C only contains literals with terms of depth at most 1, and at least one non-ground literal $l \bowtie r$ which is not strictly flat. Furthermore, if C contains a literal $x \bowtie t$, then t is of depth 0. $\diamond$

Example 6.9 The clause $C = f(a) \simeq b \vee f(x) \not\simeq d$ is weakly flat.

Definition 6.10 (Variable-inactivity preservation) Given a function $\Gamma \in \Omega_\Sigma$, a clause C is Γ-*variable-inactive preserving* if and only if:

vip-1: For every variable $x \in \mathrm{Var}(C)$ and for every literal L in C which is not strictly flat, x is a variable of a term of depth 1 in L.

vip-2: If C contains a negative literal $l \not\simeq r$ with $\mathrm{top}(l) = \mathrm{top}(r) = f$, then C also contains a literal $x \simeq t$ such that either t is a variable and $\mathrm{Var}(C) \subseteq \{x, t\}$, or $\mathrm{Var}(C) = \{x\}$. Furthermore, let $q_f = \Gamma(f)$, then:

a. if t is a variable, then $\{x, t\} = \{l|q_f, r|q_f\}$,

b. if t is a constant, then there is a constant c (not necessarily equal to t) such that $\{x, c\} = \{l|q_f, r|q_f\}$.

A set of clauses S is Γ-variable-inactive preserving if every clause in S is. $\diamond$

Example 6.11 Let $\Sigma_I = \{\mathrm{Inj}\}$, where Inj is a predicate of arity 1, and consider the theory $\mathcal{A}_I$, based on the signature $\Sigma_\mathcal{A} \cup \Sigma_I$, which is axiomatized by axioms (1) and (2) of Example 6.3, and the following axiom denoted by (**inj**):

$$\mathrm{Inj}(x) \quad \leftrightarrow \quad \forall z, w.\ (z \not\simeq w \supset \mathrm{select}(x, z) \not\simeq \mathrm{select}(x, w)).$$

Intuitively, the predicate Inj is true for array a if and only if all the elements in a are pairwise distinct (a is injective). Consider the following clausal form, logically equivalent to $\mathrm{Inj}(a)$:

$$C \quad = \quad z \simeq w \vee \mathrm{select}(a, z) \not\simeq \mathrm{select}(a, w).$$

This clause contains a single literal that is not strictly flat, $L = \text{select}(a, z) \not\simeq \text{select}(a, w)$. We have $\text{Var}(C) = \{z, w\}$, and these two variables appear in terms of depth 1 in L. Let Γ be any function in Ω_Σ such that $\Gamma(\text{select}) = 2$. Since $z \simeq w$ is also a literal in C and condition *(vip.2.a)* holds on C, this clause is Γ-variable-inactive preserving.

Definition 6.12 (External closure) Let C be a clause, S' be a set of clauses, $\Gamma \in \Omega_\Sigma$, and for every $f \in \Sigma$, let $q_f = \Gamma(f)$. C is Γ-*externally closed from* S' if for every positive literal $l \simeq r$ in C such that $\text{top}(l) = f$,

ec.1: $\text{top}(l) = \text{top}(r)$,

ec.2: all the other literals in C are strictly flat,

ec.3: $l|q_f = r|q_f$ is the only variable in C and this variable appears nowhere else in l or r.

ec.4: every negative literal in a clause of S' is $\{f\}$-symbol-free.

A set of clauses S is Γ-*externally closed from* S' if every clause in S is. ◇

Example 6.13 Let $\Sigma_S = \{\text{Swap}\}$, where Swap is a predicate that has arity 4, and consider the theory $\mathcal{A}_S$, based on signature $\Sigma_\mathcal{A} \cup \Sigma_S$ and axiomatized by (1), (2) (the axioms of $\mathcal{A}$, see Example 6.3) and the following axiom denoted by (**swp**):

$$\begin{aligned} \text{Swap}(x, y, z_1, z_2) \quad \leftrightarrow \quad & \text{select}(x, z_1) \simeq \text{select}(y, z_2) \wedge \\ & \text{select}(x, z_2) \simeq \text{select}(y, z_1) \wedge \\ & \forall w.\ (w \not\simeq z_1 \wedge w \not\simeq z_2 \supset \text{select}(x, w) \simeq \text{select}(y, w)). \end{aligned}$$

Given constants b, b', i and i', the atom $\text{Swap}(b, b', i, i')$ is true if and only if b' is identical to b, except that the elements at indices i and i' are swapped. Consider the clause

$$D \quad = \quad w \simeq i \ \vee \ w \simeq i' \ \vee \ \text{select}(b, w) \simeq \text{select}(b', w).$$

Let Γ be any function in Ω_Σ such that $\Gamma(\text{select}) = 2$, and let $S' = \{(1), (2)\}$. It is simple to check that D satisfies conditions *(ec.1)* to *(ec.4)*, and is therefore Γ-externally closed from S'.

Definition 6.14 (Immunity) Given two sets of clauses S and S' and a function $\Gamma \in \Omega_\Sigma$, S is Γ-*immune from* S' if and only if

- every clause in S is weakly flat,
- every clause in S is Γ-variable-inactive preserving,
- S is Γ-externally closed from S'. ◇

Definition 6.15 (Interaction-freeness) Given two sets of clauses S and S' and a function $\Gamma \in \Omega_\Sigma$, S is Γ-*interaction-free from* S' if the following conditions hold: let f be a function symbol, let $p_f = \Gamma(f)$, and suppose that

- either f occurs at the same time in a positive literal of a clause in S and in a literal of a clause in S',
- or f occurs at the same time in a positive literal of a clause in S' and in a literal of a clause in S.

Then for all clauses $C \in S \cup S'$ containing a literal $L = l \bowtie r$, such that f appears in l or in r:

if.1: f only appears as the top symbol of l or r,

if.2: if $C \in S$, then $l|p_f$ is a constant, and if r is neither a constant nor a variable, then $r|p_f$ is a constant,

if.3: if $C \in S'$ and L is negative, then $l|p_f$ is a term of depth 1,

if.4: if $C \in S'$ and L is positive, then $u = l|p_f$ is a term of depth 1, and if r is neither a constant nor a variable, then $r|p_f$ is either a constant or a variable in $\mathrm{Var}(u)$. $\diamond$

Example 6.16 Consider $S = \{D\}$, where D is the clause of Example 6.13 and $S' = \{(1), (2)\}$. The only function symbol these sets have in common is select. Let Γ be any function in Ω_Σ such that $\Gamma(\mathrm{select}) = 1$. Then it is clear that D satisfies conditions *(if.1)* and *(if.2)*, and that the clauses in S' satisfy condition *(if.4)*. Thus, S is Γ-interaction-free from S'. Similarly, consider the clause C from Example 6.11, then $\{C\}$ is also Γ-interaction-free from S'.

6.3 Restrictions on T_g

We finally define the notion of *flat disconnection* for T_g.

Definition 6.17 (Positive flatness) A clause C is *positively flat* if each time C contains a positive literal which is not strictly flat, this literal is flat and all the other literals in C are strictly flat. $\diamond$

Example 6.18 Consider the following clauses:

$$\begin{aligned} C &= f(a) \simeq b \vee c \not\simeq d, \\ C' &= f(f(a)) \not\simeq b \vee f(c) \not\simeq d, \end{aligned}$$

The clauses C and C' are both positively flat.

Definition 6.19 (Negative disconnection) Let C be a clause and S' be a set of clauses. C is *negatively disconnected from* S' if whenever C contains a negative literal $l \not\approx r$ such that $\mathrm{depth}(l) \geq 2$, every positive literal of a clause in S' is $\Phi(C)$-symbol-free. A set of clauses S is *negatively disconnected from* S' if every clause in S is negatively disconnected from S'. $\diamond$

Definition 6.20 (Flat-disconnection) Given a clause C and a set of clauses S', C is *flat-disconnected from* S' if and only if C is

- positively flat,
- negatively disconnected from S'.

A set of clauses S is *flat-disconnected from* S' if every clause in S is. $\diamond$

Example 6.21 Any set of flattened ground clauses is flat-disconnected from any other set of clauses S'. Indeed, such a set is trivially positively flat; since all its negative literals are strictly flat, there is no literal $l \not\approx r$ with $\mathrm{depth}(l) \geq 2$, and the set is also negatively disconnected from S'.

6.4 Subterm-inactivity

We introduce the fundamental notion of *subterm-inactivity*, which guarantees the termination of the superposition calculus on the decision problem in the considered theory. We will also show that every subterm-inactive theory is variable-inactive.

Definition 6.22 (Subterm-inactivity) Let T_g, T_1 and T_2 be three disjoint sets of clauses. The tuple $\langle T_\mathrm{g}, T_1, T_2 \rangle$ is *subterm-inactive* if there exist two functions Γ and Γ' in Ω_Σ such that:

- T_g only contains ground clauses and is flat-disconnected from $T_1 \cup T_2$,
- T_1 is Γ-immune and Γ'-interaction-free from T_2,
- T_2 is Γ-saturation-closed.

A presentation of a theory $\mathcal{T}$ is *subterm-inactive* if there exists a partition $T_\mathrm{g} \uplus T_1 \uplus T_2$ of $\mathcal{T}$ such that $\langle T_\mathrm{g}, T_1, T_2 \rangle$ is subterm-inactive. $\diamond$

Since we can flatten any set of ground clauses, we can safely add it to a subterm-inactive presentation:

Proposition 6.23 *If $\mathcal{T}$ is a subterm-inactive presentation, then for every set of ground clauses S, there exists a set of ground clauses S' such that $S' \cup \mathcal{T}$ is equisatisfiable to $S \cup \mathcal{T}$, and $S' \cup \mathcal{T}$ is subterm-inactive.*

We will give several examples of subterm-inactive theories in the following section; before that, we state the main results we obtain under the subterm-inactivity hypothesis. Given a set of clauses $\mathcal{T} = T_\mathrm{g} \uplus T_1 \uplus T_2$ such that $\langle T_\mathrm{g}, T_1, T_2 \rangle$ is a subterm-inactive tuple:

Corollary 7.31: If D be a persistent clause generated by an inference in $\mathcal{T}$, then the inference is depth-preserving and:

- either D' is ground and $\langle T_{\rm g} \cup \{D'\}, T_1, T_2\rangle$ is subterm-inactive,
- or D' is not ground and $\langle T_{\rm g}, T_1 \cup \{D'\}, T_2\rangle$ is subterm-inactive.

Theorem 7.34: A fair $\mathcal{SP}_\succ$-strategy is a decision procedure for $\mathcal{T}$.

Corollary 7.36: $\mathcal{T}$ is variable-inactive.

7 Expansion inferences in subterm-inactive presentations

We now consider the different expansion inferences that can be carried out on clauses that appear in a subterm-inactive presentation $\langle T_{\rm g}, T_1, T_2\rangle$. For sake of clarity, we will use the term "paramodulation" to mean both paramodulation and superposition. We classify these inferences according to the category the clause considered belongs to: we first consider expansions from flat-disconnected clauses, then from immune clauses, and finally from saturation-closed clauses.

7.1 Expansions from flat-disconnected clauses

Unary inferences and paramodulations into flat-disconnected clauses.

Lemma 7.1 *Let S and S' be two sets of clauses such that S is flat-disconnected from S'. Let C and C' be two ground clauses in S. Then any unary expansion rule applied to C yields a clause D which is flat-disconnected from S', and such that* $\mathrm{Maxd}(D) \leq \mathrm{Maxd}(C)$. *Similarly, any binary expansion rule applied to C and C' yields a clause D' which is flat-disconnected, and is such that* $\mathrm{Maxd}(D) \leq \max\{\mathrm{Maxd}(C), \mathrm{Maxd}(C')\}$.

Proof. The result is immediate for the unary expansion rules since the mgu is empty. Now assume that

$$C = u \simeq v \vee C_1, \text{ and } C' = l'[u] \bowtie r' \vee C'_1.$$

We have to show that the clause $D = l'[v] \bowtie r' \vee C_1 \vee C'_1$ is flat-disconnected from S'.

- We prove that D is positively flat. Suppose that D contains a positive literal L which is not strictly flat. C is positively flat, so every literal in C_1 must be strictly flat by our goodness requirement on $\succ$. Suppose L is in C'_1, then since C' is positively flat, $l'[u] \bowtie r'$ must be strictly flat and by our goodness requirement on $\succ$, cannot be maximal in C'. Hence, the only literal in C' that can be positive without being strictly flat is $l'[u] \bowtie r'$,and since C' is positively flat, this literal is flat and every literal in C'_1 is strictly flat. Therefore, D is positively flat.
- We now prove that D is negatively disconnected from S'. Suppose D contains a literal $L = l \not\simeq r$ such that $\mathrm{depth}(l) \geq 2$. Since every literal in C must be flat, L is necessarily a literal in C' which is negatively disconnected from S'. The

only function symbol that may appear in C is the top symbol of u because C is positively flat, and this function symbol also appears in C'. Since C and C' are negatively disconnected from S', this function symbol does not appear in S', thus, D is also negatively disconnected from S'. ∎

Example 7.2 Consider the three following clauses:

$$\begin{aligned} C &= f(a) \simeq b \vee c \not\simeq d, \\ C_1 &= f(a) \simeq c \vee a \simeq d, \\ C_2 &= g(f(a)) \not\simeq g(f(b)), \end{aligned}$$

and let S' be a set of clauses such that $\{C, C_1, C_2\}$ is flat-disconnected from S'. The respective paramodulations of C into C_1 and C_2 yield

$$\begin{aligned} D_1 &= b \simeq c \vee a \simeq d \vee c \not\simeq d, \\ D_2 &= g(b) \not\simeq g(f(b)) \vee c \not\simeq d \end{aligned}$$

Both D_1 and D_2 are flat-disconnected from S'.

Paramodulations into immune clauses.

Proposition 7.3 *Let $\Gamma \in \Omega_\Sigma$, S be a set of clauses and C be a clause that is Γ-immune from S. Suppose a unary or binary inference step can be applied with C as a premise. Then none of the concerned literals in C can be strictly flat.*

Proof. The strictly flat literals in C are of the form $c \bowtie c'$ and $x \bowtie t$, where t is a constant or a variable. Since C is weakly flat, it contains at least one non-ground literal L which is not strictly flat, and by our goodness requirement on $\succ$, it is clear that $c \bowtie c'$ cannot be maximal in C. Now consider a literal $x \bowtie t$. Since C is also Γ-variable-inactive preserving, x must be a variable of a term s in L of depth 1, and since $\succ$ is a CSO, we have $x \prec s$. If t is a variable, then it must also appear as a variable of a term of depth 1 in L, and if t is a constant, by our goodness requirement on $\prec$, we have $t \prec s$. Thus, in both cases, we have $\{x, t\} \prec L$. ∎

Proposition 7.4 *Let S and S' be two sets of clauses and $\Gamma \in \Omega_\Sigma$ such that S is Γ-immune from S'. Let $l \bowtie r$ be a maximal literal in $C \in S$ and suppose that* $\mathrm{depth}(r) = 0$. *Then l is of depth 1 and $r \prec l$.*

Proof. By Proposition 7.3, $l \bowtie r$ cannot be strictly flat, and since C is weakly flat, l is of depth 1. If r is a constant, then $r \prec l$ by our goodness requirement on $\succ$, and if r is a variable, then it is a variable of l by condition vip.1 of Definition 6.10, and once again we have $r \prec l$. ∎

Lemma 7.5 *Let C and C' be two clauses, S be a set of clauses and $\Gamma \in \Omega_\Sigma$, such that:*

- *C is ground, positively flat and such that its maximal literal $L = l \simeq r$ is not strictly flat,*
- *C' is Γ-immune from S.*

Let $L' = l' \bowtie r'$ be a maximal literal in C', and let D be the result of paramodulating C into C' with mgu σ. Then σ is strictly flat and D, which is of the form $r \bowtie r'\sigma \vee D'$, is positively flat.

PROOF. Since C is positively flat, every literal in C other than L is strictly flat. Also, we do not paramodulate into variables and C' is weakly flat by Definition 6.14, hence the term l' is of depth 1 and σ is strictly flat by Proposition 1.2 (1). Suppose D contains a positive literal that is not strictly flat. Then C' must also contain a positive literal that is not strictly flat, and by condition *(ec.2)* of Definition 6.12, all the other literals in C' must be strictly flat. By Proposition 7.3, this literal must therefore be L', and the only literal in D that may not be strictly flat is $r\sigma \simeq r'\sigma$ (which is equal to $r \simeq r'\sigma$ since C is ground). Since C' is weakly flat, r' is of depth at most 1, and since σ is strictly flat, $r'\sigma$ is also of depth at most 1 by Proposition 1.2 (2). Since L is flat, r must be of depth 0, hence the result. ∎

Lemma 7.6 *Let C and C' be two clauses, S be a set of clauses, and $\Gamma \in \Omega_\Sigma$, such that:*

- *C is ground, positively flat and its maximal literal $L = l \simeq r$ is not strictly flat,*
- *C' is Γ-immune from S.*

Let $L' = l' \bowtie r'$ be a maximal literal in C' and $D = r \bowtie r'\sigma \vee D'$ be the clause obtained by paramodulating C into C' with mgu σ. If D is not ground, then $r \bowtie r'\sigma$ is negative and not strictly flat, and D is Γ-variable-inactive preserving.

PROOF. Let $L' = l' \bowtie r'$ be the maximal literal considered in C', by Lemma 7.5, σ is strictly flat. By condition *(vip.1)* of Definition 6.10, we have $\mathrm{Var}(C') \subseteq \mathrm{Var}(L')$, and since $l'\sigma$ is ground because l is ground, we have $\mathrm{Var}(D) \subseteq \mathrm{Var}(C'\sigma) = \mathrm{Var}(r'\sigma)$. Since D is not ground, we cannot have $\mathrm{Var}(r') \subseteq \mathrm{Var}(l')$, and by condition *(ec.3)* of Definition 6.12, L' is negative, since otherwise we would have $\mathrm{Var}(C) \subseteq \mathrm{Var}(l)$, and D would be ground. Therefore, the literal $r \not\simeq r'\sigma$ is not ground.

Let M be a literal in D other than $r \not\simeq r'\sigma$ that is not strictly flat. Since C is positively flat, there must exist a literal L_1 in C' such that $M = L_1\sigma$. Since C' is Γ-variable-inactive preserving and $\mathrm{Var}(D) \subseteq \mathrm{Var}(C'\sigma)$, for every $x \in \mathrm{Var}(D)$, x is a variable in a term of depth 1 in M. We now show that the same condition holds for $r \not\simeq r'\sigma$. Since $\mathrm{Var}(r'\sigma) \neq \emptyset$, r' is not a constant, and it cannot be a variable either, otherwise it would appear in $l'\sigma$ by condition *(vip.1)* of Definition 6.10. Thus, r' is of depth 1, and so is $r'\sigma$. Therefore, $r \not\simeq r'\sigma$ is not strictly flat, and condition *(vip.1)* holds on D.

Let $u \not\simeq v$ be a literal in D such that $\mathrm{top}(u) = \mathrm{top}(v) = f$. This literal cannot be $r \bowtie r'\sigma$ because r is of depth 0. Since C is positively flat, there must exist a literal

$u' \not\simeq v'$ in C' such that $u \not\simeq v = (u' \not\simeq v')\sigma$, and $\text{top}(u') = \text{top}(v') = f$. By condition *(vip.2)*, C' contains a literal $x \simeq t$, with $\text{Var}(C') \subseteq \{x, t\}$, hence, D contains a literal $x\sigma \simeq t\sigma$ with $\text{Var}(D) \subseteq \{x\sigma, t\sigma\}$. Let $q_f = \Gamma(f)$, if t is a variable, then by condition *(vip.2.a)*, $\{x, t\} = \{u'|q_f, v'|q_f\}$, and therefore, $\{x\sigma, t\sigma\} = \{u|q_f, v|q_f\}$. It is simple to check that conditions *(vip.2.a)* or *(vip.2.b)* hold on D, depending on what the images of x and t by σ are. If t is a constant, then by condition *(vip.2.b)*, we have $\text{Var}(C') = \{x\}$, and since D is not ground, $\text{Var}(D) = \{x\}$. Therefore, σ must be empty, $u = u'$ and $v = v'$. So, condition *(vip.2.b)* holds on D which is Γ-variable-inactive preserving. ∎

Theorem 7.7 *Let C be a ground clause, S and S' be two sets of clauses, and $\Gamma \in \Omega_\Sigma$, such that:*

- *C is positively flat with maximal literal $L = l \simeq r$,*
- *S is Γ-immune from S'.*

Let C' be a clause in S, and D be the result of paramodulating C into C' with mgu σ. Then D is flat-disconnected from $S \cup S'$, and if it is not ground, then $S \cup \{D\}$ is Γ-immune from S'.

Proof. Let $L = l \simeq r$ be the maximal literal in C, $L' = l' \bowtie r'$ be the maximal literal in C' and first suppose that L is strictly flat. Then by our goodness requirement on $\succ$, C must be strictly flat. Furthermore, since we do not paramodulate into variables, σ must be empty, and it is easy to check that D and therefore $S \cup \{D\}$ are Γ-immune from S'.

Now consider the case where L is not strictly flat. Then by Lemma 7.5, σ is strictly flat and D is positively flat. By Proposition 1.5, $\text{Maxd}(D) \leq \max\{\text{Maxd}(C), \text{Maxd}(C')\}$, hence $\text{Maxd}(D) \leq 1$, D is therefore weakly flat and trivially negatively disconnected from $S \cup S'$.

If D is not ground, then by Lemma 7.6, $r \not\simeq r'\sigma$ is not strictly flat and D is Γ-variable-inactive preserving. Suppose D contains a positive literal that is not strictly flat. Then by condition *(ec.2)* of Definition 6.12, $r \not\simeq r'\sigma$ would have to be strictly flat, which is not the case. Thus, D cannot contain any positive literal that is not strictly flat. Hence, D is trivially Γ-externally closed from S' and $S \cup \{D\}$ is Γ-immune from S'. ∎

Corollary 7.8 *Under the hypotheses of Theorem 7.7, if D is not ground and there exists a $\Gamma' \in \Omega_\Sigma$ such that S is Γ'-interaction-free from S', then $S \cup \{D\}$ is also Γ'-interaction-free from S'.*

Proof. For $f \in \Sigma$, let $p_f = \Gamma'(f)$. By Lemma 7.6, D contains the literal $r \not\simeq r'\sigma$ that is not strictly flat, and by Lemma 7.5, D is positively flat. From these two facts we conclude that D does not contain any positive literal with a term of depth 1. Furthermore, D is weakly flat, hence every term appearing in D is of depth at most 1 and condition *(if.1)* of Definition 6.15 trivially holds on D. Let $u \bowtie v$ be a literal in D such that f appears

in u or in v. If $u \bowtie v = r \not\simeq r'\sigma$, then r is a constant since $l \simeq r$ is flat and ground, and by hypothesis, so is $r'|p_f$. Otherwise, there must exist a literal $u' \bowtie v'$ in C' such that $u \bowtie v = (u' \bowtie v')\sigma$, and it is easy to check that $u \bowtie v$ verifies the required conditions and that condition *(if.2)* holds on D. ∎

Example 7.9 Consider the following two clauses, from the theory of arrays with injectivity:

$$\begin{aligned} C &= \text{select}(a, i) \simeq e \vee i \not\simeq i', \\ C' &= z \simeq w \vee \text{select}(a, z) \not\simeq \text{select}(a, w). \end{aligned}$$

Let Γ be any function in Ω_Σ such that $\Gamma(\text{select}) = 2$ and S' be any set of clauses such that $\{C'\}$ is Γ-immune from S'. Let D be the clause generated by the paramodulation of C into C', we have

$$D = i \simeq w \vee e \not\simeq \text{select}(a, w).$$

This clause is not ground, and $\{C', D\}$ is Γ-immune from S'.

Paramodulations into saturation-closed clauses.

Proposition 7.10 *Let C be a positively flat clause with maximal literal $L = l \simeq r$ that is not strictly flat, and C' be an ordered flat clause with maximal literal $L' = l' \bowtie r'$. Let D be the result of paramodulating C into C' with mgu σ, then σ is strictly flat and the only literal in D that may not be strictly flat is $(l'[r] \bowtie r')\sigma$.*

Proof. Since we do not paramodulate into variables, l must be unifiable with a subterm u' of depth 1 in l'. By Proposition 1.2 (1), σ is strictly flat, thus, the only literal in D that is not strictly flat is $(l'[r] \bowtie r')\sigma$. ∎

Lemma 7.11 *Let C be a clause that is ground and flat-disconnected from a set of clauses S and let $\Gamma \in \Omega_\Sigma$. Suppose that S is Γ-saturation-closed and let C' be a clause in S. Let $L = l \simeq r$ and $L' = l' \bowtie r'$ be maximal literals in C and C' respectively, suppose L is not strictly flat and let D be the result of paramodulating C into C'. If D is ground, then every function symbol appearing in D also appears in L and D is flat-disconnected from S.*

Proof. By Proposition 7.10, the only literal in D that may not be strictly flat is $(l'[r] \bowtie r')\sigma$. First suppose that L' is positive, we show that $(l'[r] \simeq r')\sigma$ is flat. By condition *(icp.1)* of Definition 6.12, l' contains a unique subterm u' of depth 1, and since $l \simeq r$ is flat, $l'[r]$ is of depth 1. If $\text{top}(r') = \text{top}(l')$, then by condition *(icp.4)*, $r'|q_f$ and $l'|q_f$ are the same variable, let x be this variable, and $\text{Var}(l') \setminus \text{Var}(u') = \{x\}$. Since σ is the mgu of u' and l, we have $x\sigma = x$ and D cannot be ground, a contradiction. Therefore, we must have $\text{top}(r') \neq \text{top}(l')$, and by condition *(icp.3)*, $r'\sigma$ is a constant. Hence, $(l[r] \simeq r')\sigma$ is flat and D is flat-disconnected from S.

If L' is negative, then by condition *(icn.2)*, no function symbol appearing in L' can appear in a positive literal of a clause in S. Since $(l'[r] \not\simeq r')\sigma$ contains the same function symbols as L', D is negatively disconnected from S. Thus, D is again flat-disconnected from S. ∎

Example 7.12 Consider the theory of arrays, and let C and C' be the two following clauses:

$$\begin{aligned} C &= \text{store}(a, i, e) \simeq b \vee a \simeq b \vee e \not\simeq e', \\ C' &= \text{select}(\text{store}(x, z, v), z) \simeq v. \end{aligned}$$

Then C is flat-disconnected from $\{C'\}$, which is Γ-saturation-closed for any $\Gamma \in \Omega_\Sigma$. Let D be the result of the paramodulation of C into C', we have

$$D = \text{select}(b, i) \simeq e \vee a \simeq b \vee e \not\simeq e'.$$

This clause is ground and flat-disconnected from $\{C'\}$.

Corollary 7.13 *Under the hypotheses of Lemma 7.11, if S' is a set of clauses that is Γ-immune from S, then D is negatively disconnected from S'.*

Proof. Let $D' \in S'$ be a clause containing a positive literal $M = u \simeq v$ such that $\text{top}(u) = f$. Since D' is weakly flat, u and v are of depth at most 1 and by condition *(ec.1)* of Definition 6.12, f is the only function symbol that appears in M. Suppose that f also appears in a negative literal in D, then by Proposition 7.10, this literal must be $(l'[r] \not\simeq r')\sigma$ and L', the maximal literal of C', must be negative. By Lemma 7.11, f must also appear in L'. This is impossible, since by condition *(ec.4)*, L' must be $\{f\}$-symbol-free. ∎

Lemma 7.14 *Let C be a clause that is ground and flat-disconnected from a set of clauses S' and let $\Gamma \in \Omega_\Sigma$. Suppose that S' is Γ-saturation-closed, and let C' be a clause in S'. Let $L = l \simeq r$ and $L' = l' \bowtie r'$ be the maximal literals in C and C' respectively, and let D be the result of paramodulating C into C'. If D is not ground, then it is Γ-immune from S'.*

Proof. Let u' be the subterm of l' that l is unifiable with. Since $l\sigma = u'\sigma$ and $\text{Var}(D) \subseteq \text{Var}(C'\sigma)$, L' must be positive, otherwise by condition *(icn.1)* of Definition 6.12, we would have $\text{Var}(D) \subseteq \text{Var}(u'\sigma) = \emptyset$ and D would be ground. By condition *(icp.1)*, we have $\text{depth}(l') = 2$, and by conditions *(icp.3)* and *(icp.4)*, we have $\text{depth}(r') \leq 1$. Since r is a constant, we conclude that both $(l'[r])\sigma$ and $r'\sigma$ are of depth at most 1, and $(l'[r] \simeq r')\sigma$ is not strictly flat. Since C is ground, we have $\text{Var}(D) = \text{Var}(C'\sigma)$ and by condition *(icp.2)*, $\text{Var}(C'\sigma) \subseteq \text{Var}(l'\sigma)$. Since D is not ground, necessarily, $\text{Var}(C') \not\subseteq \text{Var}(u')$, and by condition *(icp.3)*, we must have $\text{top}(l') = \text{top}(r')$. Therefore, $\text{top}(l'[r]) = \text{top}(r')$, and condition *(ec.1)* holds. Also, $(l'[r] \simeq r')\sigma$ is not ground, and since all the other literals in D are strictly flat by Proposition 7.10, D is weakly flat and condition *(ec.2)* holds on D.

By condition *(icp.4)*, we have $\mathrm{Var}(l') \setminus \mathrm{Var}(u') = \{l'|q_f\} = \{r'|q_f\}$, so that $\{(l'[r])|q_f\} = \mathrm{Var}(l'[r]\sigma) = \mathrm{Var}(r'\sigma)$, and $(l'[r])|q_f = r'|q_f$. Hence, $\mathrm{Var}(C'\sigma) = \mathrm{Var}(l') \setminus \mathrm{Var}(u') = \{(l'[r])|q_f\}$. Therefore, condition *(ec.3)* also holds on D.

Since $l \simeq r$ is flat, the function symbols appearing in $(l'[r] \simeq r')\sigma$ also appear in L', which is positive. By condition *(icn.2)*, for every negative literal M appearing in a clause of S', L' is $\Phi(M)$-symbol-free. Thus, every such M is $\{f\}$-symbol-free, and condition *(ec.4)* holds on D, which is Γ-externally closed from S'.

The only literal in D that is not strictly flat is $(l'[r] \simeq r')\sigma$, and it is such that every variable in D appears in one of its terms of depth 1. Thus, condition *(vip.1)* holds on D. Since condition *(vip.2)* trivially holds on D, this clause is Γ-variable-inactive preserving and therefore Γ-immune from S'. ∎

Example 7.15 Consider the clause C of Example 7.12, let

$$C'' \quad = \quad z \simeq w \vee \mathrm{select}(\mathrm{store}(x, z, v), w) \simeq \mathrm{select}(x, w),$$

and let D' be the clause generated by the paramodulation of C into C''. We therefore have

$$D' \quad = \quad i \simeq w \vee \mathrm{select}(b, w) \simeq \mathrm{select}(a, w) \vee a \simeq b \vee e \not\simeq e'.$$

For any $\Gamma \in \Omega_\Sigma$ such that $\Gamma(\mathrm{select}) = 2$, D' is Γ-immune from $\{C'\}$.

Lemma 7.16 *Under the hypotheses of Lemma 7.14, for any set of clauses S and $\Gamma' \in \Omega_\Sigma$ such that S is Γ'-interaction-free from S', $S \cup \{D\}$ is Γ'-interaction-free from S'.*

Proof. By Proposition 7.10, the only literal in D that is not strictly flat is $(l'[r] \bowtie r')\sigma$, hence any function symbol appearing in D also appears in this literal. By Lemma 7.14 this literal is positive and contains terms of depth at most 1 and by condition *(ec.1)* of Definition 6.12, we have $\mathrm{top}(l') = \mathrm{top}(r')$. Let $f = \mathrm{top}(l')$, this function symbol is therefore the only one to appear in D, and it appears as the top symbol of $(l'[r])\sigma$ and of $r'\sigma$, hence condition *(if.1)* holds on D. Let $p_f = \Gamma'(f)$, by condition *(if.4)* we have $l'|p_f = u'$, and since $l \simeq r$ is flat, r must be a constant. Therefore, $((l'[r])|p_f)\sigma$ is a constant. Still by condition *(if.4)*, if r' is not a constant, then $r'|p_f$ is either a constant or a variable in $\mathrm{Var}(u')$, so $(r'\sigma)|p_f$ is a constant, and condition *(if.2)* holds on D. Thus, $S \cup \{D\}$ is Γ'-interaction-free from S'. ∎

We now prove that when a presentation $T_\mathrm{g} \uplus T_1 \uplus T_2$ is such that $\langle T_\mathrm{g}, T_1, T_2 \rangle$ is subterm-inactive and an inference involving a clause in T_g is applied, then subterm-inactivity is preserved by adding the generated clause to T_g or T_1.

Theorem 7.17 *Let $\mathcal{T} = T_\mathrm{g} \uplus T_1 \uplus T_2$ be a set of clauses such that $\langle T_\mathrm{g}, T_1, T_2 \rangle$ is subterm-inactive, and let C be a clause in T_g with maximal literal L.*

- *If D is a persistent clause generated by a unary inference on C, then $\langle T_\mathrm{g} \cup \{D\}, T_1, T_2 \rangle$ is subterm-inactive, and* $\mathrm{Maxd}(D) \leq \mathrm{Maxd}(C)$.

- *Let C' be a clause in $T_g \cup T_1 \cup T_2$, let D be the clause obtained after the paramodulation of C into C', and suppose that D is persistent. Then either D is ground and $\langle T_g \cup \{D\}, T_1, T_2\rangle$ is subterm-inactive, or $\langle T_g, T_1 \cup \{D\}, T_2\rangle$ is subterm-inactive. Furthermore, in both cases, we have* $\mathrm{Maxd}(D) \leq \max\{\mathrm{Maxd}(C), \mathrm{Maxd}(C')\}$.

PROOF. Let $L = l \simeq r$ be the maximal literal in C. If D is obtained by a unary inference on C or a paramodulation into $C' \in T_g$, then $T_g \cup \{D\}$ is trivially ground, and this set is flat-disconnected from $T_1 \cup T_2$ by Lemma 7.1.

Suppose $C' \in T_1$ and D is ground. Since we do not paramodulate into variables, L cannot be strictly flat, otherwise the mgu σ would be empty and D would not be ground. By Lemma 7.5, D is positively flat and σ is strictly flat. All the other literals in D must be strictly flat, and all the literals in C' are of depth at most 1 because C' is weakly flat. Therefore, every literal in D is of depth at most 1, and D is trivially negatively disconnected from $T_1 \cup T_2$. Therefore, $T_g \cup \{D\}$ is flat-disconnected from $T_1 \cup T_2$. If D is not ground, then by Theorem 7.7, $T_1 \cup \{D\}$ is Γ-immune from T_2. By Corollary 7.8, this set is also Γ-interaction-free from T_2, and we have the result.

Let $C' \in T_2$, since C' is subsymbol-free from Σ^0, L cannot be strictly flat. If D is ground, then by Lemma 7.11 D is flat-disconnected from T_2. By Corollary 7.13, D is negatively disconnected from T_1, hence $T_g \cup \{D\}$ is flat-disconnected from $T_1 \cup T_2$. If D is not ground, then by Lemma 7.14 D is Γ-immune from T_2, and by Lemma 7.16, D is Γ-interaction-free from T_2. Hence $T_1 \cup \{D\}$ is Γ-immune from T_2. By Proposition 7.10, σ is strictly flat and $\mathrm{Maxd}(D) \leq \max\{\mathrm{Maxd}(C), \mathrm{Maxd}(C')\}$. ∎

7.2 Expansions from immune clauses

Unary inferences and paramodulations into immune clauses

Proposition 7.18 *Let S and S' be two sets of clauses and $\Gamma \in \Omega_\Sigma$ such that S is Γ-interaction-free from S'. Let C be a clause in S with a maximal literal L, and C' a clause in S' with a maximal literal L'. If l is a maximal term of depth 1 in L and l' is a maximal term in L', then l is not unifiable with any non-variable subterm of l'.*

PROOF. Let $f = \mathrm{top}(l)$ and assume that l is unifiable with a non-variable subterm of l'. Then the top symbol of this subterm must be f, and by condition *(if.1)*, f can only appear as the top symbol of l'. Hence, l must be unifiable with l'. Let $p_f = \Gamma(f)$, by condition *(if.2)*, $l|_{p_f}$ is a constant, and by conditions *(if.3)* and *(if.4)*, $l'|_{p_f}$ must be a term of depth 1. Thus, l and l' cannot be unifiable, we obtain a contradiction. ∎

Lemma 7.19 *Let S and S' be two sets of clauses, $\Gamma \in \Omega_\Sigma$ such that S is Γ-immune from S', and let C be a clause in S. Then no equational factoring inference can be applied to C.*

PROOF. Suppose we can apply an equational factoring inference step in C, between $L = l \simeq r$ and $L' = l' \simeq r'$. By Proposition 7.3, one of L or L' is not strictly flat, and since there is at most one positive literal that is not strictly flat by condition *(ec.2)*, the

other literal must be strictly flat. Without loss of generality, suppose L' is strictly flat and L is not. In order for l and l' to be unifiable, l' must be a variable, and by condition *(vip.1)*, l' must appear in l or r. By condition *(ec.3)*, l' appears in l and therefore l and l' are not unifiable. We obtain a contradiction. ∎

Lemma 7.20 *Let S and S' be two sets of clauses and $\Gamma \in \Omega_\Sigma$ such that S is Γ-immune from S', and let C and C' be two clauses in S. Let $L = l \simeq r$ and $L' = l' \simeq r'$ be maximal literals appearing respectively in C and C', suppose l and l' are unifiable, and let σ be their mgu. Then $\emptyset \neq \mathrm{Var}(r\sigma) = \mathrm{Var}(r'\sigma)$. Furthermore, if $\mathrm{top}(r) = f$ and $q_f = \Gamma(f)$, then $(r\sigma)|_{q_f}$ and $(r'\sigma)|_{q_f}$ are the same variable.*

Proof. By Proposition 7.3, neither L nor L' is strictly flat, and we have $\mathrm{top}(l) = \mathrm{top}(l') = f$. By condition *(ec.1)* we have $\mathrm{top}(r) = \mathrm{top}(l) = \mathrm{top}(l') = \mathrm{top}(r')$. By condition *(ec.3)*, $l|_{q_f}$ and $r|_{q_f}$ (resp. $l'|_{q_f}$ and $r'|_{q_f}$) are the same variable which appears nowhere else in l (resp. l'). Since $l\sigma = l'\sigma$, $(r|_{q_f})\sigma$ and $(r'|_{q_f})\sigma$ must be the same variable, and $\mathrm{Var}(r\sigma) \neq \emptyset$. Finally, again by condition *(ec.3)*, $\mathrm{Var}(l) = \mathrm{Var}(r)$ and $\mathrm{Var}(r') = \mathrm{Var}(l')$, so $\mathrm{Var}(r\sigma) = \mathrm{Var}(r'\sigma)$. ∎

This lemma also implies that when D is ground, then L' must be negative:

Corollary 7.21 *Let S and S' be two sets of clauses and $\Gamma \in \Omega_\Sigma$ such that S is Γ-immune from S', and let C and C' be two clauses in S with respective maximal literals L and L'. If D, the clause resulting of the paramodulation of C into C' is ground, then L' must be negative.*

Proposition 7.22 *Let S and S' be two sets of clauses and $\Gamma \in \Omega_\Sigma$ such that S is Γ-immune from S'. Let C and C' be two clauses in S with respective maximal literals $L = l \simeq r$ and $L' = l' \bowtie r'$. Let D be the clause obtained after the paramodulation of C into C' with mgu σ, then we have $\mathrm{Var}(D) = \mathrm{Var}(C'\sigma) = \mathrm{Var}(r\sigma \bowtie r'\sigma)$.*

Proof. By hypothesis, $\mathrm{Var}(D) = \mathrm{Var}(C\sigma) \cup \mathrm{Var}(C'\sigma)$. By condition *(ec.3)*, $\mathrm{Var}(C\sigma) = \mathrm{Var}(l\sigma)$, and since $l\sigma = l'\sigma$, we have $\mathrm{Var}(C\sigma) = \mathrm{Var}(l'\sigma)$, thus $\mathrm{Var}(D) = \mathrm{Var}(C'\sigma)$.

By Proposition 7.3, both l and l' are of depth 1 and by condition *(vip.1)*, we therefore have $\mathrm{Var}(C'\sigma) = \mathrm{Var}(l'\sigma) \cup \mathrm{Var}(r'\sigma)$. Since $\mathrm{Var}(l) = \mathrm{Var}(r)$ and $l\sigma = l'\sigma$, we deduce that $\mathrm{Var}(C'\sigma) = \mathrm{Var}(r\sigma) \cup \mathrm{Var}(r'\sigma) = \mathrm{Var}(r\sigma \bowtie r'\sigma)$. ∎

Lemma 7.23 *Let S and S' be two sets of clauses, and $\Gamma \in \Omega_\Sigma$ such that S is Γ-immune from S', and suppose a reflection inference step is applied to a clause $C \in S$. Then the resulting clause D can either be deleted by a Deletion inference rule, or is ground and flat-disconnected from $S \cup S'$.*

Proof. Suppose that D is obtained from C after a reflection inference step on $L = l \not\simeq r$. If the corresponding mgu is empty, then l and r are syntactically identical, and they are both of depth 1 by Proposition 7.3. Let $f = \mathrm{top}(l)$, then $l|_{q_f} = r|_{q_f}$ since l and r are unifiable, and by condition *(vip.2)*, C contains the literal $l|_{q_f} \simeq r|_{q_f}$ and so does D.

Therefore, D can be deleted by the Deletion rule. We now assume that the corresponding mgu σ is nonempty.

If $\mathrm{top}(l) \neq \mathrm{top}(r)$, the only case where l and r can be unifiable is when one of them is a variable. By the weakly flat hypothesis, the other one must be of depth 0 and Proposition 7.3 proves that this case is not possible.

Now suppose that $\mathrm{top}(l) = \mathrm{top}(r)$. By Proposition 7.3, we must have $\mathrm{top}(l) = \mathrm{top}(r) = f$, where f is a function symbol. Still by the weakly flat hypothesis, we have $\mathrm{depth}(l) = \mathrm{depth}(r) = 1$, and σ is strictly flat by Proposition 1.2 (1). There are two cases to consider:

- Suppose C contains a literal $x \simeq t$ where t is a variable. Then by condition *(vip.2.a)* we have $\{x, t\} = \{l|q_f, r|q_f\}$, and since $(l|q_f)\sigma = (r|q_f)\sigma$, the clause D can be deleted by the Deletion rule.

- Suppose C contains a literal $x \simeq t$ where t is a constant. Then by condition *(vip.2.b)*, $x\sigma$ must be a constant, and by condition *(vip.2)*, D is ground. By condition *(ec.2)*, D cannot contain any positive literal which is not strictly flat, and therefore, D is positively flat. Since C is weakly flat, it cannot contain any term of depth strictly greater than 1, and since σ is strictly flat, neither can D. Thus, D is trivially flat-disconnected from $S \cup S'$. ∎

Example 7.24 Let C be the following clause:

$$z \simeq w \vee \mathrm{select}(a, z) \not\simeq \mathrm{select}(a, w),$$

then the reflection rule applied to C yields the clause $z \simeq z$, which can be deleted by the Deletion inference rule.

Lemma 7.25 *Let S and S' be two sets of clauses and $\Gamma \in \Omega_\Sigma$ such that S is Γ-immune from S', and let C and C' be two clauses in S with respective maximal literals $L = l \simeq r$ and $L' = l' \bowtie r'$. Let D be the clause obtained after the paramodulation of C into C'. If D is not ground, then it is weakly flat and $S \cup \{D\}$ is Γ-externally closed from S'.*

Proof. By Proposition 7.3, neither L nor L' is strictly flat, and by Proposition 7.4, l and l' are of depth 1, hence the mgu σ is strictly flat. By Proposition 7.22, we have $\mathrm{Var}(D) = \mathrm{Var}(r\sigma \bowtie r'\sigma)$. By condition *(ec.1)*, r is of depth at least 1, and since C is weakly flat, it is of depth 1. Hence, $r\sigma$ is also of depth 1 and $r\sigma \bowtie r'\sigma$ is not strictly flat. In order to prove that D is weakly flat, we need to show that $r'\sigma$ is not a variable. If it were, then r' would also be a variable, and since C' is weakly flat, l' would be of depth 0, a contradiction.

Suppose D contains a positive literal that is not strictly flat. By condition *(ec.3)*, all the literals in C other than L are strictly flat, hence, C' must contain a positive literal that is not strictly flat. Since L' is not strictly flat, it is necessarily a positive literal in C', and $r\sigma \simeq r'\sigma$ is the only literal in D not to be strictly flat. Therefore, condition *(ec.2)* holds on D.

- By condition *(ec.1)*, we have $\text{top}(l) = \text{top}(r)$ and $\text{top}(l') = \text{top}(r')$. Since $l\sigma = l'\sigma$, condition *(ec.1)* holds on D.

- By condition *(ec.3)*, $\text{Var}(C) = \{r|q_f\}$, $\text{Var}(C') = \{r'|q_f\}$, and $(r|q_f)\sigma = (l|q_f)\sigma = (l'|q_f)\sigma = (r'|q_f)\sigma$. Since $(r|q_f)\sigma$ appears nowhere else in r or r', condition *(ec.3)* holds on D.

- Since condition *(ec.4)* holds on C, f appears in no negative literal of a clause in S', and the same condition holds on D.

Therefore, D is Γ-externally closed from S'. ∎

Lemma 7.26 *Let S and S' be two sets of clauses and $\Gamma \in \Omega_\Sigma$ such that S is Γ-immune from S', and let C and C' be two clauses in S with respective maximal literals $L = l \simeq r$ and $L' = l' \bowtie r'$. Let D be the clause obtained after the paramodulation of C into C'. If D is not ground, then it is Γ-variable-inactive preserving.*

Proof. In what follows, for any $f \in \Sigma$, let $q_f = \Gamma(f)$. By Proposition 7.22 we have $\text{Var}(D) = \text{Var}(C'\sigma)$. Let $C' = l' \bowtie r' \vee C'_1$, since C is internally closed from S', by condition *(ec.2)* the literals in D that are not strictly flat are $r\sigma \bowtie r'\sigma$ and those in C'_1. We show that condition *(vip.1)* holds on $r\sigma \bowtie r'\sigma$. Suppose $\text{depth}(r'\sigma) = \text{depth}(r') = 0$, then since C' is variable-inactive preserving, by condition *(vip.1)*, we have $\text{Var}(C') = \text{Var}(l')$, and $\text{Var}(D) = \text{Var}(l'\sigma) = \text{Var}(l\sigma)$. Since C is Γ-externally closed from S', by condition *(ec.3)*, we have $\text{Var}(l\sigma) = \text{Var}(r\sigma)$. Thus, $\text{Var}(D) = \text{Var}(r\sigma)$, and since $r\sigma$ is not strictly flat, condition *(vip.1)* holds on $r\sigma \bowtie r'\sigma$. Since C' is Γ-variable-inactive preserving, condition *(vip.1)* holds on D.

Suppose D contains a negative literal $u\sigma \not\simeq v\sigma$ other than $r\sigma \bowtie r'\sigma$, such that $\text{top}(u) = \text{top}(v) = f$. Then since C and C' are Γ-externally closed from S', by condition *(ec.2)*, L' cannot be positive and $u \not\simeq v$ must be a literal in C'. Since C' is Γ-variable-inactive preserving, by condition *(vip.2)*, it contains a literal $x \simeq t$ such that $\text{Var}(C') \subseteq \{x, t\}$, and either $\{x, t\} = \{l|q_f, r|q_f\}$, or there exists a constant c such that $\{x, c\} = \{l|q_f, r|q_f\}$. We have $\text{Var}(C'\sigma) \subseteq \{x\sigma, t\sigma\}$, and conditions *(vip.2.a)* and *(vip.2.b)* hold, depending on whether $t\sigma$ is a variable or a constant.

Now suppose that $r\sigma \bowtie r'\sigma$ is negative and that $\text{top}(r) = \text{top}(r') = f$.

- Since $\text{top}(r) = \text{top}(l)$ and $\text{top}(l) = \text{top}(l')$, we must have $\text{top}(l') = \text{top}(r')$, and by condition *(vip.2)*, there exists a literal $y \simeq s$ such that $\text{Var}(C') \subseteq \{y, s\}$. If s is a variable, by condition *(vip.2.a)*, we have $\{y, s\} = \{l'|q_f, r'|q_f\}$. Let $x = y\sigma$ and $t = s\sigma$, we have $\text{Var}(D) \subseteq \{x, t\}$. It is then easy to check that conditions *(vip.2.a)* and *(vip.2.b)* hold on D.

- If s is a constant, then by condition *(vip.2.b)*, $\text{Var}(C') = \{y\}$, and since D is not ground, we must have $y = r'|q_f$. Let $c = l'|q_f$, then $(l|q_f)\sigma = c$ and therefore, $(r|q_f)\sigma = c$. Hence condition *(vip.2.b)* holds again. ∎

Example 7.27 Consider the two following clauses:

$$\begin{aligned} C &= z \simeq i \vee \mathrm{select}(a, z) \simeq \mathrm{select}(b, z), \\ C' &= z \simeq w \vee \mathrm{select}(a, z) \not\simeq \mathrm{select}(b, w). \end{aligned}$$

Let D be the clause generated by the paramodulation of C into C', we have

$$D = z \simeq i \vee z \simeq w \vee \mathrm{select}(b, z) \not\simeq \mathrm{select}(b, w).$$

Let S' be any set of clauses such that $\{C, C'\}$ is Γ-immune from S' for some $\Gamma \in \Omega_\Sigma$. Then D is also Γ-immune from S'.

Theorem 7.28 *Let $T = T_\mathrm{g} \uplus T_1 \uplus T_2$ be a set of clauses such that $\langle T_\mathrm{g}, T_1, T_2 \rangle$ is subterm-inactive, and let C be a clause in T_1 with maximal literal L.*

- *If D is a persistent clause generated by a unary inference on C, then D is ground and flat-disconnected from $T_1 \cup T_2$. Furthermore, we have* $\mathrm{Maxd}(D) \leq \mathrm{Maxd}(C)$.
- *Let C' be a clause in $T_\mathrm{g} \cup T_1 \cup T_2$, let D be the clause obtained after the paramodulation of C into C', and suppose that D is persistent. Then either D is ground and flat-disconnected from $S \cup S'$, or D is not ground and $S \cup \{D\}$ is Γ-immune from S'. Furthermore, we have* $\mathrm{Maxd}(D) \leq \max\{\mathrm{Maxd}(C), \mathrm{Maxd}(C')\}$.

Proof. By Lemma 7.19, the only unary inference that can be applied to C is a reflection inference, and by Lemma 7.23, if the resulting clause is persistent, then it is ground and flat-disconnected from $T_1 \cup T_2$.

Since we do not paramodulate into variables, by Proposition 7.18, we do not need to consider the case where $C' \in T_2$. Suppose that $C' \in T_\mathrm{g}$, with maximal literal $L' = l' \bowtie r'$. Since l is of depth 1 by Proposition 7.4 and l' is ground, l' is at least of depth 1 and since C' is negatively disconnected from T_1, l' is of depth 1. Thus, σ is flat and by condition *(ec.3)*, D is ground. Suppose D contains a negative literal with a term of depth at least 2. Since C is weakly flat and σ is strictly flat, such a literal must also have occurred in C'. Since C' is negatively disconnected from T_1, l can contain no function symbol appearing in C', a contradiction since l and l' are unifiable. Suppose D contains a positive literal that is not strictly flat. Since C' is positively flat and by condition *(ec.2)*, this literal is necessarily $r\sigma \simeq r'$. Since L' is flat, r' is of depth 0, and since r is of depth 1, this literal is flat. Therefore, D is flat-disconnected from $T_1 \cup T_2$ and $\mathrm{Maxd}(D) \leq \max\{\mathrm{Maxd}(C), \mathrm{Maxd}(C')\}$.

Now suppose that $C' \in T_1$. If D is ground, then by Corollary 7.21, L' must be a negative literal, and C' cannot contain any positive literal that is not strictly flat. Also, since C is Γ-externally closed from T_2, all its literals other than L are strictly flat by condition *(ec.2)*. Thus, since σ is strictly flat, D only contains terms of depth at most 1, and is negatively disconnected from $T_1 \cup T_2$. Also, D contains no positive literal that is not strictly flat and is therefore positively flat, hence, $T_\mathrm{g} \cup \{D\}$ is flat-disconnected from $T_1 \cup T_2$.

If D is not ground, then $T_1 \cup \{D\}$ is weakly flat and Γ-externally closed from T_2 by Lemma 7.25, and it is Γ-variable-inactive preserving by Lemma 7.26. ∎

7.3 Expansions from saturation-closed clauses

Lemma 7.29 *Let C' be a ground clause, S be a set of clauses and $\Gamma \in \Omega_\Sigma$ such that S is Γ-internally closed and C' is flat-disconnected from S. Then there can be no inference from a clause $C \in S$ into C'.*

PROOF. Suppose we are paramodulating from the maximal literal $l \simeq r$ in C. By condition *(icp.1)* l must be of depth 2, and since C' is ground, l must be unifiable with a term l' of depth at least 2 in C'. Since C' is positively flat, it can only contain positive literals that are flat and l' must be in a negative literal in C'. Since C' is negatively disconnected from S, C is $\Phi(C')$-symbol-free and l and l' cannot be unifiable. ∎

Theorem 7.30 *Let $T = T_\mathrm{g} \uplus T_1 \uplus T_2$ be a set of clauses such that $\langle T_\mathrm{g}, T_1, T_2\rangle$ is subterm-inactive, and let C be a clause in T_2. Then any inference involving C is eventually deleted.*

PROOF. Since T_2 is saturated, every unary inference on C and every binary inference between C and a clause $C' \in T_2$ is eventually deleted. There can be no paramodulations from C into a clause in $T_\mathrm{g} \cup T_1$ by Proposition 7.18 and Lemma 7.29. ∎

7.4 Main theorem on subterm-inactive presentations

By Theorems 7.17, 7.28 and 7.30, we have the following result:

Corollary 7.31 *Let $\mathcal{T} = T_\mathrm{g} \uplus T_1 \uplus T_2$ be a set of clauses such that $\langle T_\mathrm{g}, T_1, T_2\rangle$ is a subterm-inactive tuple and let D be a persistent clause generated by an inference on $\mathcal{T}$. Then the inference is depth-preserving and:*

- *either D' is ground and $\langle T_\mathrm{g} \cup \{D'\}, T_1, T_2\rangle$ is subterm-inactive,*
- *or D' is not ground and $\langle T_\mathrm{g}, T_1 \cup \{D'\}, T_2\rangle$ is subterm-inactive.*

Definition 7.32 Given an integer d, we define the following sets of clauses:

$$\begin{aligned} \mathcal{C}_d(\Sigma, \mathcal{X}) &= \{C \mid \text{every term in } C \text{ is of depth at most } d\}, \\ \mathcal{C}_d(\Sigma) &= \{C \in \mathcal{C}_d(\Sigma, \mathcal{X}) \mid \text{every term in } C \text{ is ground}\}. \end{aligned}$$ ◇

Lemma 7.33 *Let $S_0 = T_\mathrm{g} \uplus T_1 \uplus T_2$ be a set of clauses such that $\langle T_\mathrm{g}, T_1, T_2\rangle$ is subterm-inactive for $\succ$, and let $d = \mathrm{Maxd}(S_0)$. If S_∞ is the set of persistent clauses generated by a fair $\mathcal{SP}_\succ$-strategy starting from S_0, then $S_\infty \subseteq \mathcal{C}_d(\Sigma) \cup \mathcal{C}_1(\Sigma, \mathcal{X}) \cup T_2$, and is therefore finite.*

Theorem 7.34 (Main theorem on subterm-inactive presentations) *If $\mathcal{T}$ is a subterm-inactive presentation, then any fair $\mathcal{SP}_\succ$-strategy is a decision procedure for $\mathcal{T}$.*

PROOF. Let $\mathcal{T}$ be a subterm-inactive presentation, by definition we have a partition $T_{\mathrm{g}} \uplus T_1 \uplus T_2$ of $\mathcal{T}$ such that $\langle T_{\mathrm{g}}, T_1, T_2 \rangle$ is subterm-inactive. Let S be a set of ground clauses and S' be the set obtained by flattening S. By Proposition 6.23, $\langle S' \cup T_{\mathrm{g}}, T_1, T_2 \rangle$ is subterm-inactive, and by Lemma 7.33 we have the result. ■

We also have the following property:

Lemma 7.35 *If $\langle T_{\mathrm{g}}, T_1, T_2 \rangle$ is subterm-inactive, then $T_{\mathrm{g}} \cup T_1 \cup T_2$ is variable-inactive.*

PROOF. Let $L = x \simeq t$ be a literal appearing in a clause in $T_{\mathrm{g}} \cup T_1 \cup T_2$, such that $x \notin \mathrm{Var}(t)$. Then L is either in a clause of T_1 or of T_2 since T_{g} only contains ground clauses. If L is in a clause in T_1, then this clause must be weakly flat by hypothesis, hence, t must be of depth 0. By Proposition 7.3, L cannot be maximal in this clause. If L is in a clause in T_2, then this clause must be ordered flat by hypothesis, and it therefore contains a literal $u \bowtie v$ that is not strictly flat and such that $u \succ v$. If $L = u \bowtie v$, then we would have $x \in \mathrm{Var}(u) = \mathrm{Var}(t)$ by conditions *(icn.1)* and *(icp.2)*, which is impossible. Thus L must be strictly flat, and by our goodness requirement on $\succ$, we also have $t \prec u$. Therefore, a literal $x \simeq t$ can never be maximal in a clause in $T_{\mathrm{g}} \cup T_1 \cup T_2$. ■

Corollary 7.36 *Let $\mathcal{T}$ be a subterm-inactive presentation, then $\mathcal{T}$ is also variable-inactive.*

PROOF. By Lemma 7.35, we know that if S_0 is subterm-inactive and $S_0 \vdash^*_{\mathcal{SP}_\succ} S$, then S is subterm-inactive. Therefore, S_∞ is subterm-inactive and we have the result. ■

8 Variations on the theory of arrays

In what follows, we consider the theory of arrays (see Example 6.3) and two of its extensions. It was shown in [ARR03] that a satisfiability problem in $\mathcal{A}^{\mathrm{e}}$ can be reduced to an equisatisfiable satisfiability problem in $\mathcal{A}$, and that the superposition calculus provides a satisfiability procedure for $\mathcal{A}$: a proof that the limit S_∞ is finite can be found in [ABRS05]. This kind of analysis requires long proofs: for $\mathcal{A}$, the clauses in S_∞ can belong to any one of 14 classes of clauses. We have the following result:

Lemma 8.1 *The presentation of $\mathcal{A}$ is subterm-inactive.*

PROOF. We prove that the tuple $\langle \emptyset, \emptyset, \{(1), (2)\} \rangle$ is subterm-inactive.

- The only inference that can be applied to $\{(1), (2)\}$ is a superposition between (1) and (2). This generates the clause $z \simeq z \vee \mathrm{select}(x, z) \simeq v$, which is deleted. Thus, this set is saturated.
- It is trivial to check that the clauses in $\{(1), (2)\}$ are ordered flat. Since they do not contain any constants, they are also subsymbol-free from Σ^0.

- The maximal literals in (1) and (2) are both positive and one can check that $\{(1), (2)\}$ is Γ-internally closed, for any selection function Γ such that $\Gamma(\text{select}) = 2$. ∎

Thus, as a corollary we deduce that

Corollary 8.2 *Any fair $\mathcal{SP}_{\succ}$-strategy is a decision procedure for the theory of arrays with or without extensionality.*

8.1 An injectivity predicate

Next, we consider the theory $\mathcal{A}_I$ of arrays augmented with an injectivity predicate, as defined in Example 6.11:

$$\text{Inj}(x) \quad \leftrightarrow \quad \forall z, w.\ (z \not\simeq w \supset \text{select}(x, z) \not\simeq \text{select}(x, w)).$$

We assume that each occurrence of the injectivity predicate has a constant as an argument. There is no loss of generality under this assumption. For example, the clause $\text{Inj}(f(a)) \vee B$ can be safely replaced by the clause $\text{Inj}(b) \vee B$ and the flat literal $f(a) \simeq b$, where b is a fresh constant. Still without loss of generality, we may suppose that if the injectivity predicate appears in a non-unit clause, then this clause is of the form $\text{Inj}(a) \vee \neg p$ or $\neg\text{Inj}(a) \vee \neg p$, where p is a propositional variable. Indeed, such a formula can be obtained from S by repeatedly replacing clauses of the form $\text{Inj}(a) \vee D$ (resp. $\neg\text{Inj}(a) \vee D$), where D is not a propositional variable, by the clauses $\text{Inj}(a) \vee \neg p_a$ and $p_a \vee D$ (resp. $\neg\text{Inj}(a) \vee \neg p_a$ and $p_a \vee D$), where p_a is a fresh propositional variable. The formula thus obtained is equisatisfiable to S (see [RV01] for details).

We remove all occurrences of the predicate Inj in the following way. For every constant a, we consider the clause C_a and its negated form C'_a, respectively defined by:

$$\begin{array}{lcl} C_a & = & \forall z, w.\ z \simeq w \vee \text{select}(a, z) \not\simeq \text{select}(a, w), \\ C'_a & = & (sk_1 \not\simeq sk_2 \wedge \text{select}(a, sk_1) \simeq \text{select}(a, sk_2)), \end{array}$$

where sk_1 and sk_2 are fresh (Skolem) constants. Note that, by definition, $\text{Inj}(a)$ is logically equivalent to $\forall z, w.\ (z \not\simeq w \supset \text{select}(a, z) \not\simeq \text{select}(a, w))$, and C_a is the clausal form of the latter formula. Thus, C_a and $\text{Inj}(a)$ are logically equivalent; similarly, C'_a and $\neg\text{Inj}(a)$ are also logically equivalent.

We can therefore safely replace every clause of the form $\text{Inj}(a) \vee \neg p$ by $C_a \vee \neg p$, and every clause of the form $\neg\text{Inj}(a) \vee \neg p$ by the clausal form of $C'_a \vee \neg p$.

Example 8.3 Let $S = \{\neg\text{Inj}(a) \vee \neg p_a,\ \text{Inj}(b) \vee \neg p_b\}$, then after the transformation we obtain

$$\begin{array}{lll} S' = & \{ & sk_1 \not\simeq sk_2 \vee \neg p_a, \\ & & \text{select}(a, sk_1) \simeq \text{select}(a, sk_2) \vee \neg p_a, \\ & & z \simeq w \vee \text{select}(b, z) \not\simeq \text{select}(b, w) \vee \neg p_b\}, \end{array}$$

where z and w are implicitly universally quantified variables.

Given a set of clauses S, the reduced set of clauses thus obtained is equisatisfiable to S, and we have the following:

Lemma 8.4 *Let $\{a_1, \ldots, a_n\}$ be a set of constants, and for every element $i \in \{1, \ldots, n\}$, let p_i be a propositional variable, and define*

$$C_i \quad = \quad \forall z, w.\ z \simeq w \vee \mathrm{select}(a_i, z) \not\simeq \mathrm{select}(a_i, w).$$

The theory $\mathcal{A} \cup \{C_i \vee p_i \mid i = 1, \ldots, n\}$ is subterm-inactive.

Proof. We show that the tuple $\langle \emptyset, \{C_1 \vee p_1, \ldots, C_n \vee p_n\}, \{(1), (2)\}\rangle$ is subterm-inactive. In Lemma 8.1, we showed that $\{(1), (2)\}$ is Γ-saturation-closed with $\Gamma(\mathrm{select}) = 2$. Consider any function $\Gamma' \in \Omega_\Sigma$ such that $\Gamma'(\mathrm{select}) = 1$, and a clause $C_i \vee p_i$. This clause is Γ-immune from $\{(1), (2)\}$: we have shown that C_i is Γ-variable-inactive preserving in Example 6.11, and it is simple to verify that $C_i \vee p_i$ is also Γ-externally closed from $\{(1), (2)\}$; the other conditions are trivial to verify. It is also Γ'-interaction-free from $\{(1), (2)\}$, hence the result.

Since these conditions are satisfied for every clause of the form $C_i \vee p_i$, it is clear that $\langle \emptyset, \{C_1 \vee p_1, \ldots, C_n \vee p_n\}, \{(1), (2)\}\rangle$ is subterm-inactive and the proof is complete. ∎

Thus, an $\mathcal{SP}_\succ$-strategy together with a fair search plan can be used to test the satisfiability of this set. We therefore have the following result:

Corollary 8.5 *A fair $\mathcal{SP}_\succ$-strategy is a decision procedure for $\mathcal{A}_I$.*

8.2 A Swap predicate

Given the theory $\mathcal{A}_S$ of arrays augmented with a swap predicate, as defined in Example 6.13, consider a ground $\mathcal{A}_S$-formula S. Up to adding flat equalities to S, unless stated otherwise, we assume that each occurrence of the swap predicate only has constants as arguments. As in the case of the injectivity predicate, we assume that the clauses in which the swap predicate appears are of the form $\mathrm{Swap}(b, b', i, i') \vee \neg p$ or $\neg\mathrm{Swap}(b, b', i, i') \vee \neg p$.

We remove all occurrences of the predicate Swap in the following way: for every tuple of constants $G = \langle b, b', i, i'\rangle$, where b and b' are of sort array and i and i' are of sort index, we consider the formula D_G and its negated form D'_G respectively defined by:

$$\begin{array}{rcl} D_G & = & \mathrm{select}(b, i) \simeq \mathrm{select}(b', i') \wedge \mathrm{select}(b, i') \simeq \mathrm{select}(b', i) \wedge \\ & & \qquad \forall w.\ (w \not\simeq i \wedge w \not\simeq i' \supset \mathrm{select}(b, w) \simeq \mathrm{select}(b', w)) \\ D'_G & = & \mathrm{select}(b, i) \not\simeq \mathrm{select}(b', i') \vee \mathrm{select}(b, i') \not\simeq \mathrm{select}(b', i) \vee \\ & & \qquad sk \not\simeq i \wedge sk \not\simeq i' \wedge \mathrm{select}(b, sk) \not\simeq \mathrm{select}(b', sk), \end{array}$$

where sk is a fresh (Skolem) constant. By definition, $\mathrm{Swap}(b, b', i, i')$ and D_G are logically equivalent, and one can check that $\neg\mathrm{Swap}(b, b', i, i')$ and D'_G are also logically equivalent.

Given a tuple $G = \langle b, b', i, i'\rangle$, we then replace every clause of the form $\mathrm{Swap}(b, b', i, i') \vee \neg p$ by the clausal form of $D_G \vee \neg p$, and every clause of the form $\neg\mathrm{Swap}(b, b', i, i') \vee \neg p$ by the clausal form of $D'_G \vee \neg p$. The set we obtain is equivalent to

S. In the following lemma, the clause Dk comes from the clausal form of D_G; the rest of the clausal form of D_G is ground, as is the clausal form of D'_G: they are not needed to prove subterm-inactivity.

Lemma 8.6 *Let $\{b_k, b'_k, i_k, i'_k \mid k = 1, \ldots, m\}$ be a set of constants and for every $k \in \{1, \ldots, m\}$, let p_k be a propositional variable, and consider the clause*

$$D_k \quad = \quad w \simeq i_k \vee w \simeq i'_k \vee \text{select}(b_k, w) \simeq \text{select}(b'_k, w).$$

The theory $\mathcal{A} \cup \{D_k \vee p_k \mid k = 1, \ldots, m\}$ is subterm-inactive.

Proof. We prove that $\langle \emptyset, \{D_1 \vee p_1, \ldots, D_m \vee p_m\}, \{(1), (2)\}\rangle$ is subterm-inactive. We showed that $\{(1), (2)\}$ is Γ-saturation-closed with $\Gamma(\text{select}) = 2$. Consider a clause D_k, it can be seen by applying Definition 6.14 that $D_k \vee p_k$ is Γ-immune from $\{(1), (2)\}$. As shown in Example 6.16, D_k is Γ'-interaction-free from $\{(1), (2)\}$ for any Γ' such that $\Gamma'(\text{select}) = 1$, and it is simple to show that $D_k \vee p_k$ is also Γ'-interaction-free from $\{(1), (2)\}$. Hence, for every k, $\langle \emptyset, \{D_k \vee p_k\}, \{(1), (2)\}\rangle$ is subterm-inactive. Since this is true for every k, we have the result. ∎

Corollary 8.7 *Any fair $\mathcal{SP}_\succ$-strategy is a decision procedure for $\mathcal{A}_S$.*

Finally, consider the theory $\mathcal{A}'$ axiomatized by (1), (2), (**inj**) and (**swp**), and let Γ and Γ' be selection functions such that $\Gamma(\text{select}) = 2$ and $\Gamma'(\text{select}) = 1$. Given the sets $A = \{C_i \vee p_i \mid i = 1, \ldots, n\}$ and $B = \{D_k \vee p'_k \mid k = 1, \ldots, m\}$, and for the selection functions Γ and Γ' defined above,

- $\langle \emptyset, A, \{(1), (2)\}\rangle$ is subterm-inactive by Lemma 8.4,
- $\langle \emptyset, B, \{(1), (2)\}\rangle$ is subterm-inactive by Lemma 8.6.

We deduce that the tuple $\langle \emptyset, A \cup B, \{(1), (2)\}\rangle$ is also subterm-inactive. We therefore have the following result:

Theorem 8.8 *A fair $\mathcal{SP}_\succ$-strategy is a decision procedure for $\mathcal{A}'$.*

8.3 A non-obvious example

The next example shows that although the conditions required for a tuple to be subterm-inactive are quite strong, some of them are tight, and allow us to point out some non-obvious results.

Example 8.9 Consider the following predicate:

$$\text{Const}_y(x) \quad \leftrightarrow \quad \forall z.\ \text{select}(x, z) \simeq y,$$

that expresses the property that an array represents a constant function. It is easy to check that given two constants a and e, $\mathcal{T} = \{(1), (2)\} \cup \{\text{Const}_e(a)\}$ is not subterm-inactive: there exists no Γ such that $\text{Const}_e(a)$ is Γ-immune from any other set (condition

(ec.1) does not hold), or Γ-saturation-closed (condition *(icp.1)* does not hold). Actually, $\mathcal{T}$ is not even variable-inactive; consider the following set:

$$\begin{array}{rcl} S & = & \{\mathrm{store}(a,i,e_1) \simeq a',\ \mathrm{Const}_e(a) \simeq \mathsf{true},\ \mathrm{Const}_{e'}(a') \simeq \mathsf{true}\}, \\ & \leftrightarrow & \{\mathrm{store}(a,i,e_1) \simeq a',\ \mathrm{select}(a,z) \simeq e,\ \mathrm{select}(a',z) \simeq e'\}. \end{array}$$

A superposition of $\mathrm{store}(a,i,e_1) \simeq a'$ into axiom (2) yields the clause

$$w \simeq i \ \vee\ \mathrm{select}(a,w) \simeq \mathrm{select}(a',w) \tag{4}$$

Simplifications of this clause by $\mathrm{select}(a',z) \simeq e'$ and $\mathrm{select}(a,z) \simeq e$ yields $w \simeq i \vee e \simeq e'$. This clause is not variable-inactive, and since it cannot be deleted, S_∞ is not variable-inactive either.

9 A collection of decision procedures

The approach based on subterm-inactivity allows us to re-obtain other termination results for $\mathcal{SP}$ on $\mathcal{T}$-satisfiability problems and generalize them to $\mathcal{T}$-decision problems.

Finite sets with or without extensionality

The theory of finite sets is based on the signature $\Sigma_{set} = \{\mathrm{mem}, \mathrm{ins}\}$, where mem and ins both have arity 2. Intuitively, $\mathrm{mem}(e,s)$ is true if e is an element of the s, and $\mathrm{ins}(e,s)$ inserts element e into the set s. The theory is defined by the following presentation, denoted by $\mathcal{FS}$:

$$\forall x, v. \qquad \mathrm{mem}(v, \mathrm{ins}(v,x)) \simeq \mathsf{true}, \tag{5}$$

$$\forall x, v, w. \qquad v \not\simeq w \Rightarrow \mathrm{mem}(v, \mathrm{ins}(w,x)) \simeq \mathrm{mem}(v,x). \tag{6}$$

The theory of finite sets with extensionality is presented by $\mathcal{FS}^{\mathrm{e}}$, which consists of axioms (5) and (6) along with the following extensionality axiom:

$$\forall x, y. \qquad (\forall v.(\mathrm{mem}(v,x) \simeq \mathrm{mem}(v,y))) \Rightarrow x \simeq y. \tag{7}$$

It was proved in [ARR03, Theorem 8.1] that any $\mathcal{FS}^{\mathrm{e}}$-decision problem can be reduced to an $\mathcal{FS}$-decision problem. We have the following result:

Lemma 9.1 $\langle \emptyset, \emptyset, \{(5),(6)\}\rangle$ *is subterm-inactive.*

Proof. All one has to do is to verify that $\{(5),(6)\}$ is saturation-closed. This is the case, since the superposition of (5) into (6) generates $v \simeq v \vee \mathrm{mem}(v,x) \simeq \mathsf{true}$, and this clause can be deleted by the Deletion inference rule. Thus, $\{(5),(6)\}$ is saturated, and it is simple to check that it is subsymbol-free from Σ^0 and that both clauses are ordered-flat. Let Γ be any function in Ω_Σ such that $\Gamma(\mathrm{mem}) = 1$. Conditions *(icp.1)*, *(icp.2)* and *(icp.3)* hold on axiom (5), and conditions *(icp.1)*, *(icp.2)* and *(icp.4)* hold on axiom (6), so that $\{(5),(6)\}$ is Γ-internally closed. ∎

Recursive data structures

The class of recursive data structures, which was defined above, enjoys the following property:

Lemma 9.2 *$\langle \emptyset, \{(\mathbf{ext})\}, Ac(n) \rangle$ is subterm-inactive.*

PROOF. Let Γ be a function in Ω_Σ such that for every $f \in \Sigma_{sel}$, $\Gamma(f) = 1$. We show that $\{(\mathbf{ext})\}$ is Γ-immune from $Ac(n)$. This set is trivially Γ-externally closed from $Ac(n)$ since every positive literal in $(\mathbf{ext})$ is strictly flat. Also, $(\mathbf{ext})$ is weakly flat and conditions *(vip.1)* and *(vip.2.a)* hold, so that $\{(\mathbf{ext})\}$ is Γ-variable-inactive preserving. Since $\{(\mathbf{ext})\} \cup Ac(n)$ only contains positive literals that are strictly flat, $\{(\mathbf{ext})\}$ is trivially Γ-interaction free from $Ac(n)$.

We now show that $Ac(n)$ is saturation-closed. It is simple to check that this set is saturated and subsymbol-free from Σ^0. Since it only contains unit clauses, all these clauses are trivially ordered-flat. The clauses in $Ac(n)$ are all of the form $t[x] \not\simeq x$, so that condition *(icn.1)* holds. Since these clauses are all negative, condition *(icn.2)* trivially holds. ■

Summary

Theorem 9.3 *The superposition calculus yields $\mathcal{T}$-decision procedures for the following theories:*

- *Equality with Uninterpreted Functions (EUF).*
- *Arrays with or without extensionality, possibly augmented with an injectivity predicate, a swap predicate or both.*
- *Finite sets with or without extensionality.*
- *Recursive data structures.*

PROOF. The result is obvious for the theory of equality with uninterpreted functions, which is presented by the empty set, and was shown for the variations of the theory of arrays in the previous section. Lemmas 9.1 and 9.2 prove the result for finite sets with or without extensionality and recursive data structures, respectively. ■

10 Discussion

In this paper, we first considered a class of theories representing recursive data structures, each of which is defined by an infinite set of axioms. We showed that the superposition calculus can be used as the basis of a satisfiability procedure for any theory in this class, and this result was obtained by defining a reduction that permits to restrict the number of acyclicity axioms to be taken into account.

A main issue we plan to investigate is complexity, since the basic procedure is exponential. A linear algorithm for such structures was obtained in [Opp80], but it excludes uninterpreted function symbols. The setting of [NO03] includes uninterpreted function symbols, but the authors gave a polynomial algorithm only for the case where $k = 1$ (the theory of integer offsets). We intend to investigate making the complexity of the rewrite-based procedure dependent on k, and improving the bound for $k = 1$.

From the point of view of practical efficiency, we plan to test the performance of a state-of-the-art theorem prover on problems featuring this theory, possibly combined with those of [ARR03, ABRS05], and compare it with systems implementing decision procedures from other approaches. In this context, we may work on designing specialized search plans for satisfiability problems. Another direction for future work is to examine how the rewrite-based approach applies to recursive data structures with an atom predicate, or multiple constructors.

We also introduced the notion of subterm-inactive theory, that guarantees that $\mathcal{SP}$ yields $\mathcal{T}$-decision procedures. Almost all the conditions for subterm-inactivity are static, which means they can be tested automatically and only once. We showed that several theories, including most of those considered in [ARR03, ABRS05], and two extensions of the theory of arrays, satisfy these conditions, which indicates that they are not too strong. Still, some of the theories of [ABRS05] are not subterm-inactive. They are the theory of possibly empty lists, the theory of records and the theory of integer offsets modulo. We intend to investigate how to weaken the subterm-inactivity conditions to obtain a larger class of subterm-inactive theories.

An important issue is how to combine subterm-inactive theories with Presburger arithmetic, especially for the theory of arrays. Indeed, using Presburger arithmetic on indices allows one to work on more complex properties about arrays, such as testing whether subarrays are identical.

Acknowledgments

The authors wish to thank Alessandro Armando and Silvio Ranise for bringing recursive data structures and injective arrays to their attention.

Bibliography

[ABRS05] Alessandro Armando, Maria Paola Bonacina, Silvio Ranise, and Stephan Schulz. On a rewriting approach to satisfiability procedures: Extension, combination of theories and an experimental appraisal. In Bernhard Gramlich, editor, *Frontiers of Combining Systems, 5th International Workshop, FroCoS 2005, Vienna, Austria, September 19-21, 2005, Proceedings*, volume 3717 of *Lecture Notes in Computer Science*, pages 65–80. Springer, 2005. Full version available at `http://www.sci.univr.it/~bonacina/papers/TR2005rewsat.pdf`.

[ABRS06] Alessandro Armando, Maria Paola Bonacina, Silvio Ranise, and Stephan Schulz. On a rewriting approach to satisfiability procedures: Theories of data structures, modularity and experimental appraisal. Technical Report RR 36/2005, Dipartimento di Informatica, Università degli Studi di Verona, 2006.

[ARR03] A. Armando, S. Ranise, and M. Rusinowitch. A Rewriting Approach to Satisfiability Procedures. *Info. and Comp.*, 183(2):140–164, June 2003.

[BB04] Clark W. Barrett and Sergey Berezin. CVC lite: A new implementation of the Cooperating Validity Checker. In Rajeev Alur and Doron Peled, editors, *Computer Aided Verification, 16th International Conference, Boston, MA, USA, July 13-17, 2004, Proceedings*, volume 3114 of *Lecture Notes in Computer Science*, pages 515–518. Springer, 2004.

[BDS00] Clark W. Barrett, David L. Dill, and Aaron Stump. A framework for cooperating decision procedures. In David A. McAllester, editor, *Automated Deduction - CADE-17, 17th International Conference on Automated Deduction, Pittsburgh, PA, USA, June 17-20, 2000, Proceedings*, volume 1831 of *Lecture Notes in Computer Science*, pages 79–98. Springer, 2000.

[BDS02] Clark W. Barrett, David L. Dill, and Aaron Stump. Checking satisfiability of first-order formulas by incremental translation to SAT. In Ed Brinksma and Kim Guldstrand Larsen, editors, *Computer Aided Verification, 14th International Conference, CAV 2002,Copenhagen, Denmark, July 27-31, 2002, Proceedings*, volume 2404 of *Lecture Notes in Computer Science*, pages 236–249. Springer, 2002.

[BE06] Maria Paola Bonacina and Mnacho Echenim. Generic theorem proving for decision procedures. Technical report, Università degli studi di Verona, 2006. Available at `http:profs.sci.univr.it/~echenim/`.

[dMRS02] Leonardo Mendonça de Moura, Harald Rueß, and Maria Sorea. Lazy theorem proving for bounded model checking over infinite domains. In Andrei Voronkov, editor, *Automated Deduction - CADE-18, 18th International Conference on Automated Deduction, Copenhagen, Denmark, July 27-30, 2002, Proceedings*, volume 2392 of *Lecture Notes in Computer Science*, pages 438–455. Springer, 2002.

[DP01] Nachum Dershowitz and David A. Plaisted. Rewriting. In J.A. Robinson and A. Voronkov, editors, *Handbook of Automated Reasoning*, volume I, pages 535–610. Elsevier Science Publishers, 2001.

[NO03] Robert Nieuwenhuis and Albert Oliveras. Congruence closure with integer offsets. In Moshe Y. Vardi and Andrei Voronkov, editors, *Logic for Programming, Artificial Intelligence, and Reasoning, 10th International Conference, LPAR 2003, Almaty, Kazakhstan, September 22-26, 2003, Proceedings*, volume 2850 of *Lecture Notes in Computer Science*, pages 78–90. Springer, 2003.

[NR01] Robert Nieuwenhuis and Albert Rubio. Paramodulation-based theorem proving. In John Alan Robinson and Andrei Voronkov, editors, *Handbook of Automated Reasoning*, pages 371–443. Elsevier and MIT Press, 2001.

[Opp80] D. Oppen. Reasoning about recursively defined data structures. *J. of the ACM*, 27(3), July 1980.

[RV01] Alexandre Riazanov and Andrei Voronkov. Splitting without backtracking. In Bernhard Nebel, editor, *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence, IJCAI 2001, Seattle, Washington, USA, August 4-10, 2001*, pages 611–617. Morgan Kaufmann, 2001.

[SBD04] Aaron Stump, Clark W. Barrett, and David L. Dill. CVC: A Cooperating Validity Checker. In Rajeev Alur and Doron Peled, editors, *Computer Aided Verification, 16th International Conference, CAV 2004, Boston, MA, USA, July 13-17, 2004, Proceedings*, volume 3114 of *Lecture Notes in Computer Science*, pages 500–504. Springer, 2004.

[ZSM04] Ting Zhang, Henny B. Sipma, and Zohar Manna. Decision procedures for recursive data structures with integer constraints. In David A. Basin and Michaël Rusinowitch, editors, *Automated Reasoning - Second International Joint Conference, IJCAR 2004, Cork, Ireland, July 4-8, 2004, Proceedings*, volume 3097 of *Lecture Notes in Computer Science*, pages 152–167. Springer, 2004.