Available online at www.sciencedirect.com

ScienceDirect

Electronic Notes in Theoretical Computer Science 174 (2007) 55–70

Electronic Notes in Theoretical Computer Science

www.elsevier.com/locate/entcs

Rewrite-Based Satisfiability Procedures for Recursive Data Structures

Maria Paola Bonacina[1] Mnacho Echenim[2]

Dipartimento di Informatica, Università degli Studi di Verona, Italy

Abstract

If a rewrite-based inference system is guaranteed to terminate on the axioms of a theory $\mathcal{T}$ and any set of ground literals, then any theorem-proving strategy based on that inference system is a rewrite-based decision procedure for $\mathcal{T}$-satisfiability. In this paper, we consider the class of theories defining *recursive data structures*, that might appear out of reach for this approach, because they are defined by an infinite set of axioms. We overcome this obstacle by designing a problem reduction that allows us to prove a general termination result for all these theories. We also show that the theorem-proving strategy decides satisfiability problems in any combination of these theories with other theories decided by the rewrite-based approach.

Keywords: Rewrite-based inference systems, recursive data structures

1 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 [3], 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 $\mathcal{T}$ and any set of ground literals, then it is a decision procedure for $\mathcal{T}$*-satisfiability*. In [3], this idea was applied to a standard inference system, the superposition calculus $\mathcal{SP}$, and several theories, including those of *arrays* and *possibly cyclic non-empty lists*.

[1] Email: `mariapaola.bonacina@univr.it`

[2] Email: `echenim@sci.univr.it`

doi:10.1016/j.entcs.2006.11.039

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. The notion of variable-inactive theories appear in [1], along with several experimental results that showed the practicality of the rewrite-based approach. This variable-inactivity condition guarantees that $\mathcal{SP}$ terminates on a combination of theories, provided it terminates on each individual theory. Thus, it was shown in [1] that an $\mathcal{SP}$-based strategy is a satisfiability procedure for any combination of the theories considered in [3] and [1].

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 [3]), and for the theory of lists with nil (see [2]). A theory of lists that was not yet examined is that of *acyclic lists*, where formulae such as $\mathrm{car}(x) \simeq x$ are unsatisfiable. This theory, along with that of *integer offsets* studied in [8,1], belongs 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. The contributions of the paper are the following:

- 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 [3,1,2], 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.

Related work.

Theories of recursive data structures were studied by Oppen in [10], where he described a linear satisfiability procedure for the case where uninterpreted function symbols are excluded. The authors of [11] 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. However, this type completion can be expensive in practice as the number of guesses explodes with the number of constructors. More recently, the authors of [6] devised an algorithm for a more general class of recursive data structures: the algorithm is based on a multi-sorted logical framework and solves $\mathcal{T}$-decision problems for several recursive data structures that may be mutually recursive, each of which may include several constructors. This algorithm uses a combination of the techniques of [10,11]: as in [10], it performs congruence and unification closure, and it handles multiple constructors with a type completion as in [11]. The algorithm is presented as an elegant combination of standard inference rules for congruence closure (see, e.g., [5]) and unification closure (see, e.g, [7]), with more specialized inference rules for type completion or the detection of cycles. In particular, the inference rules for type completion make this procedure more flexible and less expensive than that of [11].

In this paper, we consider the recursive data structures as defined in [10], since our aim is 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.

Preliminaries

In the following, given a signature Σ, we consider the standard definitions of Σ-terms, Σ-literals and Σ-theories. The symbol $\simeq$ denotes unordered equality and $\bowtie$ is either $\simeq$ or $\not\simeq$. Unless stated otherwise, the letters x and y will denote variables, d and e elements of an interpretation domain, and all other lower-case letters will be constants or function symbols in Σ. Given a term t, $\mathrm{Var}(t)$ denotes the set of variables appearing in t. If t is a constant or a variable, then the *depth* of t is $\mathrm{depth}(t) = 0$, and otherwise, $\mathrm{depth}(f(t_1,\ldots,t_n)) = 1+\max\{\mathrm{depth}(t_i)\,|i = 1,\ldots,n\}$. The *depth* of a literal is defined by $\mathrm{depth}(l \bowtie r) = \mathrm{depth}(l) + \mathrm{depth}(r)$. A positive literal is *flat* if its depth is 0 or 1, and a negative literal is *flat* if its depth is 0. We will make use of the following standard result: given a signature Σ and a Σ-theory $\mathcal{T}$, let S be a finite set of Σ-literals. Then there exists a signature Σ' obtained from Σ by adding a finite number of constants, and a finite set S' of flat Σ'-literals such that S' is $\mathcal{T}$-satisfiable if and only if S is.

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, 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 and only if $s \succ t$. More details on orderings can be found, e.g., in [4]. A CSO is extended to literals and clauses by multiset extension as usual, and when no confusion is possible we will mention maximal literals without any reference to $\succ$.

The *superposition calculus*, or $\mathcal{SP}$, is a *rewrite-based inference system* which is refutationally complete for first-order logic with equality (see, e.g., [9]). It consists

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

Fig. 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.

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 a standard way. Given a CSO $\succ$, we write $\mathcal{SP}_{\succ}$ for $\mathcal{SP}$ with $\succ$. An $\mathcal{SP}_{\succ}$*-derivation* is a sequence

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

each S_i being a set of clauses obtained by applying an expansion or a contraction rule to clauses in S_{i-1}. Such a derivation yields a set of *persistent clauses*:

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

which can of course be infinite. Given a finite set of ground literals S, in order to prove that the set of persistent clauses obtained by a fair $\mathcal{SP}_{\succ}$-derivation from $\mathcal{T} \cup S$ is finite, we may impose additional restrictions on the CSO $\succ$. Any CSO verifying these restrictions will be termed as $\mathcal{T}$*-good*. We also say that an $\mathcal{SP}_{\succ}$-strategy is $\mathcal{T}$*-good* if the CSO $\succ$ is $\mathcal{T}$-good.

A clause C is *variable-inactive for* $\succ$ 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

Strict Subsumption	$$\begin{array}{c} C \quad D \\ \hline\hline C \end{array}$$	$D \gtrdot C$
Simplification	$$\begin{array}{c} C[u] \quad l \simeq r \\ \hline\hline C[r\sigma] \quad l \simeq r \end{array}$$	$u = l\sigma,\ l\sigma \succ r\sigma,\ C[u] \succ (l \simeq r)\sigma$
Deletion	$$\begin{array}{c} C \vee t \simeq t \\ \hline\hline \end{array}$$	

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

Fig. 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.

are variable-inactive for $\succ$. A theory 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 $\succ$.

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

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

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:

$$Ac[2] = \{\mathrm{car}(\mathrm{car}(x)) \not\simeq x,\ \mathrm{cdr}(\mathrm{cdr}(x)) \not\simeq x,\ \mathrm{car}(\mathrm{cdr}(x)) \not\simeq x,\ \mathrm{cdr}(\mathrm{car}(x)) \not\simeq x\}.$$

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:

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

According to the methodology of [3,1,2], 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 $Red_{\mathcal{RDS}_k}(S)$ 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":

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

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:

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

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 (**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:

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

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:

$$Ac[1] \cup \{(\mathbf{ext})\} \cup S \text{ and } Ac[2] \cup \{(\mathbf{ext})\} \cup S \text{ are satisfiable,}$$
$$Ac[3] \cup \{(\mathbf{ext})\} \cup S \text{ and } \quad Ac \cup \{(\mathbf{ext})\} \cup S \quad \text{are unsatisfiable.}$$

We will 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 $i = 1, \ldots, m$,

(i) $f_i \in \Sigma_{sel}$,

(ii) for all $j \in \{i+1, \ldots, m\}$, $d_j \neq d_i$,

(iii) if $i \leq m-1$, then $d_{i+1} = f_i^I(d_i)$.

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

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 $k = 2$, let $S = \{\mathsf{sel}_1(c) \simeq c_1, \mathsf{sel}_2(c) \simeq c_2\}$, $D = \{1, 2, 3\}$, and define:

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

Then $M = (D, I)$ is a model for S, $\langle 1, \mathsf{sel}_1, 2, \mathsf{sel}_1 \rangle$ is an acyclic 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.*

In Lemma 4.8, we will show how to construct a model M' for $Ac[n+1] \cup \{(\mathbf{ext})\} \cup S$, given a model M for $Ac[n] \cup \{(\mathbf{ext})\} \cup S$. The construction involves breaking cyclic M-paths while preserving satisfiability, and this preservation will be guaranteed for M-paths containing *selector-free* elements.

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.

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

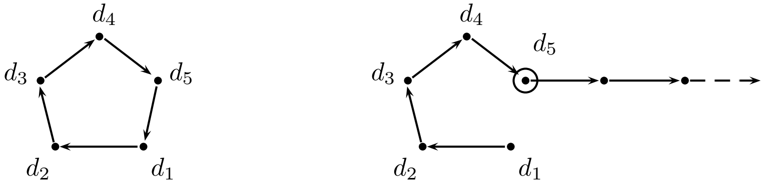

Fig. 3. Breaking cyclic M-paths: d_5 is selector-free

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 $M' = (D', J)$, and of course, $D \subseteq D'$.

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 $d = d'$.

$M' \models S$**:** 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 $M' \models S$.

□

A simple induction using Lemma 4.8 shows that if S is an $\mathcal{RDS}_k$-reduced set containing n selectors and $Ac[n] \cup S$ is satisfiable, then for every $k \geq 0$, $Ac[n+k] \cup S$

is also satisfiable. We therefore deduce:

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.*

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

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 an $\mathcal{RDS}_k$-good CSO:

Definition 5.1 A CSO $\succ$ is $\mathcal{RDS}_k$-good if $t \succ c$ for every ground compound term t and every constant c.

Lemma 5.2 *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 $i_1, \dots, i_k$ *is a permutation of* $1, \dots, k$, $0 \leq j_1 \leq j_2 \leq j_3 \leq k$, $j_3 > 0$ *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 *(iii)*: any paramodulation from a clause in *(iii)* generates a clause in categories *(iii)*, *(iv)* or *(v)*.

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.

□

Example 5.3 Consider the case where $k = 3$, and suppose we want to test the unsatisfiability of the following set:

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

- 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
$$c \simeq c' \vee \left(\bigvee_{i=1}^{3} d_i \not\simeq d'_i\right).$$
- 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 5.2. We therefore deduce:

Corollary 5.4 *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 5.2.

Theorem 5.5 *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 5.2.

- 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 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 $(\mathbf{ext})$. In fact, it is shown in [2] that any 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 5.6 *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 5.2. 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 [3,1], including those of arrays and records with or without extensionality.

6 Conclusion

In this paper, we 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 [10], but it excludes uninterpreted function symbols. The setting of [8] 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 [3,1], 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.

The work of [6] bears some similarity to ours since it is also based on a set of inference rules, and basic equational reasoning (i.e., congruence closure) is done by rewriting, as in $\mathcal{SP}$. It would hence be interesting to investigate how their techniques can be applied to our framework, and in particular whether $\mathcal{SP}$ can be enriched with specialized inference rules to handle acyclicity instead of including the acyclicity axioms in the satisfiability problems. Another direction for future work is to examine how the rewrite-based approach applies to recursive data structures with multiple constructors.

Acknowledgement

The authors wish to thank Silvio Ranise for bringing this class of theories to their attention.

References

[1] 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, *Proc. 5th FroCoS*, volume 3717 of *LNAI*, pages 65–80. Springer, 2005. Full version available as [2].

[2] 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.

[3] Alessandro Armando, Silvio Ranise, and Michaël Rusinowitch. A rewriting approach to satisfiability procedures. *Inf. Comput.*, 183(2):140–164, 2003.

[4] Franz Baader and Tobias Nipkow. *Term Rewriting and All That.* Cambridge University Press, 1998.

[5] Leo Bachmair, Ashish Tiwari, and Laurent Vigneron. Abstract congruence closure. *J. Autom. Reasoning*, 31(2):129–168, 2003.

[6] C. Barrett, I. Shikanian, and C. Tinelli. An abstract decision procedure for satisfiability in the theory of recursive data types. In *Proceedings of PDPAR'06*, 2006. Full version available at http://www.cs.nyu.edu/~barrett/pub.html#tech.

[7] Jean-Pierre Jouannaud and Claude Kirchner. Solving equations in abstract algebras: A rule-based survey of unification. In *Computational Logic - Essays in Honor of Alan Robinson*, pages 257–321, 1991.

[8] 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.

[9] 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.

[10] Derek C. Oppen. Reasoning about recursively defined data structures. *J. ACM*, 27(3):403–411, 1980.

[11] 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.