On variable-inactivity and polynomial $\mathcal{T}$-satisfiability procedures

Maria Paola Bonacina[*] **Mnacho Echenim**[†]

Dipartimento di Informatica

Università degli Studi di Verona

Strada Le Grazie 15, I-39134 Verona, Italy

Abstract

Verification problems require to reason in theories of data structures and fragments of arithmetic. Thus, decision procedures for such theories are needed, to be embedded in, or interfaced with, proof assistants or software model checkers. Such decision procedures ought to be sound and complete, to avoid false negatives and false positives, efficient, to handle large problems, and easy to combine, because most problems involve multiple theories. The *rewrite-based approach* to decision procedures aims at addressing these sometimes conflicting issues in a uniform way, by harnessing the power of general first-order theorem proving. In this article, we generalize the rewrite-based approach from deciding the satisfiability of sets of ground literals to deciding that of *arbitrary ground formulæ* in the theory. Next, we present *polynomial rewrite-based satisfiability procedures* for the theories of *records with extensionality* and *integer offsets*. The generalization of the rewrite-based approach to arbitrary ground formulæ and the polynomial satisfiability procedure for the theory of records with extensionality use the same key property – termed *variable-inactivity* – that allows one to combine theories in a simple way in the rewrite-based approach.

Keywords: rewrite-based theorem-proving, theory reasoning, satisfiability modulo theories, decision procedures, theories of data structures

[*] E-mail: mariapaola.bonacina@univr.it; Tel.: +39 045/802.7046

[†] E-mail: echenim@sci.univr.it; Tel.: +39 045/802.7908

1 Introduction

Decision procedures for satisfiability modulo a theory, or, equivalently, $\mathcal{T}$-satisfiability, where $\mathcal{T}$ is a theory, are essential components of reasoning-based verification systems from various approaches (e.g., PVS [31], ACL2 [21], CVC Lite [4], Zap [23], Simplify [16], MathSAT [11], Yices [17] and Barcelogic [28]). With this motivation, much research is being devoted to the concept and design of *$\mathcal{T}$-satisfiability procedures*, that are efficient and simple to combine. Indeed, most verification problems of practical interest require the system to reason in several theories at the same time. The problem of *combination of theories* is how to solve $\mathcal{T}$-satisfiability problems, where $\mathcal{T}$ is the union of theories $\mathcal{T}_1, \ldots, \mathcal{T}_n$, by using theory solvers for each $\mathcal{T}_i$ in a modular way. A standard solution is to resort to the *Nelson-Oppen combination scheme*, that was pioneered in [27] and developed by several authors (e.g., [20] for a survey). This scheme requires that all involved theories are *stably infinite*, which means a ground formula is $\mathcal{T}$-satisfiable if and only if it is $\mathcal{T}$-satisfiable in a model with domain of infinite cardinality.

The *rewrite-based approach* [3, 1, 8] investigates ways to apply generic first-order theorem proving to $\mathcal{T}$-satisfiability problems. The central idea is that if a refutationally complete inference system $\mathcal{SP}$ is guaranteed to *terminate* on $\mathcal{T}$-satisfiability problems, a fair $\mathcal{SP}$-strategy is a decision procedure for $\mathcal{T}$-satisfiability, where $\mathcal{SP}$ (e.g., [29]) is a standard, rewrite-based inference system, named from superposition. This methodology was applied successfully to the $\mathcal{T}$-satisfiability of sets of ground literals in the theories of *equality*, *non-empty possibly cyclic lists* (so-called lists à la Shostak and lists à la Nelson-Oppen), *arrays with or without extensionality* and *finite sets with extensionality* [3]; *records with or without extensionality*, *possibly empty possibly cyclic lists*, *integer offsets* and *integer offsets modulo* [1]; and *recursive data structures* as defined in [30], with one constructor and multiple selectors, including *non-empty acyclic lists* [8].

The combination of theories is addressed as *modularity of termination*: if $\mathcal{SP}$ terminates on $\mathcal{T}_i$-satisfiability problems, it terminates also on $\mathcal{T}$-satisfiability problems, provided the combined theories are *variable-inactive* [1]. All the above mentioned theories from [3, 1, 8] satisfy this requirement. Variable-inactivity appears to be a rather versatile property. In [10, 1], it was shown that it implies stable infiniteness, thereby relating combination of theories in the rewrite-based approach to combination of theories à la Nelson-Oppen. In [22], variable-inactivity, and the *meta-saturation* notion of [24], later revised in [25], were used to devise simple tests for stable infiniteness and a so-called deduction completeness property.

In this article, we study variable-inactivity to pursue two objectives:

1. Generalizing the rewrite-based approach from $\mathcal{T}$-satisfiability of sets of ground literals to that of ground formulæ and

2. Lowering the complexity of rewrite-based $\mathcal{T}$-satisfiability procedures.

1.1 Rewrite-based $\mathcal{T}$-decision procedures

In the literature on decision procedures for theory reasoning, a procedure is said to be a *$\mathcal{T}$-satisfiability procedure*, if it decides $\mathcal{T}$-satisfiability of sets of ground literals, and a *$\mathcal{T}$-decision procedure*, if it decides $\mathcal{T}$-satisfiability of arbitrary ground formulæ. Of course, a $\mathcal{T}$-satisfiability procedure could be applied to a formula after it has been reduced to disjunctive normal form, but this approach is not practical. Another method would be to explore how to integrate rewrite-based $\mathcal{T}$-satisfiability procedures with a SAT-solver, that is, a solver for the satisfiability problem in propositional logic. This is the approach of many authors (e.g., [5, 13, 2, 18, 11, 28]) for $\mathcal{T}$-satisfiability procedure based on congruence closure algorithms, to reason about ground equalities, with the theory axioms built-in. A lose integration of the E theorem prover [32], used as a rewrite-based $\mathcal{T}$-satisfiability procedure, with a SAT-solver, was experienced with in the haRVey system [14]. A tight integration has never been realized. Since SAT-solvers are based on case analysis by backtracking, and rewrite-based inference engines are *proof-confluent*, which means they do not require backtracking, their tight integration would require to address the issues posed by the interplay of two radically different sorts of control.

A different direction was explored in [7], by introducing a set of conditions collectively termed *subterm-inactivity.* In that article we proved that subterm-inactivity guarantees the termination of $\mathcal{SP}$ on $\mathcal{T}$-decision problems. In the above catalogue of theories, subterm-inactivity is satisfied by decision problems in the theories of *equality*, *finite arrays with or without extensionality*, *recursive data structures*, including *integer offsets* as a special case, *sets with or without extensionality* and two extensions of the theory of arrays with additional predicates. Thus, an $\mathcal{SP}$-strategy is a $\mathcal{T}$-decision procedure for problems in these theories and in their combinations, because subterm-inactivity implies variable-inactivity [7]. However, other theories, such as those of *lists*, *records* and *integer offsets modulo* do not satisfy subterm-inactivity. Furthermore, although most subterm-inactivity conditions can be tested automatically in principle, they are interwoven in a complex way and therefore hard to understand.

In this article, we present a more general and much simpler approach. We prove that variable-inactivity alone is sufficient to guarantee the termination of $\mathcal{SP}$ on $\mathcal{T}$-decision problems. More precisely, if $\mathcal{T}$ is variable-inactive, a rewrite-based $\mathcal{T}$-satisfiability procedure also yields a rewrite-based $\mathcal{T}$-decision procedure. This finding draws on an analysis of $\mathcal{SP}$-inferences in variable-inactive theories. It follows that an $\mathcal{SP}$-strategy is a $\mathcal{T}$-decision procedure for any variable-inactive theory for which it is a $\mathcal{T}$-satisfiability procedure. Since subterm-inactivity implies variable-inactivity, this result improves that of [7] by weakening the sufficient condition for termination. As mentioned above, while only some of the theories covered by the rewrite-based approach [3, 1, 8] are subterm-inactive, *all* of them are variable-inactive. This means that an $\mathcal{SP}$-strategy is a $\mathcal{T}$-decision procedure for the theories of *equality*, *non-empty possibly cyclic lists*, *arrays with or without extensionality*, *finite sets with or without extensionality*, *records with or without extensionality*, *possibly empty possibly cyclic lists*, *integer offsets modulo*, and *recursive data structures*, including *integer offsets* and *non-empty acyclic lists* as subcases.

1.2 Polynomial rewrite-based $\mathcal{T}$-satisfiability procedures

Using first-order theorem-proving strategies as $\mathcal{T}$-satisfiability procedures offers several advantages. First, it is not necessary to provide *ad hoc* proofs of correctness and completeness for each procedure for each theory, since the inference system is known to be sound and refutationally complete for first-order logic with equality. Second, *proof generation* and *model generation* are considered as desirable, but not standard, features for $\mathcal{T}$-satisfiability procedures at least since [26]. On the other hand, proof generation is a standard feature of theorem provers, and if the input set is satisfiable, the strategy generates a finite saturated set, that may form a basis to build a model [12]. Third, one can apply an existing theorem prover "off the shelf," as was done in [1] with very good results, or else modify it for $\mathcal{T}$-satisfiability, re-using already developed data structures, algorithms and code.

These advantages are balanced by the consideration that the rewrite-based $\mathcal{T}$-satisfiability procedures in [3, 1, 8] are exponential, except those for non-empty possibly cyclic lists [3], records without extensionality and integer offsets modulo [1]. For some theories, the $\mathcal{T}$-satisfiability procedure is exponential, because the problem itself is exponential, as it is the case for arrays. In general, it is clearly desirable to have polynomial procedures and to know if they can be obtained. In the second part of this article, we present *polynomial* rewrite-based $\mathcal{T}$-satisfiability procedures for the theories of *records with extensionality* and *integer offsets*. Thus, all the above advantages can be combined with polynomial complexity for more theories. While other polynomial $\mathcal{T}$-satisfiability procedures for integer offsets are known (e.g., [28]), to the best of our knowledge ours is the first polynomial $\mathcal{T}$-satisfiability procedures for records with extensionality.

The polynomial result for records with extensionality rests on two ingredients that may apply to all variable-inactive theories. The first one is the same analysis of derivable clauses that we use to obtain the $\mathcal{T}$-decision procedures. The second one is a technique where the $\mathcal{SP}$-strategy is used as a pre-processor for part of the problem. The result on integer offsets does not use variable-inactivity, but requires a reduction to *finitize* the problem, because the axiomatization of the theory of integer offsets is *infinite*. A first reduction was devised in [1] and extended to any theory of recursive data structures in [8]. That reduction was based on preprocessing the input set of unit clauses and introducing an *injectivity lemma*, which resulted in an exponential procedure. In this article we show how to select a finite subset of the axioms, while preserving satisfiability of the considered problem. The resulting reduction is cleaner and simpler than that of [1], since it requires neither preprocessing nor introducing additional lemmata. Because it stems from a better understanding of the theory, it yields a polynomial $\mathcal{T}$-satisfiability procedure.

The article is organized as follows: Section 2 recalls basic definitions; Section 3 presents variable-inactivity, the analysis of $\mathcal{SP}$-inferences in variable-inactive theories, and the proof of the termination of $\mathcal{SP}$ on $\mathcal{T}$-decision problems. Sections 4 and 5 show how to obtain polynomial rewrite-based $\mathcal{T}$-satisfiability procedures for the theories of records with extensionality and integer offsets, respectively.

2 Preliminaries

Given a signature Σ, we assume the standard definitions of Σ-terms, Σ-literals, Σ-clauses and Σ-sentences, where Σ is omitted when it is clear from context. Clauses are variable-disjoint and a clause is *positive* (resp. *negative*) if it only contains positive (resp. negative) literals. A *theory* is presented by a set $\mathcal{T}$ of Σ-sentences, called its *presentation* or *axiomatization*. The theory presented by $\mathcal{T}$ is the set of all its theorems: $Th(\mathcal{T}) = \{\varphi \mid \mathcal{T} \models \varphi\}$. For notation, $\simeq$ is unordered equality[1], $\bowtie$ is either $\simeq$ or $\not\simeq$, while $=$ is identity. Lower-case letters l, r, u, v, t denote generic terms, while we reserve w, x, y, z for variables and a, b, c for constants. The notation $l[u]$ represents a term where u appears as subterm in context l. Upper-case letters C, D, C', D' denote clauses or disjunctions of literals, L is used for literals, and S, F, N for sets of clauses. More notation may be introduced as needed.

We consider the standard definition of Σ-*algebras* and Σ-*models*. Given a Σ-algebra $M = (D, I)$, for all function symbols $f \in \Sigma$, $[f]^I$ denotes the interpretation of f, and for all constant symbols $c \in \Sigma$, $I(c)$ denotes the interpretation of c. If M is a model for a formula φ (or, equivalently, a set of clauses S), by the Löwenheim-Skolem theorem, we may assume that the domain D is denumerable. In the sequel, we assume that all considered Σ-algebras have denumerable domain. Since an arbitrary ground formula can be reduced to a set of ground clauses, we have:

Definition 2.1 For a presentation $\mathcal{T}$, a $\mathcal{T}$-*satisfiability problem* is given by $\mathcal{T} \cup S$, where S is a set of ground unit clauses. A $\mathcal{T}$-*decision problem* is given by $\mathcal{T} \cup S$, where S is a set of ground clauses. $\diamond$

In either case, the problem is to decide whether $\mathcal{T} \cup S$ is satisfiable. For sets of clauses S and S', we write $S \equiv_{\mathrm{s}} S'$ to say that S and S' are *equisatisfiable*, that is, S has a model if and only if S' has a model.

For a term t, the *depth* of t, denoted by $\mathrm{depth}(t)$, is 0 if t is a constant or variable, and $\mathrm{depth}(f(t_1, \ldots, t_n)) = 1 + \max\{\mathrm{depth}(t_i) \mid i = 1, \ldots, n\}$ otherwise. For literals, $\mathrm{depth}(l \bowtie r) = \mathrm{depth}(l) + \mathrm{depth}(r)$.

[1] That is, $l \simeq r$ stands for $l \simeq r$ or $r \simeq l$.

Definition 2.2 A positive literal is *flat* if its depth is 0 or 1; a negative literal is *flat* if its depth is 0; a literal is *strictly flat* if its depth is 0. A clause is *flat*, respectively, *strictly flat*, if all its literals are. $\diamond$

The operation of *flattening* consists of transforming a *finite set of ground* Σ*-clauses* S into a finite set of ground clauses $S_1 \uplus S_2$ over a signature Σ', in such a way that:

- Σ' is obtained by adding a finite number of constants to Σ,
- every clause in S_1 is unit and flat,
- every clause in S_2 is strictly flat and
- $\mathcal{T} \cup S \equiv_{\mathrm{s}} \mathcal{T} \cup S_1 \cup S_2$.

For example, the set $S = \{f(a) \not\simeq f(b) \vee f(a) \not\simeq f(c)\}$ can be transformed into the sets of clauses $S_1 = \{f(a) \simeq a',\ f(b) \simeq b',\ f(c) \simeq c'\}$ and $S_2 = \{a' \not\simeq b' \vee a' \not\simeq c'\}$ by introducing fresh constants a', b' and c'. Intuitively, the clauses in S_1 *define* functions, while those in S_2 establish *constraints* on individuals. This flattening operation generalizes that of [3, 1] for unit clauses: if all clauses in S are unit, S_2 is empty.

The Superposition Calculus

A *simplification ordering* $\succ$ is an ordering that is *stable*, *monotonic* and contains the *subterm ordering*: if $u \succ v$, then $l[u]\sigma \succ l[v]\sigma$ for any context l and substitution σ, and if v is a subterm of u then $u \succ v$. A *complete simplification ordering*, or CSO, is a simplification ordering that is total on ground terms. We write $v \prec u$ if $u \succ v$. More details on orderings can be found in surveys such as [15]. We say that a CSO is *good*, if $t \succ c$, whenever t is a compound term and c a constant. This condition was part of the $\mathcal{T}$*-goodness* requirement on the ordering, for all theories considered in [1]. In this article, a theory independent requirement on the ordering suffices, and therefore we call this property simply *goodness.*

A *strategy*, denoted by $\mathfrak{S}$, is given by an inference system and a search plan that controls the application of the inference rules. The *superposition calculus*, or $\mathcal{SP}$, is a *refutationally complete*

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 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): $u\sigma \not\preceq t\sigma$; ***(ii)***: $\forall L \in D : (u \simeq t)\sigma \not\preceq L\sigma$; ***(iii)***: $l[u']\sigma \not\preceq r\sigma$;

(iv): $\forall L \in C : (l[u'] \bowtie r)\sigma \not\preceq L\sigma$; ***(v)***: $\forall L \in C : (u' \not\simeq u)\sigma \not\prec L\sigma$;

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

In standard terminology, $D \vee u \simeq t$ *paramodulates into* $C \vee l[u'] \bowtie r$.

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.

rewrite-based inference system for first-order logic with equality. It consists of *expansion inference rules* (see Figure 1) and *contraction inference rules* (see Figure 2). Since it is based on a CSO on terms and literals, we write $\mathcal{SP}_\succ$ to emphasize the ordering. A strategy with inference system $\mathcal{SP}_\succ$ is called an *$\mathcal{SP}_\succ$-strategy*, and it is said to be *good* if $\succ$ is. From now on, we consider only good CSO's and good $\mathcal{SP}_\succ$-strategies.

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 contraction rules 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 *$\mathcal{SP}_\succ$-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 inference rule to clauses in S_{i-1}. Let

Strict Subsumption $$\begin{array}{c} C \quad D \\ \hline\hline C \end{array} \qquad\qquad D \gtrdot C$$

Simplification $$\begin{array}{c} C[u] \quad l \simeq r \\ \hline\hline C[r\sigma] \quad l \simeq r \end{array} \qquad u = l\sigma,\ l\sigma \succ r\sigma,\ C[u] \succ (l \simeq r)\sigma$$

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 apply also subsumption of variants: if $D \mathrel{\underline{\gtrdot}} C$ and $C \mathrel{\underline{\gtrdot}} D$, 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.

$S_* = \bigcup_i S_i$ be all clauses appearing anywhere in $\{S_i\}_i$. The *limit* of a derivation is the set of *persistent clauses*: $S_\infty = \bigcup_{j \geq 0} \bigcap_{i \geq j} S_i$. If a derivation is finite and of length n, we may write $S_0 \vdash^n_{\mathcal{SP}_\succ} S_n$. A derivation $\{S_i\}_i$ is *sound*, if $S_\infty \subseteq Th(S_i)$, for all i, and it is *adequate*, if, for all i, $S_i \subseteq Th(S_\infty)$.

Since the subset of strictly flat persistent clauses will be relevant in the following, we give:

Definition 2.3 The *core limit* of a derivation $S_0 \vdash_{\mathcal{SP}_\succ} S_1 \vdash_{\mathcal{SP}_\succ} \dots S_i \vdash_{\mathcal{SP}_\succ} \dots$ is the subset $F_\infty \subseteq S_\infty$ defined by $F_\infty = \{C \mid C \in S_\infty \wedge\ C \text{ is strictly flat}\}$. $\diamond$

Rewrite-based inference systems such as $\mathcal{SP}$ enjoy the property that once something becomes redundant during a derivation, it will remain such forever, or *"once redundant, always redundant"* [29, 6]. Thus, if C is redundant in S_i, it is redundant in S_j for all $j > i$ and in S_∞.

Let $\mathfrak{S}$ be an $\mathcal{SP}_\succ$-strategy with search plan $\mathcal{P}$. The sequence $S_0 \vdash_\mathfrak{S} S_1 \vdash_\mathfrak{S} \dots S_i \vdash_\mathfrak{S} \dots$ is the unique derivation generated by $\mathfrak{S}$ from input S_0. For $\mathfrak{S}$ to be complete, $\mathcal{P}$ must be *fair*:

Definition 2.4 A derivation $S_0 \vdash_{\mathcal{SP}_\succ} \dots S_n \vdash_{\mathcal{SP}_\succ} \dots$ 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 $j \geq 0$. $\diamond$

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. We consider only fair and therefore complete $\mathcal{SP}_\succ$-strategies.

If $\mathcal{SP}_\succ$ is guaranteed to terminate on $\mathcal{T}$-satisfiability problems, generating a finite limit, a fair $\mathcal{SP}_\succ$-strategy is a rewrite-based $\mathcal{T}$-satisfiability procedure: adapting a terminology of [10], in such a case $\mathcal{T}$ is said to be *$\exists$-$\mathcal{SP}_\succ$-decidable.* The complexity of a $\mathcal{T}$-satisfiability procedure is expressed as a function of the size of S, measured by the number of subterms occurring in S. For a rewrite-based $\mathcal{T}$-satisfiability procedure, the pre-processing phase, where S is flattened, can be performed in linear time in the size of S. Each $\mathcal{SP}_\succ$-inference step takes polynomial time in the size of the set of clauses during the derivation. Since, by fairness, only inferences from persistent clauses need to be considered, the complexity of the application of the $\mathcal{SP}_\succ$-strategy depends on the size of S_∞: if S_∞ has, in the worst case, exponential or polynomial cardinality in terms of the size of S, the procedure will be exponential or polynomial, respectively.

3 Variable-inactivity

In this section, we analyze the syntactic properties of $\mathcal{SP}_\succ$-derivable clauses, that follow from variable-inactivity and goodness. Then, we use them to construct a rewrite-based $\mathcal{T}$-decision procedures for all theories admitting a rewrite-based $\mathcal{T}$-satisfiability procedure.

Definition 3.1 Given a CSO $\succ$, a clause C is *variable-inactive for* $\succ$, if no maximal literal[2] 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 a satisfiability problem $S_0 = \mathcal{T} \cup S$ is variable-inactive. $\diamond$

When no confusion is possible, we say that a clause (a set of clauses or a presentation, respectively) is variable-inactive, without mentioning $\succ$.

[2] Literal L is *maximal in* C if for all substitutions σ and literals $L' \neq L$ in C, $L\sigma \not\prec L'\sigma$.

Example 3.2 Consider the following three clauses:

$$\begin{aligned} C_1 &= \mathsf{car}(\mathsf{cons}(x, y)) \simeq x, \\ C_2 &= z \simeq w \vee \mathsf{select}(\mathsf{store}(x, z, v), w) \simeq \mathsf{select}(x, w), \\ C_3 &= \bigvee_{1 \leq j < k \leq n} (x_j \simeq x_k). \end{aligned}$$

Clause C_1 is the axiom of the theory of lists that states that the car function returns the first element of a list. This clause is variable-inactive, since the variable x appears in $\mathsf{car}(\mathsf{cons}(x, y))$. Clause C_2 is the axiom of the theory of arrays that says that if array x is updated at position z with element v, all its other elements remain unchanged. This clause is also variable-inactive, although it contains a literal $z \simeq w$ where $z \notin \mathrm{Var}(w)$, because both z and w occur in $\mathsf{select}(\mathsf{store}(x, z, v), w)$, so that $z \simeq w$ cannot be maximal by the subterm property of CSO's. Clause C_3 is a *finite cardinality axiom*: the domain of any model of this clause has cardinality at most n. C_3 is obviously not variable-inactive.

3.1 Analysis of inferences

Given a variable-inactive clause C, the following lemma provides a syntactic characterization of a maximal term in a maximal literal of C. In Lemmata 3.3 and 3.4 we assume that the clauses under consideration contain no literals of the form $x \bowtie x$. There is no loss of generality under this assumption: any clause of the form $x \simeq x \vee C$ can be deleted by the deletion rule; any clause of the form $x \not\simeq x \vee C$ can be replaced by the clause C, generated by reflection from $x \not\simeq x \vee C$, and applied to subsume $x \not\simeq x \vee C$.

Lemma 3.3 *Let C be a variable-inactive clause and $L = l \bowtie r$ be a maximal literal in C, where l is maximal in L:*

1. *If C is strictly flat then l and r are both constants;*
2. *If C is not strictly flat then l is a compound term.*

Proof. As a preliminary, we prove that l is either a compound term or a constant. By way of contradiction, suppose l is a variable. Then, since L is maximal, by variable-inactivity, l must be in $\mathrm{Var}(r)$. By assumption, L is not of the form $l \bowtie l$, thus l is a strict subterm of r. Since $\succ$ has the subterm property we must have $r \succ l$, which contradicts the assumption that l is maximal. We prove next the two claims.

1. Assume that C is strictly flat. By the preceding argument l must be a constant. As for r, if it were a variable, by variable-inactivity it should be $r \in \mathrm{Var}(l)$, which is impossible since l is a constant. Thus, r is a constant as well.

2. Assume that C is not strictly flat. By way of contradiction, suppose l is a constant. If r were a compound term, then, by goodness, $r \succ l$, which contradicts the fact that l is maximal in L. Thus, r is either a variable or a constant. If r were a variable, since L is maximal, it should be $r \in \mathrm{Var}(l)$ by variable-inactivity, which is impossible because l is a constant by assumption. Therefore, r is necessarily a constant and L is strictly flat. Since C is not strictly flat by hypothesis, it must contain another literal $L' = l' \bowtie r'$, where one side, say l', is a compound term. By goodness, it must be $l' \succ l$ and $l' \succ r$, contradicting the maximality of L in C. ■

Using Lemma 3.3, we determine which binary $\mathcal{SP}_\succ$-inferences that generate clauses apply when one of the premises is strictly flat. Such inferences can be superpositions, paramodulations or simplifications (see Figures 1 and 2); for simplicity we write paramodulation in all cases.

Lemma 3.4 *Let C be a variable-inactive clause and C' a strictly flat ground clause.*

1. *If C' paramodulates into C, then $C = l[a] \bowtie r \vee D$, $C' = a \simeq a'' \vee D'$ and the generated clause is $l[a''] \bowtie r \vee D \vee D'$.*

2. *If C paramodulates into C', then $C = a \simeq a' \vee D$, $C' = a \bowtie a'' \vee D'$, C is also strictly flat, and the generated clause is $a' \bowtie a'' \vee D \vee D'$, which is strictly flat as well.*

Thus, the mgu of a binary $\mathcal{SP}_\succ$-inference on C and C' is necessarily the empty substitution.

Proof. If C' paramodulates into C, since we do not paramodulate into variables, it must be $C' = a \simeq a'' \vee D'$ and $C = l[a] \bowtie r \vee D$. Hence, the corresponding mgu is empty and the generated clause is $l[a''] \bowtie r \vee D \vee D'$. If C paramodulates into C', since C' is strictly flat and

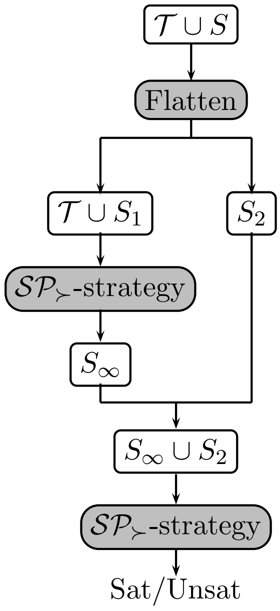

Figure 3: Solving $\mathcal{T}$-decision problems by the rewrite-based approach.

ground, its maximal literal has the form $a \bowtie a''$ and $C' = a \bowtie a'' \vee D'$ for D' also strictly flat. Let $L = u \simeq v$ be the considered maximal literal in C, and u be maximal in L. To paramodulate into a constant, u cannot be a compound term, and by Lemma 3.3(2), C must be strictly flat. Then, by Lemma 3.3(1), u and v must be constants. Hence, C is of the form $a \simeq a' \vee D$, where D is strictly flat. The mgu of the paramodulation of C into C' is therefore empty, and the generated clause is $a' \bowtie a'' \vee D \vee D'$ which is strictly flat. ■

In order to apply these lemmata, from now on we assume that all considered presentations $\mathcal{T}$ do not contain literals in the form $x \bowtie x$. Clearly, this is not a significant restriction.

3.2 Rewrite-based $\mathcal{T}$-decision procedures

In this section, we use the previous analysis of possible inferences to prove that variable-inactivity guarantees termination of a strategy on $\mathcal{T}$-decision problems, provided it terminates on $\mathcal{T}$-satisfiability problems. Figure 3 shows how the strategy is applied (recall that S_1 is unit and flat and S_2 is strictly flat).

Theorem 3.5 *If $\mathcal{T}$ is variable-inactive and $\exists$-$\mathcal{SP}_{\succ}$-decidable, any fair $\mathcal{SP}_{\succ}$-strategy $\mathfrak{S}$ is a $\mathcal{T}$-*

decision procedure.

Proof. We prove the claim by showing that $\mathfrak{S}$ terminates on a $\mathcal{T}$-decision problem $\mathcal{T} \cup S$, following the scheme of Figure 3. S is flattened into $S_1 \uplus S_2$ as described in Section 2. Since $\mathcal{T}$ is $\exists$-$\mathcal{SP}_{\succ}$-decidable, when $\mathfrak{S}$ is applied to $\mathcal{T} \cup S_1$, it generates a finite limit S_∞ (first application of the $\mathcal{SP}_{\succ}$-strategy in Figure 3). Consider next the union $S_\infty \cup S_2$ and the second application of the $\mathcal{SP}_{\succ}$-strategy in Figure 3. To prove termination, we only need to consider expansion inferences and simplifications. Let $C \in S_\infty$ and $C' \in S_2$. As far as unary inferences are concerned, no non-redundant unary inference applies to C, since C comes from S_∞ and all inferences are redundant in S_∞. If a unary inference applies to C', the corresponding mgu is empty and a strictly flat clause is generated. For binary inferences, C and C' satisfy the requirements of Lemma 3.4. If a binary inference applies to C and C', the generated clause is one of those described in Lemma 3.4 with empty mgu. Thus, every possible inference involves a constant paramodulated into and from. Since there are finitely many constants in $S_\infty \cup S_2$, only finitely many inferences may apply, and the second application of the $\mathcal{SP}_{\succ}$-strategy in Figure 3 also terminates. If it generates the empty clause, the procedure returns Unsat, and Sat otherwise. ■

Based on this result, it is possible to solve $\mathcal{T}$-decision problems by applying the scheme of Figure 3. This scheme works for all the theories considered in [3, 1, 8, 7], including those of records and integer offsets, that we study in greater detail in the second part of the article.

4 A polynomial procedure for records with extensionality

In this section, we use the above analysis to devise a polynomial $\mathcal{T}$-satisfiability procedure for the *theory of records with extensionality.* We start from the following observation: consider Figure 3 and assume that the set S_2 of strictly flat ground clauses contains only *negative* clauses. Since negative clauses can be only paramodulated into, S_2 plays a merely passive role with respect to expansion, when the $\mathcal{SP}_{\succ}$-strategy is applied to $S_\infty \cup S_2$ in Figure 3. Furthermore, by Lemma 3.4(2), we know that a clause $C \in S_\infty$ that paramodulates into a clause $C' \in S_2$ must be strictly flat and therefore in F_∞, which represents the "active core", sort of speak, of the limit. If F_∞ is made of *ground unit* clauses, none of them can be strictly subsumed by a clause in S_2, so that S_2 plays a passive role also with respect to contraction. Furthermore, the union $F_\infty \cup S_2$ is a set of Horn clauses, whose satisfiability can be tested in polynomial time (cf., e.g., [19, Theorem 4]). Thus, if the $\mathcal{T}$-satisfiability of the set S_1 of flat unit clauses is decided in polynomial time, the $\mathcal{T}$-satisfiability of $S_1 \cup S_2$ also will be decided in polynomial time, because the only additional work is given by $F_\infty \cup S_2$.

As this may look like a strong collection of requirements, we begin by seeing how satisfiability problems in the theory of records with extensionality offer precisely this kind of situation. In order to make this article self-contained and fully readable, we reproduce in the following two lemmata and one theorem from [1].

Let $\textsc{t}_1, \ldots, \textsc{t}_n$ be sorts, and let $\textsc{rec}$ be the sort of records with n fields of sort $\textsc{t}_1, \ldots, \textsc{t}_n$, respectively. Signature Σ is the sorted signature that contains, for all i, $1 \leq i \leq n$, the function symbols $\mathrm{rstore}_i : \textsc{rec} \times \textsc{t}_i \to \textsc{rec}$, which stores a value of sort $\textsc{t}_i$ in the ith-field of the given record, and $\mathrm{rselect}_i : \textsc{rec} \to \textsc{t}_i$, which extracts a value of sort $\textsc{t}_i$ from the ith-field of the given record. The theory of records is defined by the following presentation, denoted by $\mathcal{R}$:

$$\begin{array}{lcl} \forall x, v. & \mathrm{rselect}_i(\mathrm{rstore}_i(x, v)) \simeq v & \text{for all } 1 \leq i \leq n, \\ \forall x, v. & \mathrm{rselect}_j(\mathrm{rstore}_i(x, v)) \simeq \mathrm{rselect}_j(x) & \text{for all } 1 \leq i \neq j \leq n. \end{array}$$

The theory of records with extensionality is axiomatized by the presentation $\mathcal{R}^e$, which consists of the previous axioms together with:

$$\forall x, y. \quad \left(\bigwedge_{i=1}^{n} \mathrm{rselect}_i(x) \simeq \mathrm{rselect}_i(y) \right) \Rightarrow x \simeq y.$$

In this section we are concerned with satisfiability problems, and therefore the given problem is a set of ground unit clauses, or, equivalently, ground literals, which is reduced by flattening to a set S of ground flat literals. It is known that $\mathcal{SP}_\succ$ is guaranteed to terminate on problems in the form $S_0 = \mathcal{R} \cup S$, generating a *finite*, *variable-inactive* limit S_∞ with a core F_∞ made of *ground unit* clauses:

Lemma 4.1 (Lemma 2 of [1]) *If $S_0 = \mathcal{R} \cup S$, where S is a set of ground flat literals, all clauses in S_∞ are unit and are of the following kinds, where r, r' are constants of sort* $\textsc{rec}$*, and e, e' are constants of sort $\textsc{t}_i$ for some i, $1 \leq i \leq n$: the empty clause; the clauses in $\mathcal{R}$;*

$\mathrm{rselect}_i(\mathrm{rstore}_i(x, v)) \simeq v$, *$\forall i$, $1 \le i \le n$;* $\mathrm{rselect}_j(\mathrm{rstore}_i(x, v)) \simeq \mathrm{rselect}_j(x)$, *$\forall i, j$, $1 \le i \ne j \le n$; ground flat unit clauses of the form:* $r \simeq r'$, $e \simeq e'$, $e \not\simeq e'$, $\mathrm{rstore}_i(r, e) \simeq r'$, *for some i, $1 \le i \le n$,* $\mathrm{rselect}_i(r) \simeq e$, *for some i, $1 \le i \le n$;* $\mathrm{rselect}_i(r) \simeq \mathrm{rselect}_i(r')$, *for some i, $1 \le i \le n$.*

Since S_∞ has cardinality at most quadratic in the number of subterms in S, an $\mathcal{SP}_\succ$-strategy is a polynomial $\mathcal{R}$-satisfiability procedure (cf. [1, Theorem 2]). The following reduction of $\mathcal{R}^e$ to $\mathcal{R}$ was preliminarily established:

Lemma 4.2 (Lemma 1 of [1]) *Let $S = S' \uplus S_N$ be a set of ground flat literals, where S_N contains all the literals of the form $l \not\simeq r$, with l and r of sort* REC. *For all $L = l \not\simeq r \in S_N$ let $C_L = \bigvee_{i=1}^{n} \mathrm{rselect}_i(l) \not\simeq \mathrm{rselect}_i(r)$ be its associated clause. Then $\mathcal{R}^e \cup S \equiv_{\mathrm{s}} \mathcal{R} \cup S' \cup \{C_L \,|\, L \in S_N\}$.*

The $\mathcal{R}$-decision problem $S' \cup \{C_L \,|\, L \in S_N\}$ can be converted into disjunctive normal form, thereby reducing $\mathcal{R}^e$-satisfiability to $\mathcal{R}$-satisfiability. However, the $\mathcal{R}^e$-satisfiability procedure including this reduction is exponential:

Theorem 4.3 (Theorem 2 of [1]) *A fair $\mathcal{SP}_\succ$-strategy is a polynomial satisfiability procedure for $\mathcal{R}$ and an exponential satisfiability procedure for $\mathcal{R}^e$.*

As suggested at the beginning of this section, the crucial observation is that the clauses in $\{C_L \mid L \in S_N\}$ are negative. The flattening of $\{C_L \mid L \in S_N\}$ yields $S'' \cup N$, where S'' is a set of ground flat unit clauses and N is a set of ground, strictly flat and negative clauses. Thus, $\mathcal{R} \cup S' \cup \{C_L \,|\, L \in S_N\}$ is transformed into $\mathcal{R} \cup (S' \cup S'') \cup N$. Let S_∞ be the finite limit generated from $\mathcal{R} \cup (S' \cup S'')$ and F_∞ be its core. The central step is to prove that $\mathcal{R} \cup (S' \cup S'') \cup N \equiv_{\mathrm{s}} F_\infty \cup N$. Then, since $F_\infty \cup N$ is ground and Horn, its satisfiability, and, by Lemma 4.2, that of $\mathcal{R}^e \cup S$, can be decided in polynomial time.

Definition 4.4 A presentation $\mathcal{T}$ has *ground unit core limit*, if for all fair $\mathcal{SP}_\succ$-derivation from a satisfiability problem $S_0 = \mathcal{T} \cup S$, the set F_∞ is made of ground unit clauses. $\diamond$

In the following, we develop formally the above reasoning, not only for $\mathcal{R}$, but for any $\mathcal{T}$ that is (1) variable-inactive, (2) $\exists$-$\mathcal{SP}_\succ$-decidable, so that for any satisfiability problem $S_0 = \mathcal{T} \cup S$, the limit S_∞ is finite, and (3) with ground unit core limit F_∞.

Lemma 4.5 *For all sets N of ground, strictly flat and negative clauses, all persistent clauses generated from $S_\infty \cup N$ are generated from $F_\infty \cup N$ and are ground, strictly flat and negative.*

Proof. Since all inferences within S_∞ are redundant and all unary inferences within N (i.e., reflections) generate ground, strictly flat and negative clauses, we only need to consider binary inferences between $C \in S_\infty$ and $C' \in N$. Since C' is negative, a binary inference is necessarily a paramodulation of C into C' or a simplification of C' by C. Since C is variable-inactive, by assumption (1) on the presentation, and C' is strictly flat by hypothesis, Lemma 3.4 applies. By Lemma 3.4(2), C is also strictly flat, so that $C \in F_\infty$, and all clauses generated from $S_\infty \cup N$ are generated from $F_\infty \cup N$. Again by Lemma 3.4(2), $C = a \simeq a' \vee D$, $C' = a \not\simeq a'' \vee D'$ and the generated clause $E = a' \not\simeq a'' \vee D \vee D'$ is strictly flat. By assumption (3) on the core limit, C is unit, so that $D = \emptyset$, and E is also ground and negative. ■

Lemma 4.6 *For all presentations $\mathcal{T}$, satisfying the preceding conditions* (1), (2), (3), *sets of ground flat literals S, and sets N of ground, strictly flat and negative clauses, $\mathcal{T} \cup S \cup N \equiv_{\mathrm{s}} F_\infty \cup N$.*

Proof. If $F_\infty \cup N$ is unsatisfiable, then such are $S_\infty \cup N$, because $F_\infty \subseteq S_\infty$, and $\mathcal{T} \cup S \cup N$, by soundness of $\mathcal{SP}$. If $\mathcal{T} \cup S \cup N$ is unsatisfiable, then such is $S_\infty \cup N$ by adequacy of $\mathcal{SP}$. We have to prove that $F_\infty \cup N$ is unsatisfiable. If the unsatisfiability of $\mathcal{T} \cup S \cup N$ were due to $\mathcal{T} \cup S$, then $S_\infty = F_\infty = \{\Box\}$ by the refutational completeness of $\mathcal{SP}$, and the result is trivial. The non-trivial situation is the one where $\mathcal{T} \cup S \cup N$ is unsatisfiable, but $\mathcal{T} \cup S$ is satisfiable, so that $\Box \notin S_\infty$. We consider a fair $\mathcal{SP}_\succ$-derivation from $S_\infty \cup N$. Since $S_\infty \cup N$ is unsatisfiable and $\mathcal{SP}$ refutationally complete, this derivation is bound to terminate generating $\Box$. We prove that $F_\infty \cup N$ is unsatisfiable, by showing that $\Box$ must be generated from $F_\infty \cup N$. Let $N_0 = N$, $S^1_0 = S_\infty \cup N_0$ and $S^1_0 \vdash_{\mathcal{SP}_\succ} \ldots \vdash_{\mathcal{SP}_\succ} S^1_k$ be the fair derivation with $\Box \in S^1_k$. Let $S^2_0 = F_\infty \cup N_0$ and $S^2_0 \vdash_{\mathcal{SP}_\succ} \ldots \vdash_{\mathcal{SP}_\succ} S^2_i \vdash_{\mathcal{SP}_\succ} \ldots$ be also a fair derivation. Furthermore, let $S^2_* = \bigcup_i S^2_i$ and $N_* = \{C \in S^2_* \mid C$ is ground, strictly flat and negative$\}$. We prove by induction on i that $\forall i \geq 0$,

1. $S^1_i = S_\infty \cup N_i$, where $N_i = \{C \in S^1_i \mid C$ is ground, strictly flat and negative$\}$ and
2. all persistent clauses in N_i are in N_*.

Since $\Box \notin S_\infty$ and $\Box \in S^1_k$, it will follow that $\Box \in N_k$, and since $\Box$ is persistent, it will follow that

$\Box \in N_* \subseteq S_*^2$, so that $F_\infty \cup N$ is unsatisfiable. For the base case $i = 0$, we have $S_0^1 = S_\infty \cup N_0$ and $N_0 = N \subseteq N_*$ and the result is obvious. For the induction step, we assume that the preceding Claims 1 and 2 hold for i, and we show that they hold for $i{+}1$, where $S_i^1 \vdash_{\mathcal{SP}_\succ} S_{i+1}^1$. By Lemma 4.5, all persistent clauses generated from $S_\infty \cup N_i$ are generated from $F_\infty \cup N_i$ and are ground, strictly flat and negative. It follows that S_{i+1}^1 also has the form $S_\infty \cup N_{i+1}$ and all persistent clauses in N_{i+1} are in N_*. Contraction inferences do not interfere, because: (1) simplification is included in Lemma 4.5; (2) deletion applies neither to a clause in S_∞ (if it could, it would have in the derivation that generated S_∞) nor to a clause in N_i (it does not apply to negative clauses); (3) subsumption of a clause in N_i by a clause in S_∞ is harmless, and if a clause in N_i subsumes strictly a clause $C \in S_\infty$, then $C \notin F_\infty$, because F_∞ is made of ground unit clauses, and therefore such a subsumption step is also harmless with respect to the generation of persistent clauses by Lemma 4.5. ■

By assumption (3) on the core limit, $F_\infty \cup N$ is a set of ground Horn clauses, whose satisfiability can be tested in polynomial time:

Theorem 4.7 *For all presentations $\mathcal{T}$ satisfying the preceding conditions* (1), (2), (3)*, sets S of ground, flat unit clauses and sets N of ground, strictly flat and negative clauses, if a fair $\mathcal{SP}_\succ$-strategy $\mathfrak{S}$ is a polynomial $\mathcal{T}$-satisfiability procedure, then the satisfiability of $\mathcal{T} \cup S \cup N$ can be decided in polynomial time.*

Proof. For all $\mathcal{T}$-satisfiability problem $\mathcal{T} \cup S$, by hypothesis, $\mathfrak{S}$ generates a limit set S_∞ in polynomial time. F_∞ can be extracted from S_∞ in polynomial time and the satisfiability of $F_\infty \cup N$ can be decided also in polynomial time. ■

Since $\mathcal{R}$ satisfies conditions (1), (2) and (3) (cf. Lemma 4.1 and Theorem 4.3), Theorem 4.7 applies to the theory of records with extensionality:

Corollary 4.8 *A fair $\mathcal{SP}_\succ$-strategy yields a polynomial $\mathcal{R}^e$-satisfiability procedure.*

This result is an instance of a framework where an $\mathcal{SP}_\succ$-strategy is applied first to pre-process part of the problem (e.g., $\mathcal{T} \cup S$), and then to the output of pre-processing (e.g., F_∞) and the remaining part (e.g., N) of the original problem. Such an incremental scheme may find application in other contexts as well.

5 A polynomial procedure for integer offsets

The theory of integer offsets is a fragment of the theory of integers. Its signature Σ contains two unary function symbols s and p, that represent the successor and predecessor functions, respectively. This theory is presented by the following (infinite) set of axioms $\mathcal{I}$:

$$\begin{array}{rcl} \forall x.\ \ \mathsf{s}(\mathsf{p}(x)) & \simeq & x, \\ \forall x.\ \ \mathsf{p}(\mathsf{s}(x)) & \simeq & x, \\ \forall x.\ \ \mathsf{s}^i(x) & \not\simeq & x \text{ for } i > 0, \end{array}$$

where $\mathsf{s}^0(x) = x$ and $\mathsf{s}^{i+1}(x) = \mathsf{s}(\mathsf{s}^i(x))$ for $i \geq 0$. For convenience, we define the following sets of clauses:

$$\begin{array}{rcl} A_{\mathcal{I}} & = & \{\mathsf{s}(\mathsf{p}(x)) \simeq x,\ \mathsf{p}(\mathsf{s}(x)) \simeq x\}, \\ Ac(n) & = & \{\mathsf{s}^i(x) \not\simeq x \mid 0 < i \leq n\}, \\ Ac & = & \bigcup_{n\geq 0} Ac(n). \end{array}$$

An $\mathcal{I}$-satisfiability problem is given by a union $A_{\mathcal{I}} \cup Ac \cup S$, where S consists of ground flat literals. Since Ac is infinite, such a set cannot be fed to a theorem prover. The goal of this section is to determine a finite set that is equisatisfiable with $A_{\mathcal{I}} \cup Ac \cup S$ and such that an $\mathcal{SP}_{\succ}$-strategy is guaranteed to terminate and produce a limit of polynomial size. More specifically, we will show that Ac can be safely replaced by a finite subset $Ac(n)$, where n depends on S.

5.1 M-paths and cyclic M-paths

We introduce the notion of *M-path* in a Σ-algebra M. Intuitively, an M-path is a sequence of elements that are linked by the successor function; if M satisfies the acyclicity axioms, none of these M-paths will be *cyclic*.

Definition 5.1 Let $M = (D, I)$ be a Σ-algebra. For all $m \geq 2$, a tuple $p = \langle d_1, d_2, \ldots, d_m\rangle \in D^m$ is an *M-path* if, for all i, j, $1 \leq i < j \leq m$, $d_i \neq d_j$ and $d_{i+1} = [\mathsf{s}]^I(d_i)$ for $i \leq m-1$. The *set* *of elements in* p, denoted by $\mathrm{El}(p)$, is $\mathrm{El}(p) = \{d_1, \ldots, d_m\}$. The *length* of p is $|\mathrm{El}(p)|$, and p is *cyclic* if $[\mathsf{s}]^I(d_m) = d_1$. ◇

There is an M-path of length m from d to d' if $[\mathsf{s}^m]^I(d) = d'$, and for all $j < m$, $[\mathsf{s}^j]^I(d) \not\simeq d'$. Since $d = d'$ implies that M violates axiom $\mathsf{s}^m(x) \not\simeq x$, the following property holds trivially:

Proposition 5.2 *M is a model of $Ac(l)$ if and only if for all cyclic M-path p, $|\mathrm{El}(p)| > l$. M has no cyclic M-paths if and only if M is a model of Ac.*

If $M = (D, I)$ is a model of $A_{\mathcal{I}}$, then the functions $[\mathsf{s}]^I$ and $[\mathsf{p}]^I$ are bijections, and for all cyclic M-path p, $\mathrm{El}(p)$ is closed under $[\mathsf{s}]^I$ and $[\mathsf{p}]^I$:

Lemma 5.3 *If $M = (D, I) \models A_{\mathcal{I}}$ and p is a cyclic M-path, then for all $d \in \mathrm{El}(p)$, we have $[\mathsf{s}]^I(d) \in \mathrm{El}(p)$ and $[\mathsf{p}]^I(d) \in \mathrm{El}(p)$.*

Proof. Since $M \models A_{\mathcal{I}}$, the functions $[\mathsf{s}]^I$ and $[\mathsf{p}]^I$ are one the inverse of the other and are therefore bijections. It follows that for all $d \in D$, there exist a unique $d' \in D$, such that $[\mathsf{s}]^I(d') = d$ (and $[\mathsf{p}]^I(d) = d'$) and a unique $d'' \in D$, such that $[\mathsf{p}]^I(d'') = d$ (and $[\mathsf{s}]^I(d) = d''$). If $d \in \mathrm{El}(p)$, then d' and d'' also must be in $\mathrm{El}(p)$. ■

Under the same hypothesis, the interpretations of the successor and predecessor functions induce a partition of D:

Lemma 5.4 *Let $M = (D, I)$ be a model of $A_{\mathcal{I}}$ and let P be the set of all cyclic M-paths. Then for all $p, q \in P$, we have $\mathrm{El}(p) \cap \mathrm{El}(q) = \emptyset$. Furthermore, let $D' = D \setminus (\biguplus_{p \in P} \mathrm{El}(p))$. Then for all $d \in D$,*

$$(d \in D') \;\Leftrightarrow\; (\forall m \geq 1,\, [\mathsf{s}^m]^I(d) \not\simeq d) \;\Leftrightarrow\; (\forall m \geq 1,\, [\mathsf{s}^m]^I(d) \in D').$$

Proof. These claims are immediate consequences of Lemma 5.3. ■

5.2 Breaking cyclic M-paths

We will show how to construct a model for $A_{\mathcal{I}} \cup Ac \cup S$ starting from a model M for $A_{\mathcal{I}} \cup Ac(n) \cup S$, by safely *breaking* all cyclic M-paths. This will be possible provided n is at least the cardinality of the set defined as follows:

Definition 5.5 For every set S of ground flat Σ-literals, let C_S denote the set

$$\mathrm{C}_S \quad = \quad \{c \in \Sigma \,|\, \mathsf{s}(c) \simeq c' \in S \,\vee\, \mathsf{p}(c') \simeq c \in S\}.$$

Let $M = (D, I)$ be a model of S. An element $d \in D$ is *M-constrained by S* if there is a $c \in \mathrm{C}_S$ such that $I(c) = d$. $\diamond$

The cardinality of C_S is thus equal to the number of constants whose successor is defined by the constraints in S. If an element $d \in D$ is not M-constrained by S, then its image by $[\mathsf{s}]^I$ is not constrained to be the interpretation of a constant in S. We will use this property in Theorem 5.7 to break cyclic M-paths and retain satisfiability. The following proposition states a condition guaranteeing the existence of such an element in an M-path:

Proposition 5.6 *Let $M = (D, I)$ be a model of a set of ground literals S. If p is an M-path of length $n \geq |\mathrm{C}_S| + 1$, then one of the elements appearing in p is not M-constrained by S.*

Proof. Let $p = \langle d_1, \ldots, d_n \rangle$ and suppose that every element appearing in p is M-constrained by S. Then, by Definition 5.5, for all j, $1 \leq j \leq n$, if $I(c_j) = d_j$, then one of $\mathsf{s}(c_j) \simeq c'_j$ or $\mathsf{p}(c'_j) \simeq c_j$ appears in S. By hypothesis, since p is an M-path, the d_j's are all distinct, hence there must be at least n distinct constants c_j in C_S, or $|\mathrm{C}_S| \geq n$. This contradicts the hypothesis that $|\mathrm{C}_S| < n$. ■

We now prove the main theorem:

Theorem 5.7 *Let S be a set of ground flat literals. For all n, $n \geq |\mathrm{C}_S|$, if $A_{\mathcal{I}} \cup Ac(n) \cup S$ is satisfiable, then $A_{\mathcal{I}} \cup Ac \cup S$ is satisfiable.*

Proof. Let $M = (D, I)$ be a model of $A_{\mathcal{I}} \cup Ac(n) \cup S$ and let P denote the set of cyclic M-paths. To every $p \in P$ we associate two infinite, denumerable and disjoint sets E_p and E'_p. We further assume that these sets are disjoint from D, and that for every $p, q \in P$, if $p \neq q$ then $(E_p \uplus E'_p) \cap (E_q \uplus E'_q) = \emptyset$.

Let p be a cyclic M-path. By Proposition 5.2, p is of length at least $n + 1$ and by Proposition 5.6, there exists an element $d \in \mathrm{El}(p)$ that is not M-constrained by S. Let $d' \in D$ be the element such that $[\mathsf{s}]^I(d) = d'$ (see Figure 4). Furthermore, let $E_p = \{e_j \,|\, j \geq 0\}$, $E'_p = \{e'_j \,|\, j \geq 0\}$ and

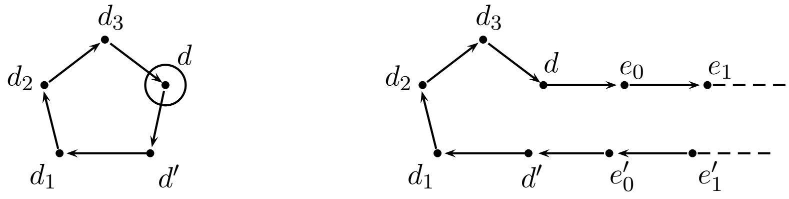

Figure 4: Breaking a cyclic M-path: an arrow from, e.g., d_1 to d_2, indicates that $[\mathsf{s}]^I(d_1) = d_2$, and the circled element d is assumed not to be M-constrained by S.

$D_p = \mathrm{El}(p) \uplus E_p \uplus E'_p$. We consider the algebra $M_p = (D_p, I_p)$ such that I_p interprets constants in the same way as I does, and:

- $\forall d'' \in \mathrm{El}(p) \setminus \{d\}$, $[\mathsf{s}]^{I_p}(d'') = [\mathsf{s}]^I(d'')$,
- $\forall d'' \in \mathrm{El}(p) \setminus \{d'\}$, $[\mathsf{p}]^{I_p}(d'') = [\mathsf{p}]^I(d'')$,
- $[\mathsf{s}]^{I_p}(d) = e_0$ and $[\mathsf{s}]^{I_p}(e'_0) = d'$,
- $[\mathsf{p}]^{I_p}(e_0) = d$ and $[\mathsf{p}]^{I_p}(d') = e'_0$,
- $\forall j \geq 0$, $[\mathsf{s}]^{I_p}(e_j) = e_{j+1}$ and $[\mathsf{s}]^{I_p}(e'_{j+1}) = e'_j$;
- $\forall j \geq 0$, $[\mathsf{p}]^{I_p}(e_{j+1}) = e_j$ and $[\mathsf{p}]^{I_p}(e'_j) = e'_{j+1}$.

Note that if $e \in \mathrm{El}(p)$ is M-constrained by S, then $[\mathsf{s}]^{I_p}(e) = [\mathsf{s}]^I(e)$. Also, by construction, we have $M_p \models A_{\mathcal{I}} \cup Ac$.

Consider the two sets

$$D' \;=\; D \setminus (\biguplus_{p \in P} \mathrm{El}(p)) \quad \text{and} \quad E \;=\; D' \uplus (\biguplus_{p \in P} D_p),$$

and let J be the following interpretation function: for every constant c, $J(c) = I(c)$, and the function $[\mathsf{s}]^J$ (resp. $[\mathsf{p}]^J$) is defined for every $d \in E$ by:

- if $d \in D'$ then $[\mathsf{s}]^J(d) = [\mathsf{s}]^I(d)$ and $[\mathsf{p}]^J(d) = [\mathsf{p}]^I(d)$,

- if $d \in D_p$ for some $p \in P$, then $[\mathsf{s}]^J(d) = [\mathsf{s}]^{I_p}(d)$ and $[\mathsf{p}]^J(d) = [\mathsf{p}]^{I_p}(d)$.

The Σ-algebra $M' = (E, J)$ is well defined and it is a model of $A_{\mathcal{I}} \cup Ac \cup S$:

$M' \models A_{\mathcal{I}}$**:** This is obvious: for example, let $d \in D_p$. Then by construction, $[\mathsf{s}]^J([\mathsf{p}]^J(d)) = [\mathsf{s}]^J([\mathsf{p}]^{I_p}(d)) = [\mathsf{s}]^{I_p}([\mathsf{p}]^{I_p}(d)) = d$.

$M' \models Ac$**:** Suppose there exists a $d \in E$ and an $m \geq 1$ such that $[\mathsf{s}^m]^J(d) = d$. If $d \in D'$ then by applying Lemma 5.4 to the model M, we deduce that $[\mathsf{s}^m]^I(d) \neq d$. Since $[\mathsf{s}^m]^J(d) = [\mathsf{s}^m]^I(d)$, we obtain a contradiction. Thus, d cannot be in D'. It cannot be in any D_p either, since for every p, every M_p-path is acyclic. Therefore, there can be no $d \in E$ and $m \geq 1$ such that $[\mathsf{s}^m]^J(d) = d$, which proves the result.

$M' \models S$**:** Since $J(c) = I(c)$ for every constant, it is clear that M' satisfies all literals $c \bowtie c' \in S$. Assume S contains a literal $\mathsf{s}(c) \simeq c'$; the case where the literal is of the form $\mathsf{p}(c) \simeq c'$ is similar. Let $d = J(c)$ and assume first that $d \in D'$. Then we have $[\mathsf{s}]^J(d) = [\mathsf{s}]^I(d) = I(c') = J(c')$. Now suppose that d is in some D_p. Then d is M-constrained by S, and by definition of I_p we have $[\mathsf{s}]^J(d) = [\mathsf{s}]^{I_p}(d) = [\mathsf{s}]^I(d) = I(c') = J(c')$. ■

In the worst case, each occurrence of s or p in S introduces a distinct constant in C_S, and in such case $|\mathrm{C}_S|$, and therefore the number of acyclicity axioms to be added, is given by the number of occurrences of s and p.

5.3 Finite saturated sets

We show that a polynomial number of clauses is generated by the superposition calculus from a set $A_{\mathcal{I}} \cup Ac(n) \cup S$, by analyzing the possible inferences in an $\mathcal{SP}_{\succ}$-derivation. The proof that the number of persistent clauses is polynomial might also be obtained by applying the results of [24] on meta-saturation, but the following detailed analysis allows us to get an additional result on the tightness of the resulting upper-bound with little additional effort.

Lemma 5.8 *If $S_0 = A_{\mathcal{I}} \cup Ac(n) \cup S$, where S is a set of ground flat literals and $n = |\mathrm{C}_S|$, then all clauses in S_∞ are unit and belong to one of the following categories:*

i) the empty clause,

ii) the clauses in $A_{\mathcal{I}}$, i.e.,

a) $\mathsf{p}(\mathsf{s}(x)) \simeq x$,

b) $\mathsf{s}(\mathsf{p}(x)) \simeq x$,

iii) clauses of the form $\mathsf{s}^i(x) \not\simeq \mathsf{p}^j(x)$, *where* $i \geq 0$, $j \geq 0$ *and* $1 \leq i + j \leq n$,

iv) ground unit clauses of the form

a) $c \simeq c'$,

b) $\mathsf{s}(c) \simeq c'$,

c) $\mathsf{p}(c) \simeq c'$,

v) clauses of the form $\mathsf{s}^i(c) \not\simeq \mathsf{p}^j(c')$, *where* $i \geq 0$, $j \geq 0$ *and* $0 \leq i + j \leq n - 1$.

Proof. The proof is by induction on the sequence $\{S_i\}_i$. For the base case, the result is obvious: all clauses in S_0 belong to categories (ii), (iii) with $j = 0$, (iv) and (v) with $i = j = 0$. For the induction case, we assume the claim is true for i and we prove it for $i + 1$. If the inference performed at stage i is a subsumption or a deletion, the result is obvious. A reflection can apply only to a clause in category (v), with $i = j = 0$, and it generates the empty clause. Since equational factoring does not apply to unit clauses, we are left with simplification, superposition or paramodulation, that we term collectively paramodulation as done earlier:

Paramodulations from (ii): consider clause (ii.a). A paramodulation into clause (ii.b) generates the trivial clause $\mathsf{s}(x) \simeq \mathsf{s}(x)$, which is removed by the deletion rule, and a paramodulation into a clause in (iii) generates a clause in (iii).

Consider clause (ii.b). A paramodulation into clause (ii.a) generates the trivial clause $\mathsf{p}(x) \simeq \mathsf{p}(x)$, which is removed by the deletion rule, and a paramodulation into a clause in (iii) generates a clause in (iii).

Paramodulations from (iv): consider a clause in (iv.a). A paramodulation into a clause in (iv) or (v) generates a clause in (iv) or (v).

Consider a clause in (iv.b). A paramodulation into clause (ii.a) generates a clause in (iv.c); a paramodulation into a clause in (iii) or in (v) generates a clause in (v) and a paramodulation into a clause in (iv.b) generates a clause in (iv.a).

Consider a clause in (iv.c). A paramodulation into clause (ii.b) generates a clause in (iv.b); a paramodulation into a clause in (iii) or in (v) generates a clause in (v) and a paramodulation into a clause in (iv.c) generates a clause in (iv.a). ■

Thus, for the theory of integer offsets, we have:

Theorem 5.9 *A fair $\mathcal{SP}_{\succ}$-strategy is a polynomial satisfiability procedure for $\mathcal{I}$.*

Proof. Consider the input set of ground literals S and let m be the number of subterms occurring in S. This set can be flattened into a set S' in linear time in m. The cardinality of $\mathrm{C}_{S'}$, and therefore the number n of acyclicity axioms needed to form $S_0 = A_{\mathcal{I}} \cup Ac(n) \cup S'$, can be determined also in linear time in m, and the cardinality of S_0 is $O(m)$. Since all terms are constructed over a finite signature, by Lemma 5.8, the number of clauses that can be generated is bound by $O(m^2)$. Testing whether an inference rule can be applied requires a syntactic unification operation and verifying that the considered literals are indeed maximal. Such tests are polynomial in the sizes of the premises of the inference rule. Since the number of clauses is bound by $O(m^2)$, at most $O(m^4)$ such tests may be performed before applying an inference rule. Application of an inference rule is also polynomial in the sizes of its premises. Thus, the strategy is guaranteed to terminate in polynomial time, regardless of whether S is satisfiable. ■

We conclude our treatment of integer offsets with an example that shows that the number of clauses in a limit set S_∞ can be quadratic in the size of S.

Example 5.10 Let $S = \{\mathsf{s}(c_i) \simeq c'_i \mid i = 1, \ldots, m\}$, where the constants c_i and c'_i are all distinct; we therefore have $\mathrm{C}_S = m$. Let $S_0 = A_{\mathcal{I}} \cup Ac(m) \cup S$ and consider the limit S_∞ generated by a fair $\mathcal{SP}_{\succ}$-derivation from S_0. For $i, j \in \{1, \ldots, m\}$, the superposition of $\mathsf{s}(c_i) \simeq c'_i$ into $\mathsf{s}^j(x) \not\simeq x$ generates the *persistent* clause $\mathsf{s}^{j-1}(c'_i) \not\simeq c_i$ (recall that $\mathsf{s}^0(c) = c$ by convention); therefore, S_∞ contains at least $O(m^2)$ clauses.

This example shows that although the rewrite-based $\mathcal{I}$-satisfiability procedure is polynomial, it is not as efficient as the procedure of [28], which has complexity $O(m\log(m))$.

6 Discussion

In the first part of this article we presented a uniform approach to reduce the problem of deciding $\mathcal{T}$-satisfiability of ground clauses to that of deciding $\mathcal{T}$-satisfiability of ground literals, without reduction to disjunctive normal form. This approach is general in at least two ways: first, because we use generic theorem proving for first-order logic with equality; second, because the sufficient condition that presentation $\mathcal{T}$ needs to satisfy, termed variable-inactivity, is fulfilled by many theories of practical interest, including theories of data structures (e.g., arrays and records with or without extensionality, lists, whether cyclic or acyclic, possibly empty or non-empty) and fragments of arithmetic, such as the so-called theory of integer offsets.

The central result is a proof of termination: if $\mathcal{T}$ is variable-inactive, and the inference engine $\mathcal{SP}$ is guaranteed to terminate on $\mathcal{T}$-satisfiability problems, then $\mathcal{SP}$ is guaranteed to terminate also on $\mathcal{T}$-decision problems. Thus, the reduction gives a practical procedure: first, the input set of clauses is flattened, separating the unit part from the non-unit part; second, the theory axioms and the unit part are submitted to an $\mathcal{SP}$-based theorem prover, which generates a finite, saturated set; third, the application of the prover to the non-unit part of the original problem and the saturated set gives the answer. At least in principle, this is a recipe to use state-of-the-art theorem provers "off the shelf," as decision procedures for ground problems in several theories of interest for verification and their combinations, since the same variable-inactivity condition was already known to guarantee termination on combinations of theories, given termination on each.

In the second part of this article, we show how uniformity and generality of the rewrite-based approach can be combined with polynomial complexity of the resulting decision procedure. We gave polynomial rewrite-based $\mathcal{T}$-satisfiability procedures for the theory of records with extensionality and the theory of integer offsets. This shows that generic theorem provers can be efficient on problems made of ground literals.

A main direction for future work is to explore how to combine uniformity and generality with efficiency also on problems made of ground clauses. Indeed, most problems of practical interest in verification consist of huge non-unit clauses, and since generic theorem provers are not designed to deal with the boolean part of a formula as efficiently as possible, we would not expect them to perform well on such problems.

On one hand, techniques such as the one applied to design a polynomial $\mathcal{T}$-satisfiability procedure for records with extensionality may be promising. That technique shows that a theorem prover can be used to pre-process the problem, by generating enough information to decide a richer problem, with additional negative clauses. We intend to investigate generalizing this approach to problems that add different kinds of non-unit clauses, with the goal of decomposing problems in ways that allow one to design more efficient $\mathcal{T}$-decision procedures.

On the other hand, we may trade in some uniformity and generality for efficiency, by studying ways to interface theorem provers with SAT-solvers or SMT-solvers, where SMT stands for satisfiability modulo a theory [9]. First-order provers are strong at reasoning with equalities and with the universally quantified variables of the theory axioms. SAT-solvers and SMT-solvers are strong at reasoning with propositional logic and arithmetic. The reasoning environments of the future will have to harness the best of both kinds of engines.

Acknowledgements. The authors wish to thank the referees for their careful reading and detailed suggestions, that helped improve an earlier version of this article.

References

[1] Alessandro Armando, Maria Paola Bonacina, Silvio Ranise, and Stephan Schulz. New results on rewrite-based satisfiability procedures. *ACM TOCL*, in press.

[2] Alessandro Armando, Claudio Castellini, Enrico Giunchiglia, and Marco Maratea. A SAT-based decision procedure for the boolean combination of difference constraints. In *Online Proc. SAT-7*, 2004. `http://www.satisfiability.org/SAT04/`.

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

[4] Clark W. Barrett and Sergey Berezin. CVC Lite: a new implementation of the Cooperating Validity Checker. In Rajeev Alur and Doron A. Peled, editors, *Proc. CAV-16*, volume 3114 of *LNCS*, pages 515–518. Springer, 2004.

[5] Clark W. Barrett, David L. Dill, and Aaron Stump. Checking satisfiability of first-order formulas by incremental translation to SAT. In Kim G. Larsen and Ed Brinksma, editors, *Proc. CAV-14*, volume 2404 of *LNCS*, pages 236–249. Springer, 2002.

[6] Maria Paola Bonacina and Nachum Dershowitz. Abstract canonical inference. *ACM Trans. on Computational Logic*, 8(1):180–208, January 2007.

[7] Maria Paola Bonacina and Mnacho Echenim. Rewrite-based decision procedures. In Myla Archer, Thierry Boy de la Tour, and Cesar Munoz, editors, *Proc. 6th STRATEGIES Workshop, 4th FLoC, August 2006*, volume 174(11) of *ENTCS*, pages 27–45. Elsevier, July 2007.

[8] Maria Paola Bonacina and Mnacho Echenim. Rewrite-based satisfiability procedures for recursive data structures. In Byron Cook and Roberto Sebastiani, editors, *Proc. 4th PDPAR Workshop, 4th FLoC, August 2006*, volume 174(8) of *ENTCS*, pages 55–70. Elsevier, June 2007.

[9] Maria Paola Bonacina and Mnacho Echenim. T-decision by decomposition. In Frank Pfenning, editor, *Proc. CADE-21*, volume 4603 of *LNAI*, pages 199–214. Springer, 2007.

[10] Maria Paola Bonacina, Silvio Ghilardi, Enrica Nicolini, Silvio Ranise, and Daniele Zucchelli. Decidability and undecidability results for Nelson-Oppen and rewrite-based decision procedures. In Ulrich Furbach and Natarajan Shankar, editors, *Proc. 3rd IJCAR*, volume 4130 of *LNAI*, pages 513–527. Springer, 2006.

[11] Marco Bozzano, Roberto Bruttomesso, Alessandro Cimatti, Tommi Junttila, Peter van Rossum, Stephan Schulz, and Roberto Sebastiani. MathSAT: Tight integration of SAT and mathematical decision procedures. *J. of Autom. Reason.*, 35(1–3):265–293, Oct. 2005.

[12] Ricardo Caferra, Alexander Leitsch, and Nicholas Peltier. *Automated Model Building*. Kluwer Academic Publishers, 2004.

[13] Leonardo de Moura, Harald Rueß, and Maria Sorea. Lazy theorem proving for bounded model checking over infinite domains. In Andrei Voronkov, editor, *Proc. CADE-18*, volume 2392 of *LNAI*, pages 438–455. Springer, 2002.

[14] David Déharbe and Silvio Ranise. Light-weight theorem proving for debugging and verifying units of code. In *Proc. SEFM03*. IEEE Computer Society Press, 2003.

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

[16] David L. Detlefs, Greg Nelson, and James B. Saxe. Simplify: a theorem prover for program checking. *J. ACM*, 52(3):365–473, 2005.

[17] Bruno Dutertre and Leonardo de Moura. A fast linear-arithmetic solver for DPLL(T). In Tom Ball and R. B. Jones, editors, *Proc. CAV-18*, volume 4144 of *LNCS*, pages 81–94. Springer, 2006.

[18] Harald Ganzinger, George Hagen, Robert Nieuwenhuis, Albert Oliveras, and Cesare Tinelli. DPLL(T): Fast decision procedures. In Rajeev Alur and Doron A. Peled, editors, *Proc. CAV-16*, volume 3114 of *LNCS*, pages 175–188. Springer, 2004.

[19] Harald Ganzinger and David A. McAllester. A new meta-complexity theorem for bottom-up logic programs. In R. Goré, A. Leitsch, and T. Nipkow, editors, *Proc. 1st IJCAR*, volume 2083 of *LNCS*, pages 514–528. Springer, 2001.

[20] Silvio Ghilardi, Enrica Nicolini, and Daniele Zucchelli. A comprehensive framework for combined decision procedures. In Bernhard Gramlich, editor, *Proc. 5th FroCoS*, volume 3717 of *LNAI*, pages 1–30. Springer, 2005.

[21] Matt Kaufmann, Panagiotis Manolios, and J Strother Moore, Eds. *Computer Aided Reasoning : ACL2 Case Studies.* Kluwer Academic Publishers, 2000.

[22] Hélène Kirchner, Silvio Ranise, Christophe Ringeissen, and Duc-Khanh Tran. Automatic combinability of rewriting-based satisfiability procedures. In Miki Hermann and Andrei Voronkov, editors, *Proc. LPAR-13*, volume 4246 of *LNCS*, pages 542–556. Springer, 2006.

[23] Shuvendu K. Lahiri and Madanlal Musuvathi. An efficient decision procedure for UTVPI constraints. In Bernhard Gramlich, editor, *Proc. 5th FroCoS*, volume 3717 of *LNAI*, pages 168–183. Springer, 2005.

[24] Christopher Lynch and Barbara Morawska. Automatic decidability. In *Proc. LICS-17*, pages 7–16. IEEE Computer Society Press, 2002.

[25] Christopher Lynch and Duc-Khanh Tran. Automatic decidability and combinability revisited. In Frank Pfenning, editor, *Proc. CADE-21*, volume 4603 of *LNAI*, pages 328–344. Springer, 2007.

[26] G. Necula and P. Lee. Efficient representation and validation of proofs. In Vaughan Pratt, editor, *Proc. LICS-13*, pages 93–104. IEEE Computer Society Press, 1998.

[27] Greg Nelson and Derek C. Oppen. Simplification by cooperating decision procedures. *ACM TOPLAS*, 1(2):245–257, 1979.

[28] Robert Nieuwenhuis and Albert Oliveras. Fast congruence closure and extensions. *Information and Computation*, 205(4):557–580, April 2007.

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

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

[31] Sam Owre, John Rushby, N. Shankar, and David Stringer-Calvert. PVS: an experience report. In Dieter Hutter, Werner Stephan, Paolo Traverso, and Markus Ullman, editors, *Applied Formal Methods—FM-Trends 98*, volume 1641 of *LNCS*, pages 338–345. Springer, 1998.

[32] Stephan Schulz. E – a brainiac theorem prover. *J. of AI Communications*, 15(2–3):111–126, 2002.