On Superposition-Based Satisfiability Procedures and Their Combination

Hélène Kirchner, Silvio Ranise, Christophe Ringeissen, and Duc Khanh Tran

LORIA & INRIA-Lorraine

Abstract. We study how to efficiently combine satisfiability procedures built by using a superposition calculus with satisfiability procedures for theories, for which the superposition calculus may not apply (e.g., for various decidable fragments of Arithmetic). Our starting point is the Nelson-Oppen combination method, where satisfiability procedures cooperate by exchanging entailed (disjunction of) equalities between variables. We show that the superposition calculus deduces sufficiently many such equalities for convex theories (e.g., the theory of equality and the theory of lists) and disjunction of equalities for non-convex theories (e.g., the theory of arrays) to guarantee the completeness of the combination method. Experimental results on proof obligations extracted from the certification of auto-generated aerospace software confirm the efficiency of the approach. Finally, we show how to make satisfiability procedures built by superposition both incremental and resettable by using a hierarchic variant of the Nelson-Oppen method.

1 Introduction

Satisfiability procedures for theories of data types such as arrays, lists, and integers are at the core of many state-of-the-art verification tools. The task of designing, proving correct, and implementing satisfiability procedures is far from simple. One of the main problem is proving the correctness of satisfiability procedures. Furthermore, data structures and algorithms for each new procedure are implemented from scratch, with little software reuse and high risk of errors.

To overcome these difficulties, an approach to flexibly build satisfiability procedures based on superposition has been proposed in [2] and it has been shown competitive with *ad hoc* satisfiability procedures in [3,1]. Following this approach, the correctness proof of a procedure for a theory T reduces to show the termination of the fair and exhaustive application of the rules of the superposition calculus [12] on an axiomatization of T and an arbitrary set of literals. Furthermore, the implementation of the satisfiability procedure for T becomes easy by using (almost) off-the-shelf an available prover implementing the superposition calculus. In this way, years of careful engineering and debugging can be effortlessly reused. Unfortunately, this approach does not allow one to build satisfiability procedures for the fragments of Arithmetic which are required by most (if not all) verification problems. Hence, there is a need to combine satisfiability procedures obtained by superposition with satisfiability procedures for the various fragments of Arithmetic based on *ad hoc* techniques (see e.g., [8]).

D.V. Hung and M. Wirsing (Eds.): ICTAC 2005, LNCS 3722, pp. 594–608, 2005.

The method proposed by Nelson and Oppen (N-O) [11] allows one to combine satisfiability procedures for theories (satisfying some requirements) by exchanging equalities or disjunction of equalities between variables. Such equalities (or their disjunction) must be entailed by the input set of literals in each component theory. Since a set S of literals entails an equality (or a disjunction of equalities) ϕ if and only if the conjunction of S and the negation of ϕ is unsatisfiable, there does not seem to be any problem in using a satisfiability procedure based on superposition in a N-O combination. However, as it is well known (see e.g. [6]), to implement the combination method efficiently, the satisfiability procedure for the component theories must be capable of deriving the formulae to exchange with other procedures. This is not obvious for satisfiability procedures obtained by superposition since latter is not known to be complete for consequence finding, i.e. we are not guaranteed that a clause which is a logical consequence of a set of clauses will be eventually derived by applying the rules of the calculus. The **first contribution** of this paper is to show that satisfiability procedures obtained by superposition deduce sufficiently many equalities between variables for convex theories (e.g., the theory of lists) or disjunction of equalities between variables for non-convex theories (e.g., the theory of arrays) to guarantee the completeness of the N-O combination method.

The capability of detecting entailed equalities is not the only requirement to efficiently implement the N-O method: the component satisfiability procedures must be incremental and resettable, i.e. it must be possible to add and remove literals to and from the state of the procedure without restarting it. Actual state-of-the-art theorem provers based on superposition do not satisfy these two requirements and each time a literal is added or removed, provers must be invoked from scratch. This may result in an unacceptable overhead. To overcome this difficulty, the **second contribution** of this paper is to propose a hierarchic variant of the N-O combination method, where the superposition prover is used as a front-end of a congruence closure algorithm which is then combined with a satisfiability procedure for Arithmetic by the standard N-O method.

Our motivation for this work is to give a firm basis to a theorem proving system, called **haRVey** [3], which we are currently developing. Experimental results on a set of benchmarks [4] extracted from program verification problems clearly show the advantages of the proposed approach.

Plan of the paper. In Section 2, we introduce some basic notions. In Section 3, we show how to directly extract entailed (disjunction of) equalities between variables from satisfiability procedures built by superposition for various theories, we discuss some experimental results, and we conclude by describing a refinement of the N-O method. In Section 4, we discuss some related work. In Section 5, we conclude and sketch the future work. All omitted proofs can be found in [9].

2 Background

We assume the usual first-order syntactic notions of signature, term, position, and substitution, as defined, e.g., in [5]. If l and r are two terms, then $l = r$

is an *equality* and $\neg(l = r)$ (also written as $l \neq r$) is a *disequality*. A literal is either an equality or a disequality. A first-order *formula* is built in the usual way over the universal and existential quantifiers, Boolean connectives, and symbols in a given first-order signature. We call a formula *ground* if it has no variable. A *clause* is a disjunction of literals. A *unit* clause is a clause with only one disjunct, equivalently a literal. The *empty* clause is the clause with no disjunct, equivalently an unsatisfiable formula.

We also assume the usual first-order notions of model, satisfiability, validity, logical consequence, and theory. A *first-order theory* is a set of first-order formulae with no free variables. When T is a finitely axiomatized theory, $Ax(T)$ denotes the set of axioms of T. All the theories in this paper are first-order theories *with equality*, which means that the equality symbol $=$ is always interpreted as the equality relation. The theory of equality is denoted with $\mathcal{E}$. A formula is *satisfiable in a theory* T if it is satisfiable in a model of T. Two formulas φ and ψ are *equisatisfiable in* T if for every model $\mathcal{A}$ of T, φ is satisfiable in $\mathcal{A}$ iff ψ is satisfiable in $\mathcal{A}$. The *satisfiability problem* for a theory T amounts to establishing whether any given finite conjunction of literals (or equivalently, any given finite set of literals) is T-satisfiable or not. A *satisfiability procedure* for T is any algorithm that solves the satisfiability problem for T (the satisfiability of any quantifier-free formula can be reduced to the satisfiability of sets of literals by converting to disjunctive normal form and then splitting on disjunctions). The reader should observe that free variables in a formula φ behave as (Skolem) constants when φ is checked for satisfiability. In the rest of the paper, we use variables and constants interchangeably when the context allows us to do so, i.e. when combining satisfiability procedures.

2.1 The Superposition Calculus $\mathcal{SP}$

In the following, $=$ is (unordered) equality, $\equiv$ is identity, $\bowtie$ is either $=$ or $\neq$, l, r, u, t are terms, v, w, x, y, z are variables, all other lower case letters are constant or function symbols. The rules of the superposition calculus $\mathcal{SP}$ used in [2] and in this paper are depicted in Figures 1 and 2.

Given a set S of clauses, an expansion inference in Figure 1 adds the clause in its conclusion to S while a contraction inference rule in Figure 2 either simplifies (e.g. *Simplification* reduces to (ordered) rewriting when C is a unit clause) or deletes a clause from S. Notice that the premises and conclusion of an expansion rule are clauses while those of a contraction rule are sets of clauses. The rules in Figures 1 and 2 are well-known in the theorem proving literature (see e.g., [12]). A fundamental feature of $\mathcal{SP}$ is the usage of a *total reduction ordering (TRO)* $\succ$ [5] on terms. The ordering $\succ$ is extended to literals in such a way that only maximal sides of maximal instances of literals are considered when applying the expansion rules of Figure 1. Since later we need a total reduction ordering $\succ_c$ on clauses, we extend the TRO $\succ$ on terms to clauses as follows: $C \succ_c D$ if $ms(C)\ (\succ_{mul})_{mul}\ ms(D)$, where C and D are clauses, $\succ_{mul}$ is the multiset extension of the TRO $\succ$ over terms (see [5] for details), and $ms(s_1 \neq s'_1 \vee \ldots \vee s_n \neq s'_n \vee t_1 = t'_1 \vee \ldots t_m = t'_m)$ returns the multi-

Superposition (SP)	$\dfrac{\Gamma \Rightarrow \Delta, l[u'] = r \quad \Pi \Rightarrow \Sigma, u = t}{\sigma(\Gamma, \Pi \Rightarrow \Delta, \Sigma, l[t] = r)}(i), (ii), (iii), (iv)$
Paramodulation (PM)	$\dfrac{\Gamma, l[u'] = r \Rightarrow \Delta \quad \Pi \Rightarrow \Sigma, u = t}{\sigma(l[t] = r, \Gamma, \Pi \Rightarrow \Delta, \Sigma)}(i), (ii), (iii), (iv)$
Reflection (R)	$\dfrac{\Gamma, u' = u \Rightarrow \Delta}{\sigma(\Gamma \Rightarrow \Delta)} \forall L \in \Gamma \cup \Delta : \sigma(u' = u) \not\preceq \sigma(L)$
Eq. Factoring (EF)	$\dfrac{\Gamma \Rightarrow \Delta, u = t, u' = t'}{\sigma(\Gamma, t = t' \Rightarrow \Delta, u = t')}$ $(i), \ \forall L \in \Gamma : \sigma(u) \not\preceq \sigma(L),$ $\forall L \in \{u' = t'\} \cup \Delta : \sigma(u = t) \not\preceq \sigma(L)$

where a clause $\neg A_1 \vee \cdots \vee \neg A_n \vee B_1 \vee \cdots \vee B_n$ is written in sequent style as $\{A_1, \ldots, A_n\} \Rightarrow \{B_1, \ldots, B_m\}$ (where the A_i's and B_j's are literals), equality is the only predicate symbol, σ is the most general unifier of u and u', u' is not a variable in *Superposition* and *Paramodulation*, L is a literal, and the following hold:

(i) $\sigma(u) \not\preceq \sigma(t)$, ***(ii)*** $\forall L \in \Pi \cup \Sigma : \sigma(u = t) \not\preceq \sigma(L)$, ***(iii)*** $\sigma(l[u']) \not\preceq \sigma(r)$, and ***(iv)*** $\forall L \in \Gamma \cup \Delta : \sigma(l[u'] = r) \not\preceq \sigma(L)$.

Fig. 1. Expansion inference rules of $\mathcal{SP}$

Subsumption	$\dfrac{S \cup \{C, C'\}}{S \cup \{C\}}$	if for some substitution θ, $\theta(C) \subseteq C'$ and for no substitution ρ, $\rho(C') \equiv C$
Simplification	$\dfrac{S \cup \{C[l'], l = r\}}{S \cup \{C[\theta(r)], l = r\}}$	if $l' \equiv \theta(l)$, $\theta(l) \succ \theta(r)$, and $\forall L \in C[\theta(l)] : L \succ (\theta(l) = \theta(r))$
Deletion	$\dfrac{S \cup \{\Gamma \Rightarrow \Delta, t = t\}}{S}$	

where C and C' are clauses and S is a set of clauses.

Fig. 2. Contraction inference rules of $\mathcal{SP}$

set $\{\{s_1, s_1, s'_1, s'_1\}, \ldots, \{s_n, s_n, s'_n, s'_n\}, \{t_1, t'_1\}, \ldots, \{t_m, t'_m\}\}$. (By abuse of notation, we abbreviate $\succ_c$ with $\succ$.)

A clause C is *redundant* with respect to a set S of clauses if either $C \in S$ or S can be obtained from $S \cup \{C\}$ by a sequence of application of the contraction rules of Figure 2. An inference is *redundant* with respect to a set S of clauses if its conclusion is redundant with respect to S. A set S of clauses is *saturated* with respect to $\mathcal{SP}$ if every inference of $\mathcal{SP}$ with a premise in S is redundant with respect to S. A *derivation* is a sequence $S_0, S_1, \ldots, S_i, \ldots$ of sets of clauses where at each step an inference of $\mathcal{SP}$ is applied to generate and add a clause (cf. expansion rules in Figure 1) or to delete or reduce a clause (cf. contraction rules in Figure 2). A derivation is characterized by its *limit*, defined as the set of persistent clauses $S_\infty = \bigcup_{j \geq 0} \bigcap_{i > j} S_i$.

Lemma 1 ([12]). *Let $S_0, S_1, \ldots, S_n, \ldots$ be a derivation and let C be a clause in $(\bigcup_i S_i) \backslash S_\infty$. Then C is redundant with respect to S_∞.*

A derivation $S_0, S_1, ..., S_i, ...$ with limit S_∞ is *fair* with respect to $\mathcal{SP}$ if for every inference in $\mathcal{SP}$ with premises in S_∞, there is some $j \geq 0$ such that the inference is redundant in S_j.

Theorem 1 ([12]). *If $S_0, S_1, \ldots$ is a fair derivation of $\mathcal{SP}$, then (*i*) its limit S_∞ is saturated with respect to $\mathcal{SP}$, (*ii*) S_0 is unsatisfiable iff the empty clause is in S_j for some j, and (*iii*) if such a fair derivation is finite, i.e. it is of the form $S_0, \ldots, S_n$, then S_n is saturated and logically equivalent to S_0.*

We say that $\mathcal{SP}$ is *refutation complete* since it is possible to derive the empty clause with a finite derivation from an unsatisfiable set of clauses (cf. (*ii*) of Theorem 1).

2.2 A Superposition Approach to Satisfiability Procedures

The rewrite-based methodology [2] uses $\mathcal{SP}$ to build in an uniform way satisfiability procedures for theories which can be finitely axiomatized by a set of clauses. For a term t, $depth(t) = 0$, if t is a constant or a variable, and $depth(f(t_1, \ldots, t_n)) = 1 + max\{depth(t_i) \mid 1 \leq i \leq n\}$. A term is *flat* if its depth is 0 or 1. For a literal, $depth(l \bowtie r) = depth(l) + depth(r)$. A positive literal is *flat* if its depth is 0 or 1. A negative literal is *flat* if its depth is 0. A flat clause is a clause containing only flat literals. Let V be a set of variables. A *V-elementary equality* is an equality of the form $x = y$ for $x, y \in V$. A *V-elementary clause* is a disjunction of V-elementary equalities.

The rewrite-based methodology for T-satisfiability consists of two phases:

1. *Flattening:* all ground literals are flattened by introducing new constants, yielding an equisatisfiable T-reduced *flat* problem.
2. *Ordering selection and termination:* any fair derivation of $\mathcal{SP}$ is shown to be finite when applied to a T-reduced flat problem, provided that the TRO $\succ$ satisfies a few properties depending on T.

If T is a theory to which the rewrite-based methodology applies, a T-satisfiability procedure can be built by implementing the flattening (this can be done once and for all), and by using a prover mechanizing $\mathcal{SP}$ with a suitable TRO $\succ$. If the final set of clauses returned by the prover contains the empty clause, then the T-satisfiability procedure returns unsat; otherwise, it returns sat.

2.3 The Nelson-Oppen Method

The N-O combination method allows us to solve the problem of checking the satisfiability of a conjunction Φ of quantifier-free literals in the union of two signature-disjoint theories T_1 and T_2 for which two satisfiability procedures are available. Since the literals in Φ may be built over symbols in T_1 or in T_2, we need to *purify* them by introducing fresh variables to name subterms. This process leaves us with a conjunction $\Phi_1 \wedge \Phi_2$ which is equisatisfiable to Φ where Φ_i contains only literals with symbols of T_i, for $i = 1, 2$. In this way, literals in Φ_i can be *dispatched* to the available decision procedure for T_i.

To show the correctness of the N-O method [10,13], the theories T_1 and T_2 must be stably-infinite. Roughly, a theory is *stably infinite* if any satisfiable quantifier-free formula is satisfiable in a model having an infinite cardinality. All

theories considered in this paper (the theory of equality, the theory of lists, the theory of arrays, and the theory of Linear Arithmetic) are stably infinite.

An efficient description of the N-O method is based on the availability of satisfiability procedures with the following properties (see [6] for an in depth discussion on these issues):

Deduction completeness. It must be capable of efficiently detecting elementary clauses which are implied by the input conjunction of literals.

Incrementality & resettability. It must be possible to add and remove literals to and from the state of the procedure without restarting it. Also, processing each literal must be computationally cheap.

The N-O method for satisfiability procedures satisfying the requirements above is depicted in Figure 3 when T_1 is the theory of equality for which the superposition calculus is known to be a satisfiability procedure (see e.g., [2]) and T_2 is Linear Arithmetic (LA) for which various satisfiability procedures are available (see e.g., [8]). Such a combination method simply consists of exchanging elementary clauses between the two procedures until either unsatisfiability is derived by one of the two, or no more elementary clauses can be exchanged. In the first case, we derive the unsatisfiability of the input formula; in the second case, we derive its satisfiability. N-O method terminates because only finitely many elementary clauses can be constructed by using the variables of both Φ_1 and Φ_2.

It is sufficient to exchange only elementary equalities when combining convex theories. A theory is *convex* if for any conjunction Γ of equalities, a disjunction D of equalities is entailed by Γ if and only if some disjunct of D is entailed by Γ. Examples of convex theories are the theory of equality, the theory of lists, and the theory of Linear Arithmetic over the Rationals ($LA(\mathcal{R})$). Since both procedures are assumed to be deduction complete, the combination method only needs to

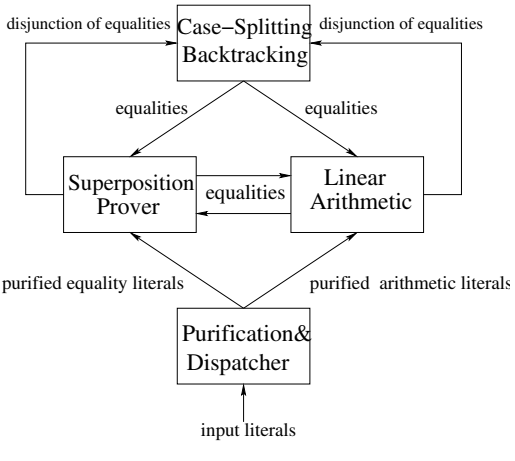

Fig. 3. The Nelson-Oppen Combination Method

pass around elementary equalities between the procedures as soon as they detect them. Adding the newly detected equalities can be done efficiently as long as the procedures are also assumed to be incremental.

When combining at least one non-convex theory such as the theory of arrays or the theory of Linear Arithmetic over the Integers ($LA(\mathcal{I})$), the combination method is more complex since the procedures should exchange elementary clauses. Although the procedures are capable of deriving the entailed elementary clauses, their processing is problematic since they are only capable of handling conjunctions of literals. The standard solution, as depicted in Figure 3, is to case-split on the derived elementary clauses and then consider each disjunct in turn by using a backtracking procedure; this can be efficiently done (see e.g., [6] for details) since both procedures are assumed to be incremental and resettable.

3 Deduction Complete, Incremental, and Resettable Satisfiability Procedures Based on Superposition

As discussed in Section 2.3, satisfiability procedures must be deduction complete, incremental, and resettable to be efficiently combined *à la* N-O. Here, we show how to extend the satisfiability procedures based on superposition of [2] with such capabilities.

3.1 Deduction Completeness

First, we need a formal definition of deduction completeness to precisely state our results.

Definition 1. *A T-satisfiability procedure is* deduction complete with respect to elementary clauses (resp. elementary equalities) *if for any T-satisfiable set S of clauses (resp. unit clauses), it returns, in addition to* sat*, a set of elementary clauses (resp. elementary equalities) D such that for any elementary clause (resp. elementary equality) C, we have $T \models S \Rightarrow C$ if and only if $D \models C$ (i.e. $S \Rightarrow C$ is T-valid if and only if C is a logical consequence of D).*

We now show how the methodology in [2] (summarized in Section 2.2) can be extended to build satisfiability procedures which are deduction complete w.r.t. elementary clauses. To this end, we must prove the following conjecture.

Conjecture 1. Let $Ax(T)$ be the set of axioms of a stably infinite theory T for which the methodology in [2] yields a satisfiability procedure, S be a T-satisfiable set of ground literals, and V be the set of all constants in S. If S' is the saturation of $Ax(T) \cup S$ and $D_V \subseteq S'$ is the set of all V-elementary clauses, then for every V-elementary clause C which is a logical consequence of $T \cup S$, C is a logical consequence of D_V.

If we are capable of proving Conjecture 1 for a certain theory T, we can build a deduction complete satisfiability procedure for T by simply extracting from a saturated set S' of clauses (not containing the empty clause) the elementary

clauses which entail all elementary clauses entailed by S'. Indeed, this is sufficient for the completeness of the N-O method depicted in Figure 3.

Since Conjecture 1 must be proved for each theory T for which the methodology of [2] (cf. Section 2.2) yields a superposition-based satisfiability procedure, we extend such a methodology with the following phase:

3. *Deduction completeness*: any set of clauses saturated by $\mathcal{SP}$ (not containing the empty clause) is shown to contain a set of elementary clauses entailing all elementary clauses which are logical consequences of the initial set of clauses.

Below, **we assume that $\succ$ is a TRO such that $t \succ c$ for each constant c and term t containing a function symbol of arity bigger than** 0. This requirement is easy to realize in practice (see [2] for more details). We also assume that the contraction rules of $\mathcal{SP}$ have higher priority than expansion rules. This is a reasonable assumption: before enlarging the search space by adding a new clause via the application of an expansion rule, one would try to reduce it as much as possible via the application of as many contraction rules as possible.

Theory of Equality $\mathcal{E}$. Let S be a set of ground literals and $Ax(\mathcal{E})$ be the empty set since equality is built-in the rules of $\mathcal{SP}$. For any TRO $\succ$, the fair and exhaustive applications of the rules of $\mathcal{SP}$ on $Ax(\mathcal{E}) \cup S$ always terminates and so $\mathcal{SP}$ can be used to build a satisfiability procedure for $\mathcal{E}$ (see [2] for details). Here, for the sake of generality, we consider S to be a set of clauses and not simply of literals.

Theorem 2. *Let S be a satisfiable set of ground clauses and V be the set of all constants occurring in S. Let S' be a saturation of S with respect to $\mathcal{SP}$ and D_V be the set of all V-elementary clauses in S'. Then for every V-elementary clause C, $S \models C$ implies $D_V \models C$.*

Proof. Since S and S' are logically equivalent, $S \models C$ if and only if $S' \models C$. Let $C \equiv c_1 = c'_1 \vee c_2 = c'_2 \vee \ldots \vee c_n = c'_n$, then $S' \models c_1 = c'_1 \vee c_2 = c'_2 \vee \ldots \vee c_n = c'_n$ is equivalent to $S' \cup \{c_1 \neq c'_1, c_2 \neq c'_2, \ldots, c_n \neq c'_n\}$ is unsatisfiable.

By the refutation completeness of $\mathcal{SP}$, we can derive the empty clause from $S' \cup \{c_1 \neq c'_1, c_2 \neq c'_2, \ldots, c_n \neq c'_n\}$ using the inference system $\mathcal{SP}$. Since S' is already saturated, only inferences between clauses in S' and clauses in $\{c_1 \neq c'_1, c_2 \neq c'_2, \ldots, c_n \neq c'_n\}$ could derive the empty clause. But then, only *Paramodulation*, *Simplification* and *Reflection* can apply since clauses in $\{c_1 \neq c'_1, \ldots, c_n \neq c'_n\}$ are all negative. Now let us analyze the form of clauses used by these three rules to derive the empty clause.

- *Paramodulation*: a clause of the form $c_1 = c'_1 \vee c_2 = c'_2 \vee \ldots \vee c_m = c'_m$ ($m \in \{1, \ldots, n\}$) in S' and $c_1 \neq c'_1$ are used to derive in one step $c_2 = c'_2 \vee \ldots \vee c_m = c'_m$. The rule repeatedly applies until the empty clause is obtained. That means the clause $c_1 = c'_1 \vee c_2 = c'_2 \vee \ldots \vee c_m = c'_m$ must be in S' and hence in D_V. But $c_1 = c'_1 \vee c_2 = c'_2 \vee \ldots \vee c_m = c'_m \models C$, consequently $D_V \models C$.

- *Simplification*: a clause $c_j \neq c'_j$ in $\{c_1 \neq c'_1, c_2 \neq c'_2, \ldots, c_n \neq c'_n\}$ is simplified (by one or many steps) to the clause $a \neq a$; and *Reflection* applies to derive the empty clause. But *Simplification* uses equalities to simplify clauses. In addition, these equalities must be elementary since they are used to simplify a disequality between constants. In other word $c_j = c'_j$ is a consequence of a set of elementary equalities which is itself a subset of D_V. Thus we have $D_V \models C$.
- *Reflection*: $a \neq a$ is used in *Reflection* and hence either $a \neq a \in S'$ or $a \neq a \in \{c_1 \neq c'_1, c_2 \neq c'_2, \ldots, c_n \neq c'_n\}$. But that type of clause cannot be in S' that is saturated and does not contain the empty clause. If it is in $\{c_1 \neq c'_1, c_2 \neq c'_2, \ldots, c_n \neq c'_n\}$, this simply means that $c_1 = c'_1 \vee c_2 = c'_2 \vee \ldots \vee c_n = c'_n$ is a tautology clause; and we have $D_V \models C$.

In all cases, we have $D_V \models C$. □

Corollary 1. *Let S be a satisfiable set of ground literals and V be the set of all constants occurring in S. Let S' be a saturation of S with respect to $\mathcal{SP}$ and Γ_V be the set of all V-elementary equalities in S'. Then for every V-elementary equality $c = c'$, $S \models c = c'$ implies $\Gamma_V \models c = c'$.*

Corollary 2. *$\mathcal{SP}$ is a deduction complete (with respect to elementary equalities) satisfiability procedure for $\mathcal{E}$.*

Proof. By Corollary 1, it is immediate to see that a deduction complete satisfiability procedure for $\mathcal{E}$ is obtained by computing a saturation of the input set of literals (unit clauses) and then collecting the elementary equalities in such a saturated set of clauses. □

Theory of Lists *à la* Shostak. The (convex) theory $\mathcal{L}$ of lists *à la* Shostak [16] is axiomatized by the following set $Ax(\mathcal{L})$ of axioms:

$$car(cons(X, Y)) = X \tag{L1}$$
$$cdr(cons(X, Y)) = Y \tag{L2}$$
$$cons(car(X), cdr(X)) = X \tag{L3}$$

where X and Y are implicitly universally quantified variables.

Lemma 2 ([2]). *Let S be a finite set of ground flat $\mathcal{L}$-literals. The clauses occurring in the saturation of $Ax(\mathcal{L}) \cup S$ w.r.t. $\mathcal{SP}$ are of the following types only, where X, Y are variables, a, b, c are constants, and $\bowtie \in \{=, \neq\}$:*

i) the empty clause;
ii) the axioms in $Ax(\mathcal{L})$: 1) $car(cons(X, Y)) = X$; 2) $cdr(cons(X, Y)) = Y$; and 3) $cons(car(X), cdr(X)) = X$;
iii) ground flat literals of the forms: 1) $c \bowtie c'$; 2) $car(a) = b$; 3) $cdr(a) = b$; and 4) $cons(a, b) = c$;
iv) equalities of the form $cons(b, cdr(a)) = a$ or $cons(car(a), b) = a$, where a, b are constants.

A consequence of this lemma [2] is that $\mathcal{SP}$ is a satisfiability procedure for $\mathcal{L}$. This is so because all saturations of $Ax(\mathcal{L}) \cup S$ are finite, since only finitely many literals of types i)–iv) can be built out of a finite signature.

Lemma 3. *Let S be a finite $\mathcal{L}$-satisfiable set of ground flat $\mathcal{L}$-literals and V be the set of constants occurring in S. Let S_g be the set of all ground clauses in the saturation of $Ax(\mathcal{L}) \cup S$ w.r.t. $\mathcal{SP}$. Then, for every V-elementary equality $c = c'$, we have that*
A) if $\mathcal{L} \cup S \models c = c'$ then $S_g \models c = c'$, and
B) $\mathcal{L} \cup S \cup \{c = c'\}$ is unsatisfiable iff $S_g \cup \{c = c'\}$ is unsatisfiable.

Proof. Let S' be a saturation of $Ax(\mathcal{L}) \cup S$. Since S is $\mathcal{L}$-satisfiable, S' does not contain the empty clause.

A) Notice that $\mathcal{L} \cup S \models c = c'$ is equivalent to $S' \cup \{c \neq c'\}$ is unsatisfiable; that also means we can derive the empty clause from $S' \cup \{c \neq c'\}$ using $\mathcal{SP}$. Since S' is saturated, we only consider inferences between clauses in S' and $c \neq c'$. For that, we consider all possible inferences between clauses listed in Lemma 2 and $c \neq c'$. We can easily see that only inferences between clauses of type (*iii.1*) and $c \neq c'$ are possible. In other words, $c = c'$ is a consequence of a subset of S_g; consequently, $S_g \models c = c'$.

B) We have that $\mathcal{L} \cup S \cup \{c = c'\}$ is unsatisfiable if and only if $S' \cup \{c = c'\}$ is unsatisfiable. Since S' is saturated, we only consider inferences between clauses in S' and $c = c'$. Again, we consider all possible inferences between clauses listed in Lemma 2 and $c = c'$ to derive the empty clause. We can see that only the following inferences are possible: (iii) and $c = c'$, (iv) and $c = c'$. This simply means $c = c'$ is a consequence of S_g. □

Theorem 3. *Let S be a finite $\mathcal{L}$-satisfiable set of ground flat $\mathcal{L}$-literals and V be the set of constants occurring in S. Let Γ_V be the set of V-elementary equalities that belong to the saturation of $Ax(\mathcal{L}) \cup S$ w.r.t. $\mathcal{SP}$. Then for every V-elementary equality $c = c'$ which is a logical consequence of $Ax(\mathcal{L}) \cup S$, $c = c'$ is a logical consequence of Γ_V.*

Proof. It follows from Lemma 3 that the subset S_g containing the ground clauses in the saturation of S is sufficient to derive V-elementary equalities, i.e. for every ground V-elementary equality $c = c'$, $S \models c = c'$ implies $S_g \models c = c'$. By Corollary 1, $S_g \models c = c'$ implies $\Gamma_V \models c = c'$. □

Corollary 3. *$\mathcal{SP}$ is a deduction complete (with respect to elementary equalities) satisfiability procedure for $\mathcal{L}$.*

Other convex theories. The result obtained for $\mathcal{L}$ can be extended to other convex theories considered in [2,1], namely the theory of lists *à la* N-O, a theory of encryption, or a theory of records by proceeding along the lines of what has been done for $\mathcal{L}$: show that only the ground clauses in a saturation are sufficient to derive entailed elementary equalities and then use Corollary 1 to conclude that the elementary equalities in the saturation are sufficient to derive all entailed elementary equalities. We do not do this here for lack of space.

Theory of Arrays. The (non-convex) theory $\mathcal{A}$ of arrays (see e.g., [2]) is axiomatized by the following finite set $Ax(\mathcal{A})$ of axioms:

$$select(store(A, I, E), I) = E \tag{A1}$$

$$I \neq J \Rightarrow select(store(A, I, E), J) = select(A, J) \tag{A2}$$

where A, I, J, E are implicitly universally quantified variables.

Lemma 4 ([2]). *Let S be a finite set of ground flat $\mathcal{A}$-literals. Then, the clauses occurring in the saturation of $Ax(\mathcal{A}) \cup S$ with respect to $\mathcal{SP}$ are of the following types only, where X is a variable, and $a, a', e, e', i, i_1, i'_1, \ldots, i_n, i'_n, j_1, j'_1 \ldots, j_m$, j'_m for $n \geq 0$, $m \geq 0$ are constants, and $\bowtie \in \{=, \neq\}$:*

i) the empty clause;
ii) the axioms in $Ax(\mathcal{A})$;
iii) ground flat literals;
iv) non-unit clauses of the form:
  a) clauses of the form $select(c, X) = select(c', X) \vee X = i_1 \vee \ldots X = i_n \vee j_1 \bowtie j'_1 \vee \ldots \vee j_m \bowtie j'_m$;
  b) $select(a, i) \bowtie e \vee i_1 \bowtie i'_1 \vee \ldots \vee i_n \bowtie i'_n$;
  c) $t = a' \vee i_1 \bowtie i'_1 \vee \ldots \vee i_n \bowtie i'_n$, where t is either a or $store(a, i, e)$;
  d) $e \bowtie e' \vee i_1 \bowtie i'_1 \vee \ldots \vee i_n \bowtie i'_n$;
  e) $i_1 \bowtie i'_1 \vee i_2 \bowtie i'_2 \vee \ldots \vee i_n \bowtie i'_n$;

A consequence of this lemma [2] is that $\mathcal{SP}$ is a satisfiability procedure for $\mathcal{A}$.

Lemma 5. *Let S be a finite $\mathcal{A}$-satisfiable set of ground flat $\mathcal{A}$-literals and V be the set of constants occurring in S. Let S_g be the set of all ground clauses in the saturation of $Ax(\mathcal{A}) \cup S$ with respect to $\mathcal{SP}$. Then, for every V-elementary clause D, we have that*
A) $Ax(\mathcal{A}) \cup S \models D \Rightarrow S_g \models D$, and
B) $Ax(\mathcal{A}) \cup S \cup \{D\}$ is unsatisfiable if and only if $S_g \cup \{D\}$ is unsatisfiable.

Theorem 4. *Let S be a finite $\mathcal{A}$-satisfiable set of ground flat $\mathcal{A}$-literals and V be the set of constant occurring in S. Let D_V be the set of V-elementary clauses that belong to the saturation of $Ax(\mathcal{A}) \cup S$ with respect to $\mathcal{SP}$. Then for every V-elementary clause C which is a logical consequence of $Ax(\mathcal{A}) \cup S$, C is a logical consequence of D_V.*

Proof. It follows from Lemma 5 that the subset S_g containing the ground clauses in the saturation of S is sufficient to derive V-elementary clauses, i.e. for every ground V-elementary clause C, $S \models C$ implies $S_g \models C$. By Theorem 2, $S_g \models C$ implies $D_V \models C$. □

Corollary 4. *$\mathcal{SP}$ is a deduction complete (with respect to elementary clauses) satisfiability procedure for $\mathcal{A}$.*

Other non-convex theories. The result obtained for $\mathcal{A}$ can be extended to the theory of arrays with extensionality and a simple theory of sets with and without extensionality considered in [2]. For lack of space, we do not develop this further.

3.2 Experiments

In order to show the efficiency of the deduction complete satisfiability procedures based on superposition presented above, we have implemented the combination method described in Figure 3 in the theorem prover **haRVey** [3]. In particular, the E prover [15] implements $\mathcal{SP}$ and we have implemented a module to inspect the saturated sets of clauses computed by the E prover and extract the elementary clauses. We have also implemented a deduction complete procedure for $LA(\mathcal{R})$ along the lines of [8].

For benchmarks, we used a selection of proof obligations from those generated to certify auto-generated aerospace software in [4]. We selected 107 (out of 356) unsatisfiable proof obligations expressing the property that each access to an array element are within the appropriate range. For example, an array variable `a` is modeled as the constant a and its i-element `a[i]` is written as $sel(a, i)$; hence, we need to reason about a combination of $\mathcal{E}$ and $LA(\mathcal{I})$. As it is common in software verification, we use the decision procedure for $LA(\mathcal{R})$ as a semi-decision procedure for $LA(\mathcal{I})$. On these benchmarks, this is sufficient since all proof obligations have been checked unsatisfiable already over the rationals. In [4], the proof obligations come with a set of axioms which approximates $LA(\mathcal{I})$ and should be sufficient to discharge (almost) all of them. Since **haRVey** is capable of handling virtually any theory which can be finitely axiomatized, we compared the behavior of the system **haRVey**($\mathcal{SP}$) with the E prover alone handling the supplied axioms for $LA(\mathcal{I})$ and the system **haRVey**($\mathcal{SP} + LA(\mathcal{R})$) featuring the combination between the decision procedure for $LA(\mathcal{R})$ and the superposition prover without axioms for $LA(\mathcal{I})$.

Table 1. Experimental results

	time-out	don't know	unsat
haRVey($\mathcal{SP}$)	5	17	85
haRVey($\mathcal{SP} + LA(\mathcal{R})$)	0	0	107

Experiments were performed on a Pentium-IV 2 GHz running Linux with 256 Kb of RAM and 1 Gb of disk space. We set a time-out of 60 seconds. A comparison of **haRVey**($\mathcal{SP}$) and **haRVey**($\mathcal{SP} + LA(\mathcal{R})$) is shown in Table 1. The column "don't know" means that the prover returned with satisfiable but since the axiomatization of $LA(\mathcal{I})$ is necessarily incomplete we interpreted it as non conclusive. From Table 1, it is clear that the incorporation of $LA(\mathcal{R})$ in **haRVey** is successful since it both eliminates the need of an explicit axiomatization of the background theory and makes the system more reliable. We believe that these results clearly show that Arithmetic reasoning has been efficiently combined with superposition theorem proving to discharge the proof obligations arising in typical software verification problems.

3.3 Incrementality and Resettability

Although the combination method in Figure 3 is already efficient in practice to tackle interesting proof obligations arising in verification (as shown in Sec-

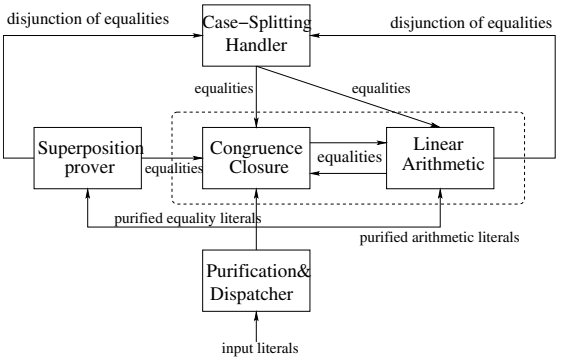

Fig. 4. The Hierarchic Nelson-Oppen Combination Method

tion 3.2), there is still room for improvement. In fact, Lemmas 3.B) and 5.B) allow us to observe that when new elementary clauses (deduced from another satisfiability procedure) must be added to a saturated set of clauses, it is only necessary to consider the ground clauses in the saturated set. This implies that it is sufficient to use the superposition calculus as a front-end for the congruence closure algorithm, which can be turned into a deduction complete, incremental, and resettable satisfiability procedure for $\mathcal{E}$ (see e.g., [6]). So, the superposition calculus must be applied only once before the combination loop in which the congruence closure algorithm and another satisfiability procedure exchange elementary clauses. Figure 4 depicts the hierarchic combination method which allows us to obtain satisfiability procedures which are both incremental and resettable. We are currently implementing the method in **haRVey** and we expect further improvements in performances. It is interesting to notice that this approach can be used in any theorem proving system featuring a combination of satisfiability procedures *à la* N-O, offering an easy and efficient way to incorporate procedures for a variety of theories extending $\mathcal{E}$.

4 Related Work

Our approach to efficiently combine a theory processed by superposition with a procedure for $LA(\mathcal{R})$ is based on the N-O method. An alternative combination method has been proposed by Shostak [16]. Such a method assumes that the theories to be combined are such that there exist functions for reducing terms to canonical form (*canonizers*) and for solving equations (*solvers*) [13]. There are essentially two different ways to use canonizers and solvers for deciding the satisfiability problem in unions of disjoint theories.

First, one can use a solver and a canonizer to build a satisfiability procedure having the capability of computing entailed elementary equalities. Then, this satisfiability procedure can be combined with others using the N-O method. So, combining theories à la Shostak can be directly viewed as a refinement of the Nelson-Oppen combination method [13]. In this way, solvers and canonizers

can be readily integrated with the satisfiability procedures based on superposition described in this paper. A second approach consists in extending the use of canonizers and solvers in order to deal with terms built over the union of the signatures of the component theories. In contrast to the N-O method, one does not need to purify the input literals. Rather, the input literals are processed directly by solvers and canonizers having the capability of transforming heterogeneous terms. This approach was initiated by Shostak and has been followed by many other papers revisiting this combination method (see again [13] for details). Recently, this approach has been used in [7] to integrate a canonizer and a solver in the superposition calculus. This yields a refutationally complete calculus on *ground* clauses whose terms are built over the union of the signatures of the component theories. This is particularly interesting to integrate some form of Arithmetic reasoning with superposition. The main drawback of this approach is the ordering relation used to restrict the applicability of the inference rules which is quite complex. Instead, our approach uses the standard and well understood framework of the superposition calculus (including the standard techniques to define ordering relations) which allows us to re-use a wide range of existing results.

5 Conclusion

In [2], the authors give a general and flexible approach to derive satisfiability procedures by superposition. In this work, we have shown that such satisfiability procedures deduce sufficiently many (disjunctions of) equalities between variables to be combined *à la* Nelson and Oppen with other satisfiability procedures without loosing completeness. Experimental results on typical software verification problems show the efficiency of the proposed approach. Moreover, it is possible to obtain a certain degree of incrementality and resettability by using a hierarchic variant of the N-O method.

There are several main lines for future work. First, we want to derive a more precise characterization of the theories for which deduction complete superposition based satisfiability procedures can be built with the methodology of [2]. Second, we intend to empirically evaluate the efficiency of our hierarchic variant of N-O combination method by conducting some experiments in **haRVey** [3]. Finally, we plan to study, along the line of [14], the combination of superposition based satisfiability procedures with satisfiability procedures for non stably infinite theories, for which the N-O method does not directly apply.

References

1. A. Armando, M. P. Bonacina, S. Ranise, and S. Schulz. On a rewriting approach to satisfiability procedures: extension, combination of theories and an experimental appraisal. In *Proc. of the 5th Int. Workshop on Frontiers of Combining Systems (FroCos'05)*, LNCS. Springer-Verlag, 2005. To appear.
2. A. Armando, S. Ranise, and M. Rusinowitch. A Rewriting Approach to Satisfiability Procedures. *Info. and Comp.*, 183(2):140–164, June 2003.

3. D. Déharbe and S. Ranise. Light-Weight Theorem Proving for Debugging and Verifying Units of Code. In *Proc. of the Int. Conf. on Software Engineering and Formal Methods (SEFM03)*. IEEE Comp. Soc. Press, 2003.
4. E. Denney, B. Fischer, and J. Schumann. Using automated theorem provers to certify auto-generated aerospace software. In *Proc. of Int. Joint Conf. On Automated Reasoning (IJCAR'04)*, volume 3097 of *LNCS*, 2004.
5. N. Dershowitz and J.-P. Jouannaud. *Handbook of Theoretical Computer Science*, volume B, chapter 6: Rewrite Systems, pages 244–320. Elsevier Science Publishers B. V. (North-Holland), 1990.
6. D. Detlefs, G. Nelson, and J. B. Saxe. Simplify: A Theorem Prover for Program Checking. Technical Report HPL-2003-148, HP Laboratories, 2003.
7. H. Ganzinger, T. Hillenbrand, and U. Waldmann. Superposition modulo a Shostak theory. In F. Baader, editor, *Automated Deduction — CADE-19*, volume 2741 of *LNAI*, pages 182–196. Springer-Verlag, 2003.
8. D. Kapur and X. Nie. Reasoning about Numbers in Tecton. In *Proc.* 8^{th} *Inl. Symp. Methodologies for Intelligent Systems*, pages 57–70, 1994.
9. H. Kirchner, S. Ranise, C. Ringeissen, and D. K. Tran. On Superposition-Based Satisfiability Procedures and their Combination (Full Version). Available at `http://www.loria.fr/~ranise/pubs/long-ictac05.ps.gz`.
10. G. Nelson. Techniques for program verification. Technical Report CS-81-10, Xerox Palo Research Center California USA, 1981.
11. G. Nelson and D. C. Oppen. Simplification by cooperating decision procedures. *ACM Trans. on Programming Languages and Systems*, 1(2):245–257, Oct. 1979.
12. R. Nieuwenhuis and A. Rubio. Paramodulation-based theorem proving. In A. Robinson and A. Voronkov, editors, *Hand. of Automated Reasoning*. 2001.
13. S. Ranise, C. Ringeissen, and D.-K. Tran. Nelson-Oppen, Shostak and the Extended Canonizer : A Family Picture with a Newborn. In *First International Colloquium on Theoretical Aspects of Computing — ICTAC 2004, Guiyang, China*, volume 3407 of *LNCS*, pages 372–386. Springer, Sep 2004.
14. S. Ranise, C. Ringeissen, and C. G. Zarba. Combining data structures with nonstably infinite theories using many-sorted logic. In *Proc. of the 5th Int. Workshop on Frontiers of Combining Systems (FroCos'05)*, LNCS. Springer-Verlag, 2005. To appear.
15. S. Schulz. E – A Brainiac Theorem Prover. *Journal of AI Communications*, 15(2/3):111–126, 2002.
16. R. E. Shostak. Deciding combinations of theories. *J. of the ACM*, 31:1–12, 1984.